Online Interval Scheduling with Predictions *

Joan Boyar† Lene M. Favrholdt†
Shahin Kamali‡ Kim S. Larsen†

†University of Southern Denmark
https://imada.sdu.dk/u/{joan,lenem,kslarsen}

‡York University
https://www.eecs.yorku.ca/~kamalis/

January 5, 2025

Abstract

In online interval scheduling, the input is an online sequence of intervals, and the goal is to accept a maximum number of non-overlapping intervals. In the more general disjoint path allocation problem, the input is a sequence of requests, each consisting of pairs of vertices of a known graph, and the goal is to accept a maximum number of requests forming edge-disjoint paths between accepted pairs. We study a setting with a potentially erroneous prediction specifying the set of requests forming the input sequence and provide tight upper and lower bounds on the competitive ratios of online algorithms as a function

*Boyar, Favrholdt, and Larsen were supported in part by the Danish Council for Independent Research grants DFF-0135-00018B and DFF-4283-00079B, and Kamali was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC). A preliminary extended abstract of some of these results was published by the same authors in the 18th International Algorithms and Data Structures Symposium (WADS), volume 14079 of Lecture Notes in Computer Science, pages 193-207. Springer, 2023.

of the prediction error. We also present asymptotically tight trade-offs between consistency (competitive ratio with error-free predictions) and robustness (competitive ratio with adversarial predictions) of interval scheduling algorithms. Finally, we provide experimental results on real-world scheduling workloads that confirm our theoretical analysis.

1 Introduction

In the *interval scheduling problem*, the input is a set of intervals with integral endpoints, each representing timesteps at which a process starts and ends. A scheduler's task is to decide whether to accept or reject each job so that the intervals of accepted jobs do not overlap except possibly at their endpoints. The objective is to maximize the number of accepted intervals, referred to as the *profit* of the scheduler. This problem is also known as *fixed job scheduling* and *k-track assignment* [39].

Interval scheduling is a special case of the *disjoint path allocation problem*, where the input is a graph G and a set of n *requests*, each defined by a pair of vertices in G. An algorithm can accept or reject each pair, given that it can form a path between the vertices of each of the accepted pairs, such that all of the paths are edge-disjoint. Interval scheduling is the particular case when G is a path graph. The disjoint path allocation problem can be solved in polynomial time for trees [33] and outerplanar graphs by a combination of [52, 48, 32], but the problem is NP-complete for general graphs [31], and even on quite restricted graphs such as series-parallel graphs [50]. The disjoint path problem is the same as call control/call allocation with all calls having unlimited duration and all bandwidths (both of the calls and the edges they would be routed on) being equal to 1 and as the maximum multi-commodity integral flow problem with edges having unit capacity.

In this work, we focus on the *online* variant of the problem, in which the set of requests is not known in advance but is revealed in the form of a request sequence, I. Each new request must either be irrevocably accepted or rejected. On acceptance, the algorithm selects and commits to a particular path, edge-disjoint from any previously selected path. We analyze an online algorithm via a comparison with an optimal offline algorithm, Opt. An online algorithm Alg is *c-competitive*, if there exists a constant b such that, for any input graph G and any sequence I of requests on G, $\textsc{Alg}(I) \geq$

$c \cdot \textsc{Opt}(I) - b$, where $\textsc{Alg}(I)$ and $\textsc{Opt}(I)$, respectively, denote the profit of Alg and Opt on I (for randomized algorithms, $\textsc{Alg}(I)$ is the expected profit of Alg). If $b = 0$, Alg is *strictly c-competitive*. The *competitive ratio* of Alg is defined as $\sup\{c \mid \textsc{Alg} \text{ is } c\text{-competitive}\}$ and its *strict competitive ratio* is $\sup\{c \mid \textsc{Alg} \text{ is strictly } c\text{-competitive}\}$. Since we consider a maximization problem, our ratios are between zero and one. Our results are strongest possible in the sense that all lower bounds (positive results) are valid with respect to the strict competitive ratio and all upper bounds (negative results) are valid with respect to the (non-strict) competitive ratio.

In interval scheduling, there is a distinction between the *any-order* setting [20], where the adversary determines the ordering of intervals, and the *sorted-order* [45] setting, where intervals arrive in order of their starting times. The focus of this paper is on the any-order setting. Since the positive results presented in the paper are established for the any-order setting, they also hold for the sorted-order setting. Note that, in contrast to [20], we do not allow preemption.

For interval scheduling on a path graph with m edges, the *strict* competitive ratio of the best deterministic algorithm is $\frac{1}{m}$ [19], and no deterministic algorithm can be better than $\frac{1}{\sqrt{m}}$-competitive as seen by the following adversarial strategy. First $\sqrt{m}$ disjoint intervals of length $\sqrt{m}$ are given. Then, for each interval S accepted by the algorithm, $\sqrt{m}$ disjoint unit intervals overlapping S are given. Note that, in [19], intervals model calls of limited duration. In that model, sequences can be repeated an arbitrary number of times, and thus, there is no difference between the strict and the non-strict competitive ratio. The best randomized algorithm has a competitive ratio of $\Theta(\frac{1}{\log m})$ [19]. For this result, the upper bound (negative result) proof of [19] also holds for our model. Moreover, the upper bound follows from our Theorem 6 with $\alpha = 0$. These results suggest that the constraints on online algorithms must be relaxed to compete with Opt. Specifically, the problem has been considered in the *advice complexity model* for path graphs [14, 34], trees [15], and grid graphs [17]. Under the advice model, the online algorithm can access error-free information on the input called advice. The objective is to quantify the trade-offs between the competitive ratio and the size of the advice. Another relaxation of the online model is to consider *priority algorithms* [21], where the algorithm is allowed to give priorities to the entire set of possible input items and always receive the input with highest priority next. The algorithm processes the items it receives in an online manner. Pri-

ority algorithms are a model for greedy algorithms that has also been studied for graph problems, in particular [18, 29]. Priority algorithms for the disjoint path allocation problem for path graphs, trees, and grid graphs are studied in [16], along with priority algorithms in a model that includes advice [24].

In recent years, there has been an increasing interest in improving the performance of online algorithms via the notion of *prediction*. Here, it is assumed that the algorithm has access to a prediction, for instance in the form of machine-learned information. Unlike the advice model, the prediction may be erroneous and is quantified by an *error measure* η. The objective is to design algorithms whose competitive ratio degrades gently as a function of η. Several online optimization problems have been studied under the prediction model, including non-clairvoyant scheduling [40, 54], makespan scheduling [41], contract scheduling [4, 2], and other variants of scheduling problems [9, 44, 12, 11].

Other online problems studied under the prediction model include bin packing [3], knapsack [55, 36, 23], caching [47, 51, 53, 6], matching problems [7, 42, 43], time series search [5], and various graph problems [28, 30, 27, 10, 13]. See also the survey by Mitzenmacher and Vassilvitskii [49] and the collection at [1].

1.1 Contributions

We study the online disjoint path allocation problem under a setting where the scheduler is provided with a set $\hat{I}$ of requests predicted to form the input sequence I. Given the erroneous nature of the prediction, some requests in $\hat{I}$ may be incorrectly predicted to be in I (false positives), and some requests in I may not be included in $\hat{I}$ (false negatives). We let the *error set* be the set of requests that are false positives or false negatives and define the error parameter $\eta(\hat{I}, I)$ to be the cardinality of the largest set of requests in the error set that can be accepted. For interval scheduling, this is the largest set of non-overlapping intervals in the error set. Thus, $\eta(\hat{I}, I) = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$, where FP and FN are the sets of false positives and negatives, respectively. We explain later that this definition of η has specific desired properties for the prediction error (Proposition 1). In the following, we use $\textsc{Alg}(\hat{I}, I)$ to denote the profit of an algorithm Alg for prediction $\hat{I}$ and input I. We also define $\gamma(\hat{I}, I) = \eta(\hat{I}, I)/\textsc{Opt}(I)$; this *normalized error* measure is helpful in describing our results because the point of reference in the competitive analysis is $\textsc{Opt}(I)$. Our first result concerns general graphs.

Disjoint-Path Allocation

We study a simple algorithm Trust, which accepts a request only if it belongs to a given optimal solution for $\hat{I}$. We show that, Trust is strictly $(1 - 2\gamma)$-competitive (Theorem 1). Furthermore, this is best possible among deterministic algorithms, even on the graph class trees (Theorem 2).

The above result demonstrates that even for trees, the problem is so hard that no algorithm can do better than the trivial Trust. Therefore, our main results concern the more interesting case of path graphs, that is, interval scheduling:

Interval Scheduling

We first show that no deterministic interval scheduling algorithm can be better than $(1 - \gamma)$-competitive (Theorem 3).

Next, we show that Trust is no better on the path than on trees; its competitive ratio is only $1 - 2\gamma$ (Theorem 4). This suggests that there is room for improvement over Trust.

Finally, we introduce our main technical result, a deterministic algorithm TrustGreedy that achieves an optimal competitive ratio of $1 - \gamma$ for interval scheduling (Theorem 5). TrustGreedy is similar to Trust in that it maintains an optimal solution for $\hat{I}$, but unlike Trust, it updates its planned solution to accept requests greedily when it is possible without a decrease in the profit of the maintained solution.

Consistency-Robustness Trade-off

We study the trade-off between *consistency* and *robustness*, which measure an algorithm's competitive ratios in the extreme cases of error-free prediction (consistency) and adversarial prediction (robustness) [46, 47]. We show that no deterministic algorithm with a constant consistency can have robustness $\omega(1/m)$. Thus, we focus on the more interesting case of randomized algorithms. (Proposition 2). Suppose that for any input I, an algorithm Alg guarantees a consistency of α and robustness of $\beta \geq \frac{1}{m^{1-\varepsilon}}$, $\varepsilon > 0$.

We show that

$$\alpha \leq 1 - \beta\left(\frac{\varepsilon \log m}{2} - O(1)\right) \text{ and}$$
$$\beta \leq (1-\alpha)\frac{2}{\varepsilon \log m} + O\left(\frac{1}{\log^2 m}\right)$$

(Theorem 6). Note that, if $\beta \in \Omega(\frac{1}{\log m})$, the constraint $\beta \geq \frac{1}{m^{1-\varepsilon}}$ is fulfilled for any $\varepsilon < 1$. Thus, for example, to guarantee a robustness of $\frac{1}{5 \log m}$, the consistency must be at most $\frac{9}{10} + O(\frac{1}{\log m})$, and to guarantee a consistency of $\frac{9}{10}$, the robustness must be at most $\frac{1}{5 \log m} + O(\frac{1}{\log^2 m})$. We also present a family of randomized algorithms that provides an almost *Pareto-optimal* trade-off between consistency and robustness (Theorem 7).

Experiments on Real-World Data

We compare our algorithms with Opt and the online Greedy algorithm (which accepts an interval if and only if it does not overlap previously accepted intervals) on real-world scheduling data from [25]. Our results are in line with our theoretical analysis: both Trust and TrustGreedy are close-to-optimal for small error values; TrustGreedy is almost always better than Greedy even for large values of error, while Trust is better than Greedy only for small error values.

Matching and Independent Set

We explain that our results on disjoint-path allocation carry over to matching in the edge-arrival model (Corollary 2) and to independent set on line graphs in the vertex-arrival model (Corollary 3).

2 Model and Predictions

Throughout the paper we let m denote the number of edges in the graph and let n denote the number of requests in the input sequence.

We assume that an oracle provides the online algorithm with a set $\hat{I}$ of requests predicted to form the input sequence I. This type of prediction arises naturally in scenarios such as call admission, where the input sequence represents the calls made on a given day. By analyzing historical data from

previous days, one can predict, albeit with some error, whether two nodes in the call network will initiate a call on that day.

One may consider alternative predictions, such as statistical information about the input. While these predictions are compact and can be efficiently learned, they cannot help achieve close-to-optimal solutions. In particular, for interval scheduling on a path with m edges, since the problem is AOC-complete, one cannot achieve a competitive ratio $c \leq 1$ with fewer than $cn/(e \ln 2)$ bits [22], even if all predictions are correct [34]. In particular, to achieve a competitive ratio $c \in \Omega(1/\log m)$—the best attainable by randomized algorithms without prediction—one requires a prediction of size $\Omega(n/\log m)$, which grows linearly with the input length (and thus cannot merely be statistical information).

2.1 Error Measure

In what follows, *true positive* (respectively, *negative*) intervals are correctly predicted to appear (respectively, not to appear) in the request sequence. *False positives* and *negatives* are defined analogously as those incorrectly predicted to appear or not appear. We let TP, TN, FP, FN denote the four sets containing these different types of intervals. Thus, $I = \mathrm{TP} \cup \mathrm{FN}$ and $\hat{I} = \mathrm{TP} \cup \mathrm{FP}$. We use $\eta(\hat{I}, I)$, to denote the error for the input formed by the sequence I, when the prediction is the set $\hat{I}$. When there is no risk of confusion, we use η instead of $\eta(\hat{I}, I)$. The error measure we use here is

$$\eta = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}),$$

and hence, the normalized error measure is

$$\gamma = \frac{\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})}{\textsc{Opt}(I)}.$$

Our error measure has three desirable properties (see below), the first two of which were recommended in Im et al. [37]. The first property ensures that improving the prediction does not lead to a larger error. The other two ensure that the error is neither too small nor too large, which is crucial for distinguishing between good and bad algorithms.

In Section 2.1.1, we discuss other natural error models, such as the Hamming distance between the request sequence and prediction, and explain why these measures do not have our desired properties.

Monotonicity

This property ensures that improving the prediction to increase the number of true positives or negatives does not increase the error. To be more precise, if we increase $|\,\mathrm{TP}\,|$ by one unit (decreasing $|\,\mathrm{FN}\,|$ by one unit) or increase $|\,\mathrm{TN}\,|$ by one unit (decreasing $|\,\mathrm{FP}\,|$ by one unit), the error must not increase. Formally, for any I, $\hat{I}$, the following must hold.

- For any $x \in I \setminus \hat{I}$, $\eta(I, \hat{I} \cup \{x\}) \leq \eta(\hat{I}, I)$.
- For any $y \in \hat{I} \setminus I$, $\eta(I, \hat{I} \setminus \{y\}) \leq \eta(\hat{I}, I)$.

Lipschitz property

This property requires the error to be at least equal to the net difference between $\textsc{Opt}(I)$ and $\textsc{Opt}(\hat{I})$, that is,

$$\eta(\hat{I}, I) \geq |\,\textsc{Opt}(I) - \textsc{Opt}(\hat{I})|\,.$$

Note that the Lipschitz property ensures that the error is not "too small", causing all algorithms to have a bad competitive ratio, even when the error, according to the error measure, is low. To be able to distinguish between good and mediocre algorithms, we must also avoid that the error becomes "too large". Hence, we also define a notion of Lipschitz-completeness, imposing an upper bound on the error. The particular upper bound that we use is motivated by the following example.

Example 1 The input is formed by a set $I = A \cup B$ of requests, with $A = \{A_1, A_2, \ldots, A_k\}$ and $B = \{B_1, B_2, \ldots, B_{k-1}\}$, where the A_i's are disjoint, the B_i's are disjoint, and B_i overlaps A_i and A_{i+1}. The profit of the optimal solution is then $\textsc{Opt}(I) = |A| = k$. Suppose the prediction is $\hat{I} = (A \setminus \{A_1, A_2\}) \cup B$, and note that $\textsc{Opt}(\hat{I}) = |B| = k - 1$. The optimal solutions for I and $\hat{I}$ are disjoint but $|\,\textsc{Opt}(I) - \textsc{Opt}(\hat{I})| = 1$, $\mathrm{FP} = 0$ and $\mathrm{FN} = 2$ (Figure 1). □

In this example, the error should be relatively small, independent of k. More generally, the error measure must not grow with the dissimilarity between the optimal solutions for I and $\hat{I}$, but rather with the size of the optimal solution for FP and FN. This is guaranteed by the following property.

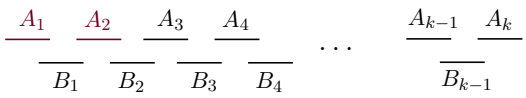


Figure 1: An illustration of Example 1. All requests appear in both I and $\hat{I}$, except $\{A_1, A_2\}$, which are false negatives.

Lipschitz-completeness

An error measure is Lipschitz-complete, if for any I, $\hat{I}$, the following holds.

$$\eta(I, \hat{I}) \leq \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}).$$

Proposition 1 The error measure $\eta(\hat{I}, I) = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$ is monotone, Lipschitz, and Lipschitz-complete.

Proof We check all properties listed above. In all cases, we leverage the straightforward monotonicity property of Opt that the optimal profit of an input I is no greater than the optimal profit of $I \cup \{x\}$ for any item x, since Opt can always choose to not include x in the solution.

- Monotonicity: First, consider increasing the number of true positives. Let $x \in I \setminus \hat{I}$. Since x is a false negative, it may or may not have been counted in $\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$, but removing it from FN (thus adding it to TP) cannot make $\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$ larger, i.e.,

  $$\eta(I, \hat{I} \cup \{x\}) = \textsc{Opt}(\mathrm{FP} \cup (\mathrm{FN} \setminus \{x\})) \leq \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = \eta(I, \hat{I})\,.$$

  Similarly, for any $y \in \hat{I} \setminus I$, $\textsc{Opt}((\mathrm{FP} \setminus \{y\}) \cup \mathrm{FN})$ cannot be larger than $\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = \eta(I, \hat{I})$, so

  $$\eta(I, \hat{I} \setminus \{y\}) = \textsc{Opt}((\mathrm{FP} \setminus \{y\}) \cup \mathrm{FN}) \leq \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = \eta(I, \hat{I})\,.$$

- Lipschitz property: We need to show that

  $$\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) \geq |\,\textsc{Opt}(I) - \textsc{Opt}(\hat{I})|\,.$$

  We note that

  $$\begin{aligned} \textsc{Opt}(I) &= \textsc{Opt}((\hat{I} \setminus \mathrm{FP}) \cup \mathrm{FN}) \\ &\leq \textsc{Opt}(\hat{I} \cup \mathrm{FN}) \\ &\leq \textsc{Opt}(\hat{I}) + \textsc{Opt}(\mathrm{FN})\,, \end{aligned}$$

which implies

$$\textsc{Opt}(I) - \textsc{Opt}(\hat{I}) \leq \textsc{Opt}(\mathrm{FN}) \leq \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}).$$

- Lipschitz-completeness: Follows trivially with the suggested bound, since $\eta = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$.

□

2.1.1 Alternative Error Measures.

In what follows, we review a few alternative error measures that do not have all of our desired properties of monotonicity, Lipschitz, and Lipschitz-completeness.

- Hamming distance between the bit strings representing the request sequence and the predictions, that is, the total number of false positives and false negatives:

  $$|\,\mathrm{FP}\,| + |\,\mathrm{FN}\,| = |I \cup \hat{I}\,| - |I \cap \hat{I}\,| = |(I \cup \hat{I}) \setminus (I \cap \hat{I})|$$

  This measure fails Lipschitz-completeness. For instance, consider $I = \{(1, m)\}$ and $\hat{I} = \bigcup_{i=1}^{m-1}(i, m)$. In this case, $\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = 1$ (any pair of intervals intersect) and $|\,\mathrm{FP}\,| + |\,\mathrm{FN}\,| = m - 2$. Thus, $|\,\mathrm{FP}\,| + |\,\mathrm{FN}\,| \not\leq \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$.

  Note that using either $|\,\mathrm{FP}\,|$ or $|\,\mathrm{FN}\,|$ alone also fails the Lipschitz-completeness property, by the same example.

- Let $\textsc{Opt}[I]$ and $\textsc{Opt}[\hat{I}]$ denote optimal solutions to the instances formed by I and $\hat{I}$, respectively. Using $\textsc{Opt}[I]$ and $\textsc{Opt}[\hat{I}]$ instead of I and $\hat{I}$ in the above measure:

  $$\big|\big(\textsc{Opt}[I] \cup \textsc{Opt}[\hat{I}]\big) \setminus \big(\textsc{Opt}[I] \cap \textsc{Opt}[\hat{I}]\big)\big|$$

  also fails Lipschitz-completeness, according to Example 1.

Some care must also be put into how the error is normalized. Below, we give two examples where the normalized error becomes too small.

- Normalizing the Hamming distance, we obtain the Jaccard distance:

$$\frac{|I \cup \hat{I}| - |I \cap \hat{I}|}{|I \cup \hat{I}|}$$

  This measure is sensitive to *dummy requests*: The adversary can construct a bad input and then add a lot of intervals to $I \cap \hat{I}$ that neither the algorithm nor OPT will choose, driving down the error, and, thus, failing the Lipschitz property.

- Normalizing by the total number of possible intervals (order m^2), the adversary can make the error of any input arbitrarily small by "scaling up" each edge to an arbitrarily long path, without changing an algorithm's profit, therefore failing the Lipschitz property.

Note that by normalizing by the size of an optimal solution instead of the size of a set of intervals, we avoid giving the adversary the power to drive down the error value.

3 Disjoint-Path Allocation

In this section, we show that a simple algorithm TRUST for the disjoint path allocation problem has an optimal competitive ratio for the graph class trees. TRUST simply relies on the prediction being correct. Specifically, it computes an optimal solution I^* in $\hat{I}$ before processing the first request. Then, it accepts any request in I^* that arrives and rejects all others.

We first establish that, *on any graph*, $\textsc{Trust}(\hat{I}, I) \geq \textsc{Opt}(I) - 2\eta(\hat{I}, I) = (1 - 2\gamma(\hat{I}, I))\,\textsc{Opt}(I)$. The proof follows by observing that

1. false negatives cause a deficit of at most $\textsc{Opt}(\mathrm{FN})$ in the schedule of TRUST compared to I^*,

2. false positives cause a deficit of at most $\textsc{Opt}(\mathrm{FP})$ relative to I^*, compared to the optimal schedule for I, and

3. $\textsc{Opt}(\mathrm{FP}) + \textsc{Opt}(\mathrm{FN}) \leq 2\,\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = 2\eta$.

Theorem 1 The algorithm TRUST is strictly $(1 - 2\gamma)$-competitive.

Proof Since I^* is an optimal selection from $\mathrm{TP} \cup \mathrm{FP}$, the largest additional set of requests that OPT would be able to accept from I compared to I^* would be an optimal selection from FN. Thus, $\textsc{Opt}(I) \leq \textsc{Opt}(I^*) + \textsc{Opt}(\mathrm{FN})$, and so $\textsc{Opt}(I^*) \geq \textsc{Opt}(I) - \textsc{Opt}(\mathrm{FN})$.

Similarly, the largest number of requests that can be detracted from TRUST is realized when requests that it planned to accept from I^* do not appear is $\textsc{Opt}(\mathrm{FP})$. Therefore, $\textsc{Trust}(\hat{I}, I) \geq \textsc{Opt}(I^*) - \textsc{Opt}(\mathrm{FP})$. Now,

$$\begin{aligned}
\textsc{Trust}(\hat{I}, I) &\geq \textsc{Opt}(I^*) - \textsc{Opt}(\mathrm{FP}) \\
&\geq \textsc{Opt}(I) - \textsc{Opt}(\mathrm{FN}) - \textsc{Opt}(\mathrm{FP}) \\
&\geq \textsc{Opt}(I) - 2\,\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) \\
&= \textsc{Opt}(I) - 2\eta(\hat{I}, I) \\
&= \big(1 - 2\gamma(\hat{I}, I)\big)\,\textsc{Opt}(I)
\end{aligned}$$

□

The following result shows that Theorem 1 is tight for the most interesting case of relatively small errors, and Corollary 1 shows that this is the case even for trees.

Theorem 2 For $0 \leq \gamma \leq \frac{1}{4}$, the competitive ratio of any deterministic algorithm for the online disjoint path allocation problem is at most $1 - 2\gamma$.

Proof Let ALG be any deterministic algorithm. We prove that, for any $0 \leq \gamma \leq \frac{1}{4}$, there exists a graph, a set of predicted requests $\hat{I}_w$, and a request sequence I_w such that $\textsc{Alg}(\hat{I}_w, I_w) \leq \big(1 - 2\gamma(\hat{I}_w, I_w)\big)\,\textsc{Opt}(I_w)$, $\gamma(\hat{I}_w, I_w)$ is arbitrarily close to γ, and $\textsc{Opt}(I_w)$ is arbitrarily large. More specifically, we prove the following:

For any $0 \leq \gamma \leq \frac{1}{4}$ and $0 < \varepsilon < \frac{1}{3}$, there exists a graph, a set of predicted requests $\hat{I}_w$, and a request sequence I_w such that

1. $\textsc{Alg}(\hat{I}_w, I_w) \leq \big(1 - 2\gamma(\hat{I}_w, I_w)\big)\,\textsc{Opt}(I_w)$,
2. $|\gamma(\hat{I}_w, I_w) - \gamma| \leq \varepsilon$, and
3. $\textsc{Opt}(I_w) \geq \frac{1}{\varepsilon}$.

Let $p = \left\lceil \frac{1}{3\varepsilon} \right\rceil$ and consider a set of p disjoint copies of the star S_8 consisting of a center vertex with 8 neighbor vertices. The prediction is fixed, but the input sequence depends on the algorithm's actions. Given that stars do

not share edges between them, the total error and the algorithm's profit are summed over all stars.

For star $0 \leq i \leq p-1$, the non-center vertices are numbered $8i+j$, where $1 \leq j \leq 8$, but we refer to these vertices by the value j. For each star, the prediction is

$$\hat{I}_w = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\} \ .$$

Note that $\textsc{Opt}(\hat{I}_w) = 3$.

For each star i, we let η_i, $\textsc{Alg}_i$, and $\textsc{Opt}_i$ denote the contribution of that star to $\eta(\hat{I}_w, I_w)$, $\textsc{Alg}(\hat{I}_w, I_w)$, and $\textsc{Opt}(I_w)$, respectively. The adversary chooses an integer ℓ between 0 and p and constructs the input sequence I_w such that the following hold:

- For the first ℓ stars,
  - $\eta_i = 1$,
  - $\textsc{Opt}_i \in \{3,4\}$, and
  - $\textsc{Alg}_i \leq \textsc{Opt}_i - 2\eta_i$.
- For the last $p-\ell$ stars, the input is equal to the prediction, so
  - $\eta_i = 0$
  - $\textsc{Opt}_i = 3$, and
  - $\textsc{Alg}_i \leq \textsc{Opt}_i - 2\eta_i$ (trivially, since $\eta_i = 0$).

This will result in

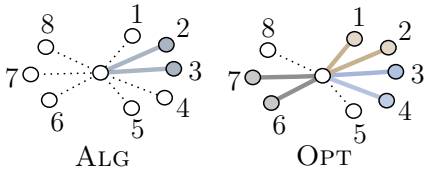
$$\begin{aligned}\textsc{Alg}(\hat{I}_w, I_w) &\leq \sum_{i=0}^{p-1} (\textsc{Opt}_i - 2\eta_i) \\ &= \textsc{Opt}(I_w) - 2\eta(\hat{I}_w, I_w) \\ &= \big(1 - 2\gamma(\hat{I}_w, I_w)\big)\, \textsc{Opt}(I_w)\end{aligned}$$

and

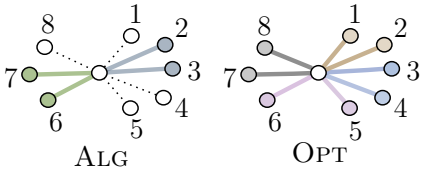
$$\textsc{Opt}(I_w) \geq 3p \geq \frac{1}{\varepsilon} \, ,$$

proving Items 1 and 3.

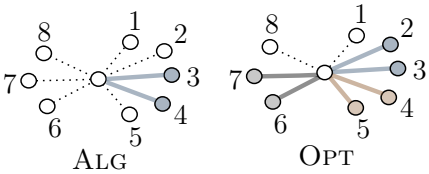
We now explain how the adversary constructs the input sequence I_w. For $0 \leq i \leq \ell - 1$, I_w starts with $\langle \mathbf{(2,3),(3,4),(6,7),(7,8)} \rangle$. Note that the



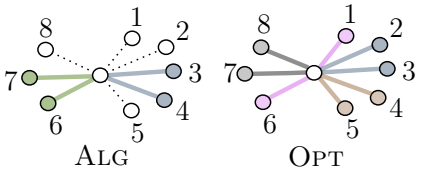
(a) $I_w = \langle (2,3), (3,4), (6,7), (7,8), \mathbf{(1,2)} \rangle$
ALG accepts $(2,3)$, rejects $(6,7)$ and $(7,8)$.



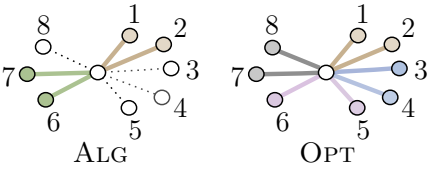
(b) $I_w = \langle (2,3), (3,4), (6,7), (7,8), \mathbf{(1,2)}, \mathbf{(5,}\boldsymbol{x}\mathbf{)} \rangle$
ALG accepts $(2,3), (7,x)$; here $x = 6$, $y = 8$.



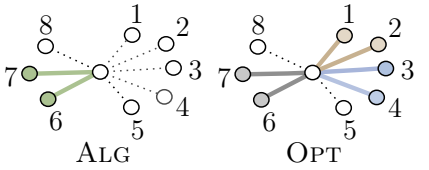
(c) $I_w = \langle (2,3), (3,4), (6,7), (7,8), \mathbf{(4,5)} \rangle$
ALG rejects $(2,3)$, accepts $(3,4)$, and rejects both $(6,7)$ and $(7,8)$.



(d) $I_w = \langle (2,3), (3,4), (6,7), (7,8), \mathbf{(4,5)}, \mathbf{(1,}\boldsymbol{x}\mathbf{)} \rangle$
ALG rejects $(2,3)$, accepts $(3,4)$ and $(7,x)$; here $x = 6$ and $y = 8$.



(e) $I_w = \langle (2,3), (3,4), (6,7), (7,8), \mathbf{(1,2)}, \mathbf{(5,}\boldsymbol{x}\mathbf{)} \rangle$
ALG rejects $(2,3)$, $(3,4)$, accepts $(7,x)$ and $(1,2)$; here $x = 6$ and $y = 8$.



(f) $I_w = \langle (2,3), (3,4), (6,7), (7,8), \mathbf{(1,2)} \rangle$
ALG rejects $(2,3)$, $(3,4)$, accepts $(7,x)$, and rejects $(1,2)$; here $x = 6$ and $y = 8$.

Figure 2: Illustration of the proof of Theorem 2. Highlighted edges indicate paths between accepted pairs.

path between 2 and 3 shares an edge with the path between 3 and 4, so ALG can accept at most one of the requests $(2,3)$ and $(3,4)$. The same is true for the requests $(6,7)$ and $(7,8)$. The rest of the input depends on the actions of ALG, as outlined in the cases below.

Case Alg accepts $(2,3)$: In this case, the next request to arrive is $\mathbf{(1,2)}$. ALG cannot accept any of the requests $(1,2)$ and $(3,4)$. The predicted request $(4,5)$ does not arrive, i.e., $(4,5)$ is a false positive. Note that there are no other false positives.

Subcase Alg rejects both $(6,7)$ and $(7,8)$: In this case, no further requests arrive (see Figure 2a), so there are no false negatives. Thus, $\textsc{Opt}(\mathrm{FN} \cup \mathrm{FP}) = 1$. Moreover, OPT accepts three requests, $(1,2)$ and $(3,4)$ combined with $(6,7)$ or $(7,8)$, while ALG accepts only $(2,3)$.

Subcase Alg accepts $(6,7)$ or $(7,8)$: We let $\{x,y\} = \{6,8\}$ such that ALG accepts $(7,x)$ and rejects $(7,y)$. Now, a false negative, $\mathbf{(5,\boldsymbol{x})}$, arrives (see Figure 2b). Since $(5,x)$ shares an edge with the false positive $(4,5)$, $\textsc{Opt}(\mathrm{FN} \cup \mathrm{FP}) = 1$. ALG accepts $\{(2,3),(7,x)\}$, and OPT accepts $\{(1,2),(3,4),(5,x),(7,y)\}$.

In both subcases, $\eta_i = 1$, $\textsc{Opt}_i \in \{3,4\}$, and $\textsc{Alg}_i = \textsc{Opt}_i - 2 = \textsc{Opt}_i - 2\eta_i$.

Case Alg rejects $(2,3)$:

Subcase Alg accepts $(3,4)$: The next request to arrive is $\mathbf{(4,5)}$, which the algorithm cannot accept. The request $(1,2)$ does not arrive, so it is a false positive.

Subsubcase Alg rejects both $(6,7)$ and $(7,8)$: In this case, no further requests arrive (see Figure 2c). Thus, ALG accepts only $(3,4)$, OPT accepts $(2,3)$ and $(4,5)$ together with $(6,7)$ or $(7,8)$, and $\textsc{Opt}(\mathrm{FN} \cup \mathrm{FP}) = 1$.

Subsubcase Alg accepts $(6,7)$ or $(7,8)$: As above, we define $\{x,y\} = \{6,8\}$ such that ALG accepts $(7,x)$ and rejects $(7,y)$. Now, a false negative, $\mathbf{(1,\boldsymbol{x})}$, arrives (see Figure 2d). Since $(1,2)$ and $(1,x)$ share an edge, $\textsc{Opt}(\mathrm{FN} \cup \mathrm{FP}) = 1$. Moreover, ALG accepts $\{(3,4),(7,x)\}$, and OPT accepts the set $\{(1,x),(2,3),(4,5),(7,y)\}$.

In both subsubcases, $\eta_i = 1$, $\textsc{Opt}_i \in \{3,4\}$, and $\textsc{Alg}_i = \textsc{Opt}_i - 2 = \textsc{Opt}_i - 2\eta_i$.

Subcase Alg rejects $(3,4)$: The next request to arrive is $\mathbf{(1,2)}$. The request $(4,5)$ is a false positive.

Subsubcase Alg accepts (6, 7) or (7, 8): As above, we define $\{x, y\} = \{6, 8\}$ such that ALG accepts $(7, x)$ and rejects $(7, y)$.

Subsubsubcase Alg accepts (1, 2): The final request is a false negative, $\boldsymbol{(5, x)}$ (see Figure 2e). Since $(4, 5)$ and $(5, x)$ share an edge, $\text{OPT}(\text{FN} \cup \text{FP}) = 1$. ALG accepts $\{(1, 2), (7, x)\}$ and OPT accepts $\{(1, 2), (3, 4), (5, x), (7, y)\}$.

Subsubsubcase Alg rejects (1, 2): In this case, no further requests arrive (see Figure 2f). Thus, ALG accepts $\{(7, x)\}$, OPT accepts $\{(3, 4), (5, x), (7, y)\}$, and $\text{OPT}(\text{FN} \cup \text{FP}) = 1$.

In both subsubsubcases, $\eta_i = 1$, $\text{OPT}_i \in \{3, 4\}$, and $\text{ALG}_i = \text{OPT}_i - 2 = \text{OPT}_i - 2\eta_i$.

Subsubcase Alg rejects both (6, 7) and (7, 8): In this case, no further requests arrive. Thus, the profit of ALG is 1 if it accepts $(1, 2)$ and 0 otherwise, while OPT accepts $(1, 2)$ and $(3, 4)$ together with $(6, 7)$ or $(7, 8)$, and $\text{OPT}(\text{FN} \cup \text{FP}) = 1$. Thus, $\eta_i = 1$, $\text{OPT}_i = 3$, and $\text{ALG}_i \leq \text{OPT}_i - 2 = \text{OPT}_i - 2\eta_i$.

This completes the proof of Items 1 and 3. For Item 2, note that

- $\gamma(\hat{I}_w, I_w) = 0$, for $\ell = 0$, and
- $\frac{1}{4} \leq \gamma(\hat{I}_w, I_w) \leq \frac{1}{3}$, for $\ell = p$.

Also note that incrementing ℓ by one increases $\eta(\hat{I}_w, I_w)$ by 1 and does not decrease $\text{OPT}(I_w)$. Thus, since $\text{OPT}(I_w) \geq 1/\varepsilon$, incrementing ℓ adds at most

$$\Delta_\gamma \leq \frac{1}{1/\varepsilon} = \varepsilon$$

to $\gamma(\hat{I}_w, I_w)$. This proves that for any $0 \leq \gamma \leq \frac{1}{4}$, the adversary can choose ℓ such that

$$|\gamma(\hat{I}_w, I_w) - \gamma| \leq \varepsilon\,.$$

□

Corollary 1 Any deterministic algorithm for the online disjoint path problem on star graphs with arbitrarily high degree or trees with arbitrarily many vertices of degree at least 8 has competitive ratio at most $1 - 2\gamma$, $1 \leq \gamma \leq \frac{1}{4}$.

Proof The stars used in the proof of Theorem 2 can be connected in several ways to form a connected graph. For instance, the proof still holds, if the vertex $8i+1$ is connected by an edge to the vertex $8(i+1)+2$, $0 \leq i \leq p-2$, or if all center vertices are identified resulting in a star of one center vertex with $8p$ neighbors. This is true, since the paths are only required to be edge disjoint, not vertex disjoint. □

4 Interval Scheduling

In this section, we first show tight upper and lower bounds on the competitive ratio of deterministic algorithms for interval scheduling. We then study trade-offs between consistency and robustness for deterministic and randomized algorithms and present some experimental results. Recall that m denotes the number of edges on the given path.

4.1 A General Upper Bound for Deterministic Algorithms

As an introduction to the difficulties in designing algorithms for the problem, we start by proving a general lower bound. We show that for $0 < \gamma < 1$, no deterministic algorithm can have a competitive ratio better than $1 - \gamma$:

Theorem 3 For any $0 \leq \gamma \leq 1$, the competitive ratio of any deterministic algorithm for the online interval scheduling problem is at most $1 - \gamma$.

Proof Let ALG be any deterministic algorithm. We prove that, for any $0 \leq \gamma \leq 1$, there exists a path graph, a set of predicted requests $\hat{I}_w$, and a request sequence I_w such that $\textsc{Alg}(\hat{I}_w, I_w) \leq \big(1 - \gamma(\hat{I}_w, I_w)\big)\,\textsc{Opt}(I_w)$, $\gamma(\hat{I}_w, I_w)$ is arbitrarily close to γ, and $\textsc{Opt}(I_w)$ is arbitrarily large. More specifically, we prove the following:

For any $0 \leq \gamma \leq 1$ and $0 < \varepsilon < 1$, there is an input sequence I_w and a set of predictions $\hat{I}_w$ such that

1. $\textsc{Alg}(\hat{I}_w, I_w) \leq \big(1 - \gamma(\hat{I}_w, I_w)\big)\,\textsc{Opt}(I_w)\,,$
2. $|\gamma(\hat{I}_w, I_w) - \gamma| \leq \varepsilon\,,$ and
3. $\textsc{Opt}(I_w) \geq \dfrac{1}{\varepsilon^2}\,.$

For any $0 < \varepsilon < 1$, let $c = \left\lceil \frac{1}{\varepsilon} \right\rceil$ and $p = \left\lceil \frac{1}{\varepsilon^2} \right\rceil$. The prediction consists of $2p$ requests:

$$\hat{I}_w = \bigcup_{i=0}^{p-1} \Big\{ \big(ci, c(i+1)\big), \big(ci, ci+1\big) \Big\} .$$

The input I_w is formed by p phases, $0 \le i \le p-1$. For each phase i, we let η_i, $\textsc{Alg}_i$, and $\textsc{Opt}_i$ denote the contribution of that phase to $\eta(\hat{I}_w, I_w)$, $\textsc{Alg}(\hat{I}_w, I_w)$, and $\textsc{Opt}(I_w)$, respectively. The adversary chooses an integer ℓ between 0 and $p-1$ and does the following.

- For $0 \le i \le \ell - 1$, the ith phase starts with the true positive $(ci, c(i+1))$, and the remainder of the phase depends on whether the algorithm accepts this request or not.

  Case Alg accepts $(ci, c(i+1))$: In this case, the phase continues with

  $$\{(ci + j, ci + (j+1)) \mid 0 \le j \le c-1\} .$$

  The first of these requests is a true positive, and the other $c-1$ are false negatives.

  Note that $\textsc{Alg}$ cannot accept any of these c requests. The optimal algorithm rejects the original request $(ci, c(i+1))$ and accepts all of the c following unit-length requests. We conclude that

  - $\eta_i = c - 1$,
  - $\textsc{Opt}_i = c$, and
  - $\textsc{Alg}_i = 1 = \textsc{Opt}_i - \eta_i$.

  Case Alg rejects $(ci, c(i+1))$: In this case, the phase ends with no further requests. Thus, $(ci, ci+1)$ is a false positive, and we obtain

  - $\eta_i = 1$,
  - $\textsc{Opt}_i = 1$, and
  - $\textsc{Alg}_i = 0 = \textsc{Opt}_i - \eta_i$.

- For $\ell \le i \le p-1$, the ith phase consists of the true positives $(ci, c(i+1))$ and $(ci, ci+1)$. Thus,

  - $\eta_i = 0$,
  - $\textsc{Opt}_i = 1$, and

– $\textsc{Alg}_i \leq 1 = \textsc{Opt}_i - \eta_i$

Since the intervals in $\mathrm{FP} \cup \mathrm{FN}$ are disjoint, we can write

$$\eta(\hat{I}_w, I_w) = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = \sum_{i=0}^{\ell-1} \eta_i \,.$$

Thus,

$$\begin{aligned}
\textsc{Alg}(\hat{I}_w, I_w) &= \sum_{i=0}^{p-1} \textsc{Alg}_i \\
&\leq \sum_{i=0}^{p-1} (\textsc{Opt}_i - \eta_i) \\
&= \textsc{Opt}(I_w) - \eta(\hat{I}_w, I_w) \\
&= \big(1 - \gamma(\hat{I}_w, I_w)\big)\, \textsc{Opt}(I_w) \,.
\end{aligned}$$

This proves Item 1.

Since $\textsc{Opt}$ accepts at least one interval in each phase,

$$\textsc{Opt}(I_w) \geq p \,,$$

which proves Item 3.

Finally, for Item 2, note that

- $\gamma(\hat{I}_w, I_w) = 0$, for $\ell = 0$, and
- $\dfrac{c-1}{c} \leq \gamma(\hat{I}_w, I_w) \leq 1$, for $\ell = p$,

where

$$1 - \frac{c-1}{c} = \frac{1}{c} = \frac{1}{\left\lceil \frac{1}{\varepsilon} \right\rceil} \leq \varepsilon \,.$$

Also note that incrementing ℓ by one, increases $\eta(\hat{I}_w, I_w)$ by 1 or $c-1$ and does not decrease $\textsc{Opt}(I_w)$. Thus, incrementing ℓ adds at most

$$\Delta_\gamma = \frac{c-1}{p} < \frac{\frac{1}{\varepsilon}}{\left\lceil \frac{1}{\varepsilon^2} \right\rceil} \leq \varepsilon$$

to $\gamma(\hat{I}_w, I_w)$. This proves that for any $0 \leq \gamma \leq 1$, the adversary can choose ℓ such that

$$|\gamma(\hat{I}_w, I_w) - \gamma| < \varepsilon \,.$$

□

4.2 Trust

The next theorem gives an upper bound on the competitive ratio of TRUST which is lower than the general upper bound of Theorem 3. The proof uses an adversarial sequence similar to that of Theorem 3.

Theorem 4 For the online interval scheduling problem, the competitive ratio of TRUST is at most $1 - 2\gamma$.

Proof We prove that, for any $0 \leq \gamma \leq \frac{1}{2}$, there exists a path graph, a set of predicted requests $\hat{I}_w$, and a request sequence I_w such that $\textsc{Trust}(\hat{I}_w, I_w) \leq \big(1 - 2\gamma(\hat{I}_w, I_w)\big)\,\textsc{Opt}(I_w)$, $\gamma(\hat{I}_w, I_w)$ is arbitrarily close to γ, and $\textsc{Opt}(I_w)$ is arbitrarily large. More specifically, we prove the following:

For any $0 \leq \gamma \leq \frac{1}{2}$ and $0 < \varepsilon \leq 1$, there exists a path graph, a set of predicted requests $\hat{I}_w$, and a request sequence I_w such that

1. $\textsc{Trust}(\hat{I}_w, I_w) \leq \big(1 - 2\gamma(\hat{I}_w, I_w)\big)\,\textsc{Opt}(I_w)$,
2. $|\gamma(\hat{I}_w, I_w) - \gamma| \leq \varepsilon$, and
3. $\textsc{Opt}(I_w) \geq \frac{1}{\varepsilon}$.

For any $0 < \varepsilon \leq 1$, let $p = \left\lceil \frac{1}{\varepsilon} \right\rceil$. Let the prediction be a set of length-2 intervals:

$$\hat{I}_w = \bigcup_{i=0}^{p-1} \big\{(3i, 3i+2), (3i+1, 3i+3)\big\}.$$

TRUST chooses an optimal solution I^* from $\hat{I}_w$. For each i, I^* will contain either $(3i, 3i+2)$ or $(3i+1, 3i+3)$. The adversary chooses an integer ℓ between 0 and p.

- For $0 \leq i \leq \ell - 1$, the adversary does the following.

  If $(3i, 3i+2)$ is in I^*, that interval will be in FP, and OPT will select $(3i+1, 3i+3)$, which will be a TP-interval. Further, I_w will contain the FN-interval, $(3i, 3i+1)$.

  If, instead, $(3i+1, 3i+3)$ is in I^*, that interval will be in FP, and OPT will select $(3i, 3i+2)$, which will be a TP-interval. Further, I_w will then contain the FN-interval, $(3i+2, 3i+3)$.

- For $\ell \le i \le p$, the adversary gives the two predicted intervals, $(3i, 3i+2)$ and $(3i+1, 3i+3)$.

Thus,

$$\textsc{Opt}(I_w) = 2\ell + (p - \ell) = p + \ell \ge \frac{1}{\varepsilon}\,,$$

proving Item 3.

For each $i < \ell$, the interval in FP and the interval in FN overlap, so

$$\eta(\hat{I}_w, I_w) = \textsc{Opt}(\mathrm{FN} \cup \mathrm{FP}) = \ell\,.$$

Since the first ℓ intervals in I^* are false positives,

$$\begin{aligned}
\textsc{Trust}(\hat{I}_w, I_w) &= p - \ell \\
&= \textsc{Opt}(I_w) - 2\ell \\
&= \textsc{Opt}(I_w) - 2\eta(\hat{I}_w, I_w) \\
&= \big(1 - 2\gamma(\hat{I}_w, I_w)\big)\,\textsc{Opt}(I_w)\,,
\end{aligned}$$

proving Item 1.

For Item 2, note that $\gamma(\hat{I}_w, I_w) = 0$, for $\ell = 0$, and $\gamma(\hat{I}_w, I_w) = p/2p = 1/2$, for $\ell = p$. Also note that incrementing ℓ by one increases $\eta(\hat{I}_w, I_w)$ by 1 and does not decrease $\textsc{Opt}(I_w)$. Thus, since $\textsc{Opt}(I_w) \ge 1/\varepsilon$, incrementing ℓ adds at most

$$\Delta_\gamma \le \frac{1}{1/\varepsilon} = \varepsilon$$

to $\gamma(\hat{I}_w, I_w)$. This proves that for any $0 \le \gamma \le \frac{1}{2}$, the adversary can choose ℓ such that

$$|\gamma(\hat{I}_w, I_w) - \gamma| \le \varepsilon\,.$$

□

In light of Theorems 1 and 4, the competitive ratio of Trust for interval scheduling is $1-2\gamma$. Meanwhile, Theorem 3 gives an upper bound of $1-\gamma$ on the competitiveness of deterministic algorithms. This gap suggests potential for improvement, which we explore in the next section.

4.3 TrustGreedy

In this section, we introduce an algorithm TrustGreedy, which achieves an optimal competitive ratio for interval scheduling.

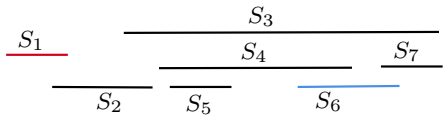


Figure 3: An illustration of TrustGreedy: Suppose $I = \langle S_1, S_2, S_3, S_4, S_5, S_7 \rangle$ and $\hat{I} = (I \setminus S_1) \cup S_6$, so that S_1 is a false negative and S_6 is a false positive. The optimal solution for $\hat{I}$ constructed by TrustGreedy is $I^* = \{S_2, S_5, S_6\}$. Initially, the set A is initialized as I^*. Upon the arrival of S_1, it replaces S_2, and A becomes $\{S_1, S_5, S_6\}$. Eventually, the algorithm accepts $\{S_1, S_5\}$.

4.3.1 The algorithm.

TrustGreedy starts by choosing an optimal solution, I^*, from the predictions in $\hat{I}$. This optimal offline solution is selected by repeatedly including an interval that ends earliest possible among those in $\hat{I}$ that do not overlap any already selected intervals.

During the online processing after this initialization, TrustGreedy maintains an updated plan, A. Initially, A is I^*. When a request, r, is contained in A, it is accepted. When a request, r, is in FN, TrustGreedy accepts if r overlaps no previously accepted intervals and can be accepted by replacing at most one other interval in A that ends no earlier than r. In that case, r is added to A, possibly replacing an overlapping interval to maintain the feasibility of A (no two intervals overlap).

As a comment, only the first interval from FN that replaces an interval r in the current A is said to "replace" it. There may be other intervals from FN that overlap r and are accepted by TrustGreedy, but they are not said to "replace" it. Figure 3 provides an illustration.

4.3.2 Analysis.

Let TG denote the set of intervals chosen by TrustGreedy on input I and prediction $\hat{I}$, and Opt the intervals chosen by some optimal offline algorithm. We define the following subsets of TG and Opt:

- $\mathrm{TG}^{\mathrm{TP}} = \mathrm{TG} \cap \hat{I} = \mathrm{TG} \cap \mathrm{TP}$
  $\mathrm{Opt}^{\mathrm{TP}} = \mathrm{Opt} \cap \hat{I} = \mathrm{Opt} \cap \mathrm{TP}$

- $\mathrm{TG}^{\mathrm{FN}} = \mathrm{TG} \cap \mathrm{FN}$
  $\textsc{Opt}^{\mathrm{FN}} = \textsc{Opt} \cap \mathrm{FN}$

Note that since $I = \mathrm{TP} \cup \mathrm{FN}$,

$$\mathrm{TG} = \mathrm{TG}^{\mathrm{TP}} \cup \mathrm{TG}^{\mathrm{FN}} \text{ and } \textsc{Opt} = \textsc{Opt}^{\mathrm{TP}} \cup \textsc{Opt}^{\mathrm{FN}} .$$

In the example of Figure 3, we have $\textsc{Opt} = \{S_1, S_5, S_7\}$, $\mathrm{TG} = \{S_1, S_5\}$, $\mathrm{TG}^{\mathrm{TP}} = \{S_5\}$, $\textsc{Opt}^{\mathrm{TP}} = \{S_5, S_7\}$, $\mathrm{TG}^{\mathrm{FN}} = \{S_1\}$ and $\textsc{Opt}^{\mathrm{FN}} = \{S_1\}$.

Lemma 1 Each interval $i \in \textsc{Opt}^{\mathrm{TP}}$ overlaps an interval in I^* extending no further to the right than i.

Proof Assume to the contrary that there is no interval in I^* that overlaps i and ends no later than i. If i does not overlap anything in I^*, we could have added i to I^* and have a feasible solution (non-overlapping intervals), contradicting the fact that I^* is optimal. Thus, i must overlap an interval $r \in I^*$, which, by assumption, must end strictly later than i.

If r is the only interval in I^* overlapping i, this contradicts the construction of I^*, since i would have been in I^* instead of r.

Otherwise, there is an interval $s \in I^* \setminus \{r\}$ overlapping i. Since r and s are both in I^* and r overlaps the right endpoint of i, s ends no later than the start of r, meaning that s overlaps i and ends before i, contradicting the assumption that no interval in I^* overlaps i and ends no later than i. □

We let U denote the set of intervals in $I^* \cap \mathrm{FP}$ that are not replaced during the execution of TrustGreedy, i.e., those intervals that stay in the plan A but never show up in I. We define a set O^{FN} consisting of a copy of each interval in $\textsc{Opt}^{\mathrm{FN}}$ and let $\mathcal{F} = O^{\mathrm{FN}} \cup U$. We define a mapping

$$f\colon \textsc{Opt} \to \mathrm{TG} \cup \mathcal{F}$$

as follows. For each $i \in \textsc{Opt}$:

1. If there is an interval in I^* that overlaps i and ends no later than i, then let r be the rightmost such interval.
   (a) If $r \in U \cup \mathrm{TG}^{\mathrm{TP}}$, then $f(i) = r$.
   (b) Otherwise, r has been replaced by some interval $t \in \mathrm{FN}$. In this case, $f(i) = t$.

2. Otherwise, by Lemma 1, i belongs to OPT^{FN}.

   (a) If there is

   - an interval in TG^{FN} that overlaps i and ends no later than i and
   - an interval in U that overlaps i's right endpoint,

   let r be the rightmost interval in TG^{FN} that overlaps i and ends no later than i. In this case, $f(i) = r$.

   (b) Otherwise, let o_i be the copy of i in O^{FN}. In this case, $f(i) = o_i$.

In the example of Figure 3, we have $U = \{S_6\}$ and $O^{\text{FN}} = \{S_1'\}$, where S_1' is a copy of S_1. Thus, we have $\mathcal{F} = \{S_1', S_6\}$ and the mapping f is from $\{S_1, S_5, S_7\}$ to $\{S_1', S_1, S_5, S_6\}$. The mapping would be $S_1 \to S_1'$, $S_5 \to S_5$, $S_7 \to S_6$.

We let F denote the subset of $\mathcal{F}$ mapped to by f and note that in step 1a, intervals are added to $F \cap U$ when $r \in U$. In step 2b, all intervals are added to $F \cap O^{\text{FN}}$.

We prove that f is an injection (Lemma 2) and F is a feasible solution (Lemma 3), and conclude that TrustGreedy has an optimal competitive ratio (Theorem 5).

Lemma 2 The mapping f is an injection.

Proof Intervals in $U \cup \text{TG}^{\text{TP}}$ are only mapped to in step 1a. If an interval $i \in \text{OPT}$ is mapped to an interval $r \in U \cup \text{TG}^{\text{TP}}$, i overlaps the right endpoint of r. There can be only one interval in OPT overlapping the right endpoint of r, so this part of the mapping is injective.

Intervals in TG^{FN} are only mapped to in steps 1b and 2a. In step 1b, only intervals that replace intervals in I^* are mapped to. Since each interval in TG^{FN} replaces at most one interval in I^* and the right endpoint of each interval in I^* overlaps at most one interval in OPT, no interval is mapped to twice in step 1b. If, in step 2a, an interval, i, is mapped to an interval, r, i overlaps the right endpoint of r. There can be only one interval in OPT overlapping the right endpoint of r, so no interval is mapped to twice in step 2a.

We now argue that no interval is mapped to in both steps 1b and 2a. Assume that an interval, i_1, is mapped to an interval, t, in step 1b. Then, there is an interval, r, such that r overlaps the right endpoint of t and i_1

overlaps the right endpoint of r. This means that the right endpoint of i_1 is no further to the left than the right endpoint of t. Assume for the sake of contradiction that an interval $i_2 \neq i_1$ is mapped to t in step 2a. Then, i_2 overlaps the right endpoint of t, and there is an interval, $u \in U$, overlapping the right endpoint of i_2. Since i_2 overlaps t and t does not extend to the right of i_1, i_2 must be to the left of i_1. Since i_2 is mapped to t, t extends no further to the right than i_2. Thus, since r overlaps both t and i_1, r must overlap the right endpoint of i_2, and hence, r overlaps u. This is a contradiction since r and u are both in I^*.

Intervals in $F \cap O^{\mathrm{FN}}$ are only mapped to in step 2b and no two intervals are mapped to the same interval in this step. □

Lemma 3 The subset F of $\mathcal{F}$ mapped to by f is a feasible solution.

Proof We first note that $F \cap U$ is feasible since $F \cap U \subseteq U \subseteq I^*$ and I^* is feasible. Moreover, $F \cap O^{\mathrm{FN}}$ is feasible since the intervals of $F \cap O^{\mathrm{FN}}$ are identical to the corresponding subsets of Opt. Thus, we only need to show that no interval in $F \cap U$ overlaps any interval in $F \cap O^{\mathrm{FN}}$.

Consider an interval $u \in F \cap U$ mapped to from an interval $i \in \textsc{Opt}$. Since i is not mapped to its own copy in $\mathcal{F}$, its copy does not belong to F. Since $i \in \textsc{Opt}$, no interval in $F \cap O^{\mathrm{FN}}$ overlaps i. Thus, it is sufficient to argue that $F \cap O^{\mathrm{FN}}$ contains no interval strictly to the left of i overlapping u.

Assume for the sake of contradiction that there is an interval $\ell \in F \cap O^{\mathrm{FN}}$ to the left of i overlapping u. Since ℓ ended up in F although its right endpoint is overlapped by an interval from U, there is no interval in I^* (because of step 1 in the mapping algorithm) or in $\mathrm{TG}^{\mathrm{FN}}$ (because of step 2a in the mapping algorithm) overlapping ℓ and ending no later than ℓ. Thus, $I^* \cup \mathrm{TG}^{\mathrm{FN}}$ contains no interval strictly to the left of u overlapping ℓ. This contradicts the fact that u has not been replaced since the interval in $\textsc{Opt}^{\mathrm{FN}}$ corresponding to ℓ could have replaced it. □

Using Lemmas 2 and 3, we obtain a lower bound matching the general upper bound of Theorem 3:

Theorem 5 The competitive ratio of TrustGreedy for the online interval scheduling problem is at least $1 - \gamma$.

Proof We show that

$$\begin{aligned}\textsc{TrustGreedy}(\hat{I}, I) &\geq \textsc{Opt}(I) - \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) \\ &= \big(1 - \gamma(\hat{I}, I)\big)\, \textsc{Opt}(I) :\end{aligned}$$

$$\begin{aligned}\textsc{Opt}(I) &\leq |\,\mathrm{TG}\,| + |F|, \text{ since, by Lemma 2, } f \text{ is an injection} \\ &\leq |\,\mathrm{TG}\,| + \textsc{Opt}(\mathcal{F}), \text{ since, by Lemma 3, } F \text{ is feasible} \\ &\leq |\,\mathrm{TG}\,| + \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}), \text{ since } U \subseteq \mathrm{FP} \text{ and } \textsc{Opt}^{\mathrm{FN}} \subseteq \mathrm{FN}\end{aligned}$$

□

4.4 Consistency-Robustness Trade-off

We study the trade-off between the competitive ratio of the interval scheduling algorithm when predictions are error-free (consistency) and when predictions are adversarial (robustness).

4.4.1 General Upper Bounds

The following proposition shows an obvious trade-off between the consistency and robustness of deterministic algorithms.

Proposition 2 For the online interval scheduling problem on a path of m vertices, any deterministic algorithm with consistency $\alpha \in \Theta(1)$ has robustness $\beta \in O(\frac{1}{m})$.

Proof If a deterministic algorithm Alg has a consistency of $\alpha \in \Theta(1)$, there exists a non-negative constant b such that, for any input I with a correct prediction $\hat{I} = I$, $\textsc{Alg}(\hat{I}, I) \geq \alpha\, \textsc{Opt}(I) - b$. Let $p = \lceil (b+1)/\alpha \rceil$ and $\ell = \lfloor m/p \rfloor$ and consider a prediction consisting of p disjoint intervals of length ℓ:

$$\hat{I} = \bigcup_{i=0}^{p-1} \{(i\ell, (i+1)\ell)\}\,.$$

Note that $\textsc{Opt}(\hat{I}) = p$. Thus, if the input starts with the p predicted intervals, Alg must accept at least one of them, since $\alpha p - b \geq 1$. Assume now that, for each of the predicted intervals accepted by Alg, the input continues with ℓ disjoint unit length intervals overlapping that interval. Since Opt

will accept each of the unit length intervals, this shows that the robustness of ALG is

$$\beta \leq \frac{1}{\ell} \leq \frac{p}{m-p+1}\,.$$

Since $p = \lceil (b+1)/\alpha \rceil$ is a constant, $\beta \in O\left(\frac{1}{m}\right)$. □

The more interesting case is randomized algorithms. The proof of the following was inspired by the proof of Theorem 13.8 in [19] for the online case without predictions, and that $\Omega(\log m)$ result was originally proven in [8]. We address the more relevant case of trade-offs when the robustness is non-trivial.

Theorem 6 Consider a (possibly randomized) α-consistent and β-robust algorithm ALG for the online interval scheduling problem. If there exists an ε, $0 < \varepsilon < 1$, such that $\beta \geq 1/m^{1-\varepsilon}$, then

$$\alpha \leq 1 - \beta \cdot \left(\frac{\varepsilon \cdot \log m}{2} - O(1) \right) \quad \text{and}$$
$$\beta \leq (1-\alpha) \cdot \frac{2}{\varepsilon \cdot \log m} + O\left(\frac{1}{\log^2 m} \right)\,.$$

Proof Let $r = \lfloor \log m \rfloor - 1$ and let $m' = 2^{r+1}$. Consider an input sequence

$$\sigma = \langle I_0, I_1, \ldots, I_{r+1} \rangle, \text{ where}$$
$$I_i = \langle (0, m'/2^i), (m'/2^i, 2m'/2^i), \ldots, (m' - m'/2^i, m') \rangle, \text{ for } 0 \leq i \leq r+1\,.$$

Note that I_i consists of 2^i disjoint intervals of length $m'/2^i$. Let

$$\sigma_i = \langle I_0, I_1, \ldots, I_i \rangle, \text{ for } 0 \leq i \leq r.$$

In order to maximize the number of small intervals that can be accepted if they arrive, an algorithm would minimize the (expected) fraction of the line occupied by the larger intervals, to leave space for the small intervals, while maintaining β-robustness.

For ALG to be β-robust, there must exist a constant b_β, independent of m, such that, for each i and any prediction $\hat{\sigma}_i$ of σ_i,

$$E[\text{ALG}(\hat{\sigma}_i, \sigma_i)] \geq \beta \cdot \text{OPT}(\sigma_i) - b_\beta = \beta \cdot 2^i - b_\beta\,.$$

Let $j = \max\{0, \lceil \log(b_\beta/\beta) \rceil\}$ and note that, for $i \geq j$, $\beta \cdot 2^i - b_\beta$ is non-negative.

Let n_i be the number of intervals from I_i accepted by ALG and let L_r be the total length of intervals from σ_r accepted by ALG. Since $\text{OPT}(\sigma_i) - \text{OPT}(\sigma_{i-1}) = 2^{i-1}$, we conclude, by linearity of expectations, that $E[L_r]$ is minimized, if

$$E[n_i] = \begin{cases} 0, & 0 \leq i < j \\ \beta \cdot \text{OPT}(\sigma_j) - b_\beta, & i = j \\ \beta \cdot 2^{i-1}, & j+1 \leq i \leq r\,. \end{cases}$$

With these values of $E[n_i]$, the expected total length of intervals from I_i, $j+1 \leq i \leq r$, accepted by ALG is $\beta \cdot 2^{i-1} \times m'/2^i = \beta \cdot m'/2$. Thus, by the linearity of expectations,

$$E[L_r] \geq \sum_{i=j+1}^{r} \beta \cdot \frac{m'}{2} = \beta \cdot \frac{m'(r-j)}{2}\,.$$

If the expected number of intervals that ALG accepts from σ_r is more than $\beta \cdot 2^r - b_\beta$, then $E[L_r]$ increases by more than one for each additional interval. Thus, by linearity of expectations, for any prediction $\hat{\sigma}$,

$$E[\text{ALG}(\hat{\sigma}, \sigma)] \leq \beta \cdot 2^r - b_\beta + \left(m' - \beta \cdot \frac{m'(r-j)}{2} \right)\,. \tag{1}$$

Now, let $\hat{\sigma}$ be the prediction consisting of exactly the intervals in σ. Then, for ALG to be α-consistent, there must be a constant b_α such that

$$E[\text{ALG}(\hat{\sigma}, \sigma)] \geq \alpha \cdot m' - b_\alpha\,. \tag{2}$$

Combining Inequalities (1) and (2), we obtain

$$\beta \cdot \frac{2^r}{m'} + 1 - \beta \cdot \frac{r-j}{2} + \frac{b_\alpha}{m'} \geq \alpha\,.$$

Since $2^r/m' = 1/2$ and $m' \geq m/2$, this reduces to

$$\begin{aligned}
\alpha &\leq 1 - \beta \cdot \frac{r - j - 1}{2} + \frac{2b_\alpha}{m} \\
&= 1 - \beta \cdot \frac{\lfloor \log m \rfloor - 1 - \lceil \log(b_\beta/\beta) \rceil - 1}{2} + \frac{2b_\alpha}{m} \\
&= 1 - \beta \cdot \frac{\log m - \log(1/\beta) - \log b_\beta - 4}{2} + \frac{2b_\alpha}{m} \\
&\leq 1 - \beta \cdot \left(\frac{\log m - \log(m^{1-\varepsilon}) - \log b_\beta - 4}{2} \right) + \frac{2b_\alpha}{m} \\
&= 1 - \beta \cdot \left(\frac{\varepsilon \cdot \log m - c}{2} \right) + \frac{2b_\alpha}{m}, \text{ where } c = \log b_\beta + 4 \\
&\leq 1 - \beta \cdot \left(\frac{\varepsilon \cdot \log m - c}{2} \right) + \beta \cdot 2b_\alpha, \text{ since } \beta \geq \frac{1}{m} \\
&= 1 - \beta \cdot \left(\frac{\varepsilon \cdot \log m}{2} - O(1) \right)
\end{aligned}$$

Solving for β, we get

$$\begin{aligned}
\beta &\leq \left(1 - \alpha + \frac{2b_\alpha}{m} \right) \cdot \frac{2}{\varepsilon \cdot \log m - c} \\
&= \left(1 - \alpha + \frac{2b_\alpha}{m} \right) \cdot \frac{2 \cdot \left(1 + \frac{c}{\varepsilon \cdot \log m - c} \right)}{(\varepsilon \cdot \log m - c) \cdot \left(1 + \frac{c}{\varepsilon \cdot \log m - c} \right)} \\
&= \left(1 - \alpha + \frac{2b_\alpha}{m} \right) \cdot \frac{2 \cdot \left(1 + \frac{c}{\varepsilon \cdot \log m - c} \right)}{\varepsilon \cdot \log m} \\
&= \left(1 - \alpha + \frac{2b_\alpha}{m} \right) \cdot \left(\frac{2}{\varepsilon \cdot \log m} + \frac{2c}{(\varepsilon \cdot \log m)^2 - c\varepsilon \cdot \log m} \right) \\
&= (1 - \alpha) \cdot \left(\frac{2}{\varepsilon \cdot \log m} + O\left(\frac{1}{\log^2 m} \right) \right) + O\left(\frac{1}{m \log m} \right) \\
&\leq (1 - \alpha) \cdot \frac{2}{\varepsilon \cdot \log m} + O\left(\frac{1}{\log^2 m} \right)
\end{aligned}$$

□

Note that as α approaches 1 (optimal consistency), β goes to $O(1/\log^2 m)$ (worst-case robustness), and as β goes to $\frac{2}{\varepsilon \log m}$ (optimal robustness), α goes to $O(1/\log m)$ (worst-case consistency).

4.4.2 RobustTrust

Next, we present a family of randomized algorithms, RobustTrust, which has a parameter $0 \leq \alpha \leq 1$ and works as follows. With a probability of α, RobustTrust applies TrustGreedy. (Applying Trust, instead of TrustGreedy, gives the same consistency and robustness results.) With probability $1 - \alpha$, RobustTrust ignores the predictions, and applies the Classify-and-Randomly-Select (CRS) algorithm described in Theorem 13.7 in [19]. CRS is strictly $\lceil \log m \rceil$-competitive (they use competitive ratios at least one in a version of the problem where requests have a limited time duration). A similar algorithm was originally proven $O(\log m)$-competitive in [8].

For completeness, we include the CRS algorithm. To avoid the problem of m possibly not being a power of 2, we define $\ell = \lceil \log m \rceil$ and $m' = 2^\ell$. Thus, the algorithm will define its behavior for a longer line and some sequences that cannot exist.

We define a set of ℓ *levels* for the possible requests. Since m' is a power of two, there is an odd number of edges, so the middle edge, e_1, in the line is well defined. We define the set $E_1 = \{e_1\}$ and let Level 1 consist of all intervals containing e_1. After Levels 1 through i are defined, we define E_{i+1} and Level $i + 1$ as follows: After removing all edges in $E_1 \cup E_2 \cup \cdots \cup E_i$ from the line, we are left with 2^i segments, each consisting of $2^{\ell-i}$ vertices. The set E_{i+1} consists of the middle edges of these segments, and Level $i + 1$ consists of all intervals, not in any of the Levels 1 through i, but containing an edge in E_{i+1}. Thus, the levels create a partition of all possible intervals.

The algorithm CRS initially chooses a level i between 1 and ℓ, each with probability $\frac{1}{\ell}$. It accepts any interval in Level i that does not overlap an interval it already has accepted. Any intervals not in Level i are rejected.

Theorem 7 For the online interval scheduling problem, RobustTrust (RT) with parameter α has consistency at least α and robustness at least $\frac{1-\alpha}{\lceil \log m \rceil}$.

Proof We investigate RobustTrust when all predictions are correct (consistency) and when predictions may be incorrect (robustness).

Suppose all predictions are correct. RobustTrust applies TrustGreedy with probability α. Since TrustGreedy is optimal when all predictions are correct, the expected profit of RobustTrust is at least $\alpha \cdot \text{Opt}$. Therefore, the competitive ratio (consistency) of RobustTrust is at least α.

Suppose some predictions are incorrect. If the intervals in Level i are the only intervals given, and CRS chooses that level, then CRS accepts as many intervals as OPT does, since each interval in Level i contains an edge in E_i, and no intervals containing more than one edge in E_i exist. Since the number of levels is $\lceil \log m \rceil$, the expected number of intervals that CRS accepts from any given level of OPT's configuration is at least $\frac{1}{\lceil \log m \rceil}$ times the number of intervals OPT accepted from that level, so by the linearity of expectations, this totals $\frac{1}{\lceil \log m \rceil}$ OPT. CRS is chosen with probability $1 - \alpha$, so the robustness is at least $\frac{1-\alpha}{\lceil \log m \rceil}$. □

4.5 Experimental Results

We present an experimental evaluation of TRUST and TRUSTGREEDY for interval scheduling in comparison with the GREEDY algorithm, which serves as a baseline online algorithm, and OPT, which serves as the performance upper bound. Our code and results are available at [38].

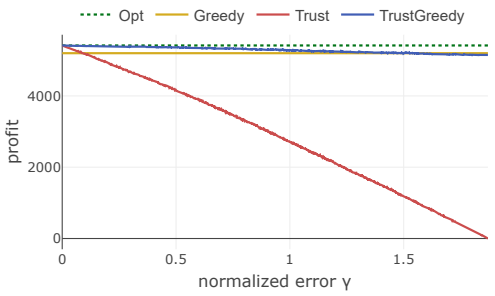
We evaluate our algorithms using real-world scheduling data for parallel machines [25]. Each benchmark from [25] specifies the start and finish times of tasks as scheduled on parallel machines with several processors. We use these tasks to generate inputs to the interval scheduling problem; Table 1 details the interval scheduling inputs we generated from benchmarks of [25]. For each benchmark with N tasks, we create an instance I of an interval scheduling problem by randomly selecting $n = \lfloor N/2 \rfloor$ tasks from the benchmark and randomly permuting them. This sequence serves as the input to all algorithms. To generate the prediction, we consider 1000 equally distanced values of $d \in [0, n]$. For each value of d, we initiate the prediction set $\hat{I}$ with the set of intervals in I, remove $|\,\mathrm{FN}\,| = d$ randomly selected intervals from $\hat{I}$ and add to it $|\,\mathrm{FP}\,| = d$ randomly selected intervals from the remain-

name	input size (N)	no. timesteps (m)	max. length	avg. length
LLNL-uBGL-2006-2	13,225	16,671,553	14,403	1,933.92
NASA-iPSC-1993-3.1	18,066	7,947,562	62,643	772.21
CTC-SP2-1996-3.1	77,205	8,986,769	71,998	11,279.61
SDSC-DS-2004-2.1	84,893	31,629,689	6,589,808	7,579.36

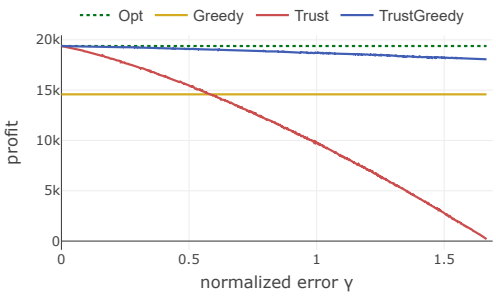
Table 1: Details on the benchmarks from [25] used in our experiments.

ing $N - n$ tasks in the benchmark. The resulting set $\hat{I}$ is given to TRUST and TRUSTGREEDY as prediction $\hat{I}$. For each value of d, we compute the normalized error $\gamma(\hat{I}, I) = \frac{\text{OPT}(\text{FN} \cup \text{FP})}{\text{OPT}(I)}$, and report the profit of TRUST and TRUSTGREEDY as a function of γ.

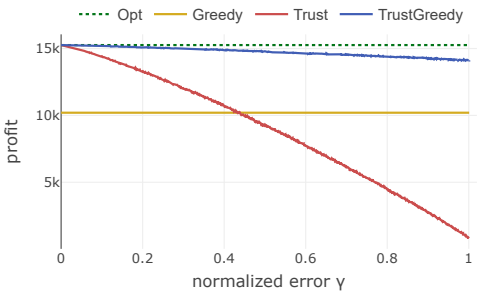
Figure 4 shows the results for two representative benchmarks from [25], namely, LLNL (the workload of the BlueGene/L system installed at Lawrence Livermore National Lab), SDSC (the workload log from San Diego Supercomputer Center), NASA-iPSC (scheduling log from Numerical Aerodynamic Simulation -NAS- Systems Division at NASA Ames Research Center) and CTC-SP2 (Cornell Theory Center IBM SP2 log). These four benchmarks are selected to represent a variety of input sizes and interval lengths. The results are aligned with our theoretical findings: TRUST quickly becomes worse than GREEDY as the error value increases, while TRUSTGREEDY degrades gently as a function of the prediction error. In particular, TRUSTGREEDY is better than GREEDY for almost all error values. We note that GREEDY



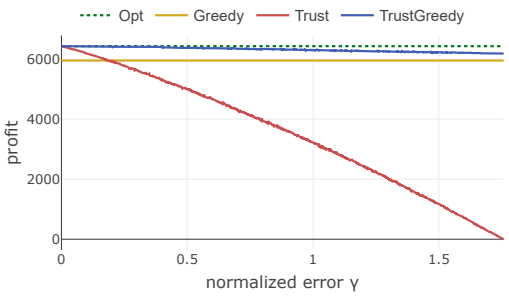
(a) LLNL-uBGL-2006-2



(b) SDSC-DS-2004-2.1-cln



(c) CTC-SP2-1996-3.1-cln

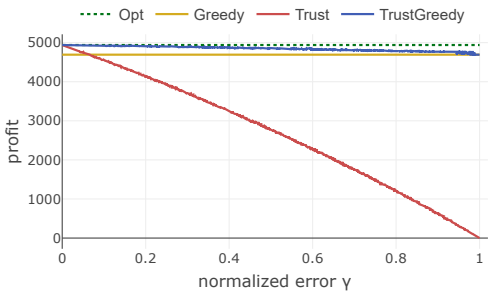


(d) NASA-iPSC-1993

Figure 4: Profit as a function of normalized error value

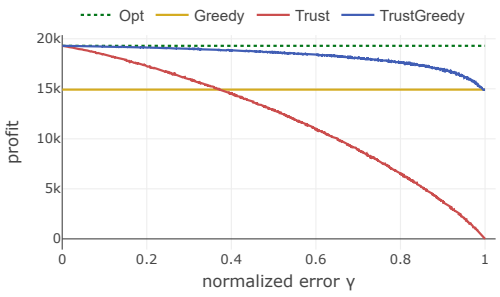
performs better when there is less overlap between the input intervals, which is the case in LLNL compared to SDSC. In an extreme case, when no two intervals overlap, GREEDY is trivially optimal. Nevertheless, even for LLNL, TRUSTGREEDY is not much worse than GREEDY for extreme values of error: the profit for the largest normalized error of $\gamma = 1.87$ was 5149 and 5198 for TRUSTGREEDY and GREEDY, respectively. Note that for SDSC, where there are more overlaps between intervals, TRUSTGREEDY is strictly better than GREEDY, even for the largest error values. It is worth noting that, in an extreme case, where $\mathrm{FP} = \mathrm{FN} = n$, the predictions contain a completely different set from the input sequence. In that case, $|\,\mathrm{FP} \cup \mathrm{FN}\,| = 2n$, and $\gamma = \frac{\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})}{\textsc{Opt}(I)}$ takes values in $[1.5, 2]$.

We also experiment in a setting where false positives and negatives contribute differently to the error set. We generate the input sequences in the



(a) LLNL-uBGL-2006-2, no FP

(b) LLNL-uBGL-2006-2, no FN



(c) SDSC-DS-2004-2.1-cln, no FP

(d) SDSC-DS-2004-2.1-cln, no FN

Figure 5: Profit as a function of normalized error value in the absence of false positives (a), (c) and false negatives (b), (d).

same way as in the previous experiments. To generate the prediction set $\hat{I}$, we consider 1000 equally-distanced values of d in the range $[0, n]$ as before. We first consider a setting in which all error is due to false negatives; for that, we generate $\hat{I}$ by removing d randomly selected intervals from I. In other words, $\hat{I}$ is a subset of the intervals in I. Figures 5a and 5c illustrate the profit of Trust and TrustGreedy in this case. We note that TrustGreedy is strictly better than both Trust and Greedy. In an extreme case, when $d = n$, $\hat{I}$ becomes empty and TrustGreedy becomes Greedy; in other words, Greedy is the same algorithm as TrustGreedy with the empty predictions set $\hat{I}$.

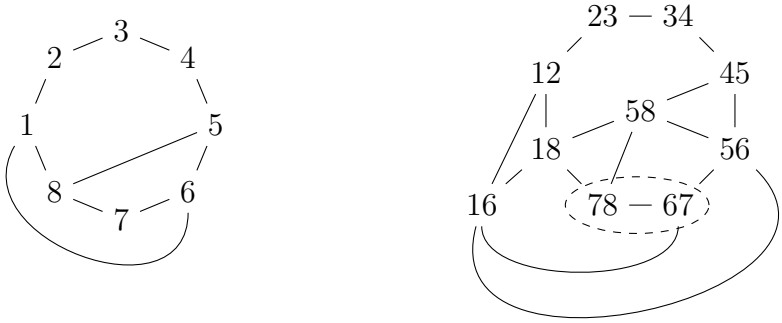
We also consider a setting in which there are no false negatives. For that, we generate $\hat{I}$ by adding d intervals to $\hat{I}$. In other words, $\hat{I}$ will be a superset of intervals in I. Figures 5a and 5c illustrate the profit of Trust and TrustGreedy in this case. In this case, the profit of Trust and TrustGreedy is similar to the setting where both false positives and negatives contributed to the error set. In particular, Trust quickly becomes worse than Greedy as the error increases, while TrustGreedy degrades gently as a function of the prediction error.

5 Related Problems: Matching and Independent Set

In [33], the authors observe that finding disjoint paths on stars is equivalent to finding maximal matchings on general graphs, where each request in the input to the disjoint path allocation problem bijects to an edge in the input graph for the matching problem. Therefore, we can extend the results of Section 3 to the following *online matching problem*. The input is a graph $G = (V, E)$, where V is known, and edges in E appear in an online manner; upon arrival of an edge, it must be added to the matching or rejected. The prediction is a set $\hat{E}$ that specifies edges in E. As before, we use FP and FN to indicate the set of false positives and false negatives and define

$$\gamma(\hat{E}, E) = \frac{\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})}{\textsc{Opt}(E)},$$

where $\textsc{Opt}(E)$ indicates the size of an optimal matching for graph $G = (V, E)$.



(a) The graph, G, for matching corresponding to the star graph for disjoint paths with requests $(1,2)$, $(2,3)$, $(3,4)$, $(4,5)$, $(5,6)$, $(6,7)$, $(7,8)$, $(1,8)$, $(1,6)$, and $(5,8)$.

(b) The line graph, G', corresponding to G, where the two digits in a vertex name in G' indicate the edge (given by its two endpoints) from G that the vertex corresponds to. The dashed ellipse indicates the contraction of the two vertices showing that G' contains $K_{2,3}$ as a minor.

Figure 6: Graphs for matching and independent set

The correspondence between the two problems is as follows: Consider a set of requests on a star. Each such request is a pair of vertices. We can assume no pair contains the star's center since all such requests should be accepted if they can be. For the matching problem, the pairs of vertices from the disjoint paths problem on the star can be the edges in the graph. A feasible solution to the disjoint paths problem corresponds to a matching and vice versa. One can similarly consider an instance of a matching problem, and the endpoints of the edges can be the non-center vertices of the star in the disjoint paths problem.

Using this correspondence between disjoint paths on a star and matchings in general graphs, for the star S_8 with non-center vertices $1, 2, \ldots, 8$ and requests $(1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (1,8), (1,6), (5,8)$, we get

the graph $G = (V, E)$ for matching, where

$$V = \{1, 2, 3, 4, 5, 6, 7, 8\} \text{ and}$$
$$E = \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (1, 8), (1, 6), (5, 8)\}.$$

See also Figure 6. Note that the edges in this graph correspond to the requests that are used in the proof of Theorem 2. The proof can be simulated in this new setting so that the number of requests accepted in the different cases in Theorem 2 is the same as the number of edges in the matchings found in the corresponding subgraphs of G. Thus, the same result holds for matchings in any graph class containing this graph.

All edges have one even-numbered endpoint and one odd, so this includes the bipartite graph class. It is also planar but not an interval or chordal graph.

Given the correspondence between interval scheduling and the matching problem, the following is immediate from Theorems 1 and 2.

Corollary 2 There is a strictly $(1 - 2\gamma)$-competitive algorithm, TRUST, for the online matching problem under the edge-arrival model. For $0 \leq \gamma \leq 1/4$, this is optimal among deterministic algorithms, even on bipartite graphs as well as planar graphs.

Using the correspondence between matchings in a graph, G, and an independent set in the line graph of G, we can get the same result for the *independent set problem* on line graphs. The line graph of a graph, G, has a vertex for each edge in G and an edge between two vertices if the corresponding edges in G share a vertex.

The line graph $G' = (V', E')$ of the graph above used for matching is defined by

$$V' = \{12, 23, 34, 45, 56, 67, 78, 18, 16, 58\},$$

where, for brevity, we use the notation 12 to denote the vertex corresponding to the edge $(1, 2)$ from G. The set of edges is then

$$E' = \{(12, 23), (23, 34), (34, 45), (45, 56), (56, 67), (67, 78), (78, 18), (18, 16), (16, 12), (58, 18), (58, 78), (58, 56), (58, 45), (12, 18), (16, 67), (16, 56)\}.$$

Requests from the proof in Theorem 2 correspond to vertices here. See also Figure 6.

We note that the graph G' is planar, but not outerplanar, since, contracting 67 and 78 into one vertex, 67-78, the sets $\{16, 58\}$ and $\{18, 56, 67\text{-}78\}$ form a $K_{2,3}$ minor, which is a so-called forbidden subgraph for outerplanarity [26, 35]. Also, it is not chordal. However, the lower bound of $1 - \gamma$ for deterministic interval scheduling algorithms (Theorem 3) clearly holds for independent sets in interval graphs, too, by considering the interval graph corresponding to a set of intervals on the line.

Summing up, we obtain:

Corollary 3 For the online independent set problem under the vertex-arrival model, the following hold.

- On line graphs, there is a strictly $(1 - 2\gamma)$-competitive algorithm, Trust.
- For $0 \leq \gamma \leq 1/4$, no deterministic algorithms can be better than $(1 - 2\gamma)$-competitive, on line graphs as well as planar graphs.
- On interval graphs, no deterministic algorithm is better than $(1 - \gamma)$-competitive.

References

[1] ALPS. Algorithms with predictions. `https://algorithms-with-predictions.github.io/`, 2022. Accessed: 2024-12-20.

[2] Spyros Angelopoulos, Diogo Arsénio, and Shahin Kamali. Competitive sequencing with noisy advice. , 2021. arxiv:2111.05281 [cs.DS].

[3] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc Renault. Online computation with untrusted advice. *Journal of Computer and System Sciences*, 144(103545):1–14, 2024.

[4] Spyros Angelopoulos and Shahin Kamali. Contract scheduling with predictions. *Journal of Artificial Intelligence Research*, 77:395–426, 2023.

[5] Spyros Angelopoulos, Shahin Kamali, and Dehou Zhang. Online search with best-price and query-based predictions. In *37th Conference on Artificial Intelligence (AAAI)*, pages 9652–9660. The Association for the Advancement of Artificial Intelligence, 2023.

[6] Antonios Antoniadis, Joan Boyar, Marek Eliáš, Lene M. Favrholdt, Ruben Hoeksma, Kim S. Larsen, Adam Polak, and Bertrand Simon. Paging with Succinct Predictions. In *40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 952–968. PMLR, 2023.

[7] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. *Discrete Optimization*, 48, part 2(100778), 2023.

[8] Baruch Awerbuch, Yair Bartal, Amos Fiat, and Adi Rosén. Competitive non-preemptive call control. In *5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 312–320. ACM/SIAM, 1994.

[9] Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1070–1080, 2021.

[10] Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 35–66. SIAM, 2022.

[11] Eric Balkanski, Vasilis Gkatzelis, and Xizhi Tan. Strategyproof scheduling with predictions. In *14th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 251 of *LIPIcs*, pages 11:1–11:22, Saarbrücken/Wadern, 2023. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[12] Evripidis Bampis, Konstantinos Dogeas, Alexander Kononov, Giorgio Lucarelli, and Fanny Pascual. Scheduling with untrusted predictions. In *31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4581–4587, 2022.

[13] Siddhartha Banerjee, Vincent Cohen-Addad, Anupam Gupta, and Zhouzi Li. Graph searching with predictions. In *14th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 251 of *LIPIcs*, pages 12:1–12:24, Saarbrücken/Wadern, 2023. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[14] Kfir Barhum, Hans-Joachim Böckenhauer, Michal Forišek, Heidi Gebauer, Juraj Hromkovič, Sacha Krug, Jasmin Smula, and Björn Steffen. On the power of advice and randomization for the disjoint path allocation problem. In *40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, volume 8327 of *Lecture Notes in Computer Science*, pages 89–101, Berlin, Heidelberg, 2014. Springer.

[15] Hans-Joachim Böckenhauer, Nina Corvelo Benz, and Dennis Komm. Call admission problems on trees. *Theoretical Computer Science*, 922:410–423, 2022.

[16] Hans-Joachim Böckenhauer, Fabian Frei, and Silvan Horvath. Priority algorithms with advice for disjoint path allocation problems. *Theoretical Computer Science*, 1021(114942):1–17, 2024.

[17] Hans-Joachim Böckenhauer, Dennis Komm, and Raphael Wegner. Call admission problems on grids with advice. *Theoretical Computer Science*, 918:77–93, 2022.

[18] Allan Borodin, Joan Boyar, Kim S. Larsen, and Nazanin Mirmohammadi. Priority Algorithms for Graph Optimization Problems. *Theoretical Computer Science*, 411(1):239–258, 2010.

[19] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[20] Allan Borodin and Christodoulos Karavasilis. Any-order online interval selection. In *21st International Workshop Approximation and Online Algorithms (WAOA)*, volume 14297 of *Lecture Notes in Computer Science*, pages 175–189, Berlin, Heidelberg, 2023. Springer.

[21] Allan Borodin, Morten N. Nielsen, and Charles Rackoff. (Incremental) priority algorithms. *Algorithmica*, 37:295–326, 2003.

[22] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, and Jesper W. Mikkelsen. The advice complexity of a class of hard online problems. *Theory of Computing Systems*, 61(4):1128–1177, 2017.

[23] Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. Online unit profit knapsack with predictions. *Algorithmica*, 86(9):2786–2821, 2024.

[24] Joan Boyar, Kim S. Larsen, and Denis Pankratov. Advice Complexity of Adaptive Priority Algorithms. *Theoretical Computer Science*, 984(114318):1–31, 2024.

[25] Steve J. Chapin, Walfredo Cirne, Dror G. Feitelson, James Patton Jones, Scott T. Leutenegger, Uwe Schwiegelshohn, Warren Smith, and David Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In *Job Scheduling Strategies for Parallel Processing (IPPS/SPDP)*, volume 1659 of *Lecture Notes in Computer Science*, pages 67–90, Berlin, Heidelberg, 1999. Springer.

[26] Gary Chartrand and Frank Harary. Planar permutation graphs. *Annales de l'Institut Henri Poincaré, série B – Probabilités et Statistiques*, 3(4):433–438, 1967.

[27] Justin Y Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David P Woodruff, and Michael Zhang. Triangle and four cycle counting with predictions in graph streams. In *10th International Conference on Learning Representations (ICLR)*, 2022.

[28] Justin Y. Chen, Sandeep Silwal, Ali Vakilian, and Fred Zhang. Faster fundamental graph algorithms via learned predictions. In *39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pages 3583–3602. PMLR, 2022.

[29] Sashka Davis and Russell Impagliazzo. Models of greedy algorithms for graph problems. *Algorithmica*, 54(3):269–317, 2009.

[30] Franziska Eberle, Alexander Lindermayr, Nicole Megow, Lukas Nölke, and Jens Schlöter. Robustification of online graph exploration methods. In *36th Conference on Artificial Intelligence (AAAI)*, pages 9732–9740. The Association for the Advancement of Artificial Intelligence, 2022.

[31] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

[32] András Frank. Edge-disjoint paths in planar graphs. *Journal of Combinatorial Theory, Series B*, 39(2):164–178, 1985.

[33] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1977.

[34] Heidi Gebauer, Dennis Komm, Rastislav Královic, Richard Královic, and Jasmin Smula. Disjoint path allocation with sublinear advice. In *21st International Conference on Computing and Combinatorics (COCOON)*, volume 9198 of *Lecture Notes in Computer Science*, pages 417–429, Berlin, Heidelberg, 2015. Springer.

[35] Frank Harary. *Graph Theory*. Addison-Wesley, Reading, Massachusetts, 1969.

[36] Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Online knapsack with frequency predictions. In *34th Advances in Neural Information Processing Systems (NeurIPS)*, pages 2733–2743. Curran Associates, Inc., 2021.

[37] Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. *ACM Transactions on Parallel Computing*, 10(4):19:1–19:26, 2023.

[38] Shahin Kamali. Interval scheduling with prediction. `https://github.com/shahink84/IntervalSchedulingWithPrediction`, 2022. Accessed: 2024-09-28.

[39] Antoon W. J. Kolen, Jan Karel Lenstra, Christos H. Papadimitriou, and Frits C. R. Spieksma. Interval scheduling: A survey. *Naval Research Logistics*, 54(5):530–543, 2007.

[40] Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving online algorithms via ML predictions. In *31st Advances in Neural Information Processing Systems (NeurIPS)*, pages 9684–9693. Curran Associates, Inc., 2018.

[41] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1859–1877. SIAM, 2020.

[42] 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 (ESA)*, volume 204 of *LIPIcs*, page 59:1–59:17, Saarbrücken/Wadern, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[43] Thomas Lavastida, Benjamin Moseley, R Ravi, and Chenyang Xu. Using predicted weights for ad delivery. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA)*, pages 21–31. SIAM, 2021.

[44] Russell Lee, Jessica Maghakian, Mohammad Hajiesmaili, Jian Li, Ramesh Sitaraman, and Zhenhua Liu. Online peak-aware energy scheduling with untrusted advice. *ACM SIGENERGY Energy Informatics Review*, 1(1):59–77, 2021.

[45] Richard J. Lipton and Andrew Tomkins. Online interval scheduling. In *5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 302–311. ACM/SIAM, 1994.

[46] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 3302–3311. PMLR, 2018.

[47] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM*, 68(24), 2021.

[48] Kazuhiko Matsumoto, Takao Nishizeki, and Nobuji Saito. An efficient algorithm for finding multi-commodity flows in planar networks. *SIAM Journal on Computing*, 14(2):289–302, 1985.

[49] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020.

[50] Takao Nishizeki, Jens Vygen, and Xiao Zhou. The edge-disjoint paths problem is NP-complete for series–parallel graphs. *Discrete Applied Mathematics*, 115:177–186, 2001.

[51] Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1834–1845. SIAM, 2020.

[52] Dorothea Wagner and Karsten Weihe. A linear-time algorithm for edge-disjoint paths in planar graphs. *Combinatorica*, 15:135–150, 1995.

[53] Alexander Wei. Better and simpler learning-augmented online caching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 176 of *LIPIcs*, pages 60:1–60:17, Saarbrücken/Wadern, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[54] Alexander Wei and Fred Zhang. Optimal robustness-consistency tradeoffs for learning-augmented online algorithms. In *33rd Advances in Neural Information Processing Systems (NeurIPS)*, pages 8042–8053. Curran Associates, Inc., 2020.

[55] Ali Zeynali, Bo Sun, Mohammad Hajiesmaili, and Adam Wierman. Data-driven competitive algorithms for online knapsack and set cover. In *35th Conference on Artificial Intelligence (AAAI)*, pages 10833–10841. The Association for the Advancement of Artificial Intelligence, 2021.