Online Interval Scheduling with Predictions*

Joan Boyar [1], Lene M. Favrholdt [1], Shahin Kamali [2], and Kim S. Larsen [1]

[1] University of Southern Denmark
{joan,lenem,kslarsen}@imada.sdu.dk
https://imada.sdu.dk/u/{joan,lenem,kslarsen}
[2] York University
kamalis@yorku.ca
https://www.eecs.yorku.ca/~kamalis/

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 involving a pair of vertices of a known graph, and the goal is to accept a maximum number of requests forming edge-disjoint paths between accepted pairs. These problems have been studied under extreme settings without information about the input or with error-free advice. We study an intermediate setting with a potentially erroneous prediction that specifies the set of intervals/requests forming the input sequence. For both problems, we provide tight upper and lower bounds on the competitive ratios of online algorithms as a function of the prediction error. For disjoint path allocation, our results rule out the possibility of obtaining a better competitive ratio than that of a simple algorithm that fully trusts predictions, whereas, for interval scheduling, we develop a superior algorithm. 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.

Keywords: Online interval scheduling. Algorithms with prediction. Competitive analysis. Disjoint paths.

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 one of their endpoints. The objective is to maximize the number of accepted intervals, referred to as the *payoff* of the scheduler. This problem is also known as *fixed job scheduling* and *k-track assignment* [33].

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 edge-disjoint paths between vertices of accepted pairs. 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 [27] and outerplanar graphs by a combination of [23,32,26], but the problem is NP-complete for general graphs [25], and even on quite restricted graphs such as series-parallel graphs [40]. The disjoint path problem is the same as call control/call allocation with 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.

* The first, second, and fourth authors were supported in part by the Danish Council for Independent Research grant DFF-0135-00018B.
The third author was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

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 sequence I of intervals. A new request must either be irrevocably accepted or rejected, subject to maintaining disjoint paths between accepted requests. We analyze an online algorithm via a comparison with an optimal offline algorithm, Opt. The *competitive ratio* of an online algorithm Alg is defined as $\inf_I \{\textsc{Alg}(I)/\textsc{Opt}(I)\}$, where $\textsc{Alg}(I)$ and $\textsc{Opt}(I)$, respectively, denote the payoff of Alg and Opt for intervals in I (for randomized algorithms, $\textsc{Alg}(I)$ is the expected payoff of Alg). Since we consider a maximization problem, our ratios are between zero and one.

For interval scheduling on a path graph with m edges, the competitive ratios of the best deterministic and randomized algorithms are respectively m and $\lceil \log m \rceil$ [16]. 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 [13,28], trees [14], and grid graphs [15]. 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.

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 machine-learned information in the form of a prediction. 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 [41,43], makespan scheduling [34], contract scheduling [4,5], and other variants of scheduling problems [8,37,11,10]. Other online problems studied under the prediction model include bin packing [2,3], knapsack [44,31,17], caching [38,42], matching problems [6,35,36], and various graph problems [22,24,21,9,12]. See also the survey by Mitzenmacher and Vassilvitskii [39] and the collection at [1].

1.1 Contributions

We study the disjoint path allocation problem under a setting where the scheduler is provided with a set $\hat{I}$ of intervals predicted to form the input sequence I. Given the erroneous nature of the prediction, some intervals in $\hat{I}$ may be incorrectly predicted to be in I (false positives), and some intervals in I may not be included in $\hat{I}$ (false negatives). We let the *error set* be the set of intervals 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 non-overlapping intervals in the error set, i.e., $\eta(\hat{I}, I) = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$. We explain later that this definition of η satisfies specific desired properties for the prediction error (Proposition 1). In the following, we use $\textsc{Alg}(\hat{I}, I)$ to denote the payoff 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 first study a simple algorithm Trust, which accepts a request only if it belongs to the set of intervals in a given optimal solution for $\hat{I}$. We show that, for any graph G, any input sequence I, and any prediction $\hat{I}$, $\textsc{Trust}(\hat{I}, I) \geq (1 - 2\gamma(\hat{I}, I))\,\textsc{Opt}(I)$ (Theorem 1). Furthermore, for any algorithm Alg and any positive integer p,

  there are worst-case input sequence I_w and prediction set $\hat{I}_w$ over a star graph with $8p$ leaves, such that $\eta(\hat{I}_w, I_w) = p$ and $\textsc{Alg}(\hat{I}_w, I_w) \leq (1 - 2\gamma(\hat{I}_w, I_w))\,\textsc{Opt}(I_w)$ (Theorem 2). Thus, Trust achieves an optimal competitive ratio in any graph that contains S_8 as a subgraph, i.e., any graph of maximum degree at least 8.

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 a negative result for deterministic interval scheduling algorithms. Given any deterministic algorithm ALG and integer p, we show there are worst-case instances I_w and predictions $\hat{I}_w$ such that $\eta(\hat{I}_w, I_w) = p$ and $\text{ALG}(\hat{I}_w, I_w) \leq (1-\gamma(\hat{I}_w, I_w))\,\text{OPT}(I_w)$ (Theorem 3, setting $c = 2$).

  Next, we present a negative result for TRUST. For any positive integer, p, we show there are worst-case instances I_w and predictions $\hat{I}_w$ such that $\eta(\hat{I}_w, I_w) = p$ and $\text{TRUST}(\hat{I}_w, I_w) = (1 - 2\gamma(\hat{I}_w, I_w))\,\text{OPT}(I_w)$. (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 for interval scheduling. 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 payoff of the maintained solution. For any input I and prediction $\hat{I}$, we show that $\text{TRUSTGREEDY}(\hat{I}, I) \geq (1-\gamma(\hat{I}, I))\,\text{OPT}(I)$ (Theorem 5), which proves optimality of TRUSTGREEDY in the light of Theorem 3.

- **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) [38]. We focus on randomized algorithms because a non-trivial trade-off is infeasible for deterministic algorithms (Proposition 2). Suppose that for any input I, an algorithm ALG guarantees a consistency of $\alpha < 1$ and robustness of $\beta \leq \frac{1}{\lceil \log m \rceil}$. We show $\alpha \leq 1 - \frac{\lfloor \log m \rfloor - 1}{2}\beta$ and $\beta \leq \frac{2}{\lfloor \log m \rfloor - 1} \cdot (1-\alpha)$ (Theorem 6). For example, to guarantee a robustness of $\frac{1}{10\lfloor \log m \rfloor}$, the consistency must be at most 19/20, and to guarantee a consistency of $\frac{2}{3}$, the robustness must be at most $\frac{2}{3}\frac{1}{\lfloor \log m \rfloor - 1}$. 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 the online GREEDY algorithm (which accepts an interval if and only if it does not overlap previously accepted intervals), and OPT on real-world scheduling data from [19]. 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.

2 Model and Predictions

We assume that an oracle provides the online algorithm with a set $\hat{I}$ of requests predicted to form the input sequence I. 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 $cm/(e \ln 2)$ bits [18].

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, $I = \text{TP} \cup \text{FN}$ and $\hat{I} = \text{TP} \cup \text{FP}$. We use $\eta(\hat{I}, I)$, to denote the error for the input formed by the set I, when the set of predictions is $\hat{I}$. 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 $\gamma = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) / \textsc{Opt}(I)$. Our error measure satisfies the following desirable properties, the first two of which were strongly recommended in Im, et al. [30]: $\eta(I, \hat{I}) \leq \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$. In Section 2, we discuss natural error models, such as Hamming distance between the request sequence and prediction, and explain why these measures do not satisfy our desired properties.

- *Monotonicity:* This property ensures that increasing 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:* Let $\textsc{Opt}(I)$ denote the number of requests in an optimal solution for the input sequence, and $\textsc{Opt}(\hat{I})$ denote the number of requests in an optimal solution for a set of predicted requests. The Lipschitz 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 this property ensures that the error is not "too small". In particular, we should not be able to decrease the error to an arbitrarily small value by adding "dummy requests". For example, *false discovery rate*, defined as $\frac{|\,\mathrm{FP}\,|}{|\,\mathrm{FP}\,|+|\,\mathrm{TP}\,|}$, does not satisfy Lipschitz property: an adversary can construct a bad input and then add a lot of intervals to $I \cap \hat{I}$, contributing to $|\,\mathrm{TP}\,|$, that neither the algorithm nor Opt will choose, driving down the error.
- *Lipschitz completeness (or simply completeness):* We need the error measure to ensure that the error is not "too large". Consider the following example for the disjoint paths problem. 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 true 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 $\mathrm{FN} = 2$. In this case, 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 Lipschitz completeness, which requires
$$\eta(I, \hat{I}) \leq \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}).$$

Proposition 1. *The error measure $\eta(\hat{I}, I) = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$ satisfies the properties of monotonicity, Lipschitz, and Lipschitz completeness.*

Proof. We check all properties listed above:

- 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 $\eta = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$. □

Alternative Error Measures. In what follows, we review a few alternative error measures that do not satisfy our desired properties of monotonicity, Lipschitz, and Lipschitz completeness (or simply completeness).

– Hamming distance between the bit strings representing the request sequence and the predictions:

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

It fails completeness.

– 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 completeness, according to the example given in connection with the definition of completeness.

– Either $|\,\mathrm{FP}\,|$ or $|\,\mathrm{FN}\,|$ fails Lipschitz property.

– 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.

– We also considered normalizing by the total number of possible intervals (order m^2), but this measure fails the Lipschitz property, as we can make the error arbitrarily small by "scaling up" each edge to an arbitrarily long path, without changing algorithms' payoffs.

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 any graph of maximal degree at least 8. Trust simply relies on the predictions being correct. Specifically, it computes an optimal solution I^* in $\hat{I}$ before processing the first request. Then, it accepts any interval 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 (i) false negatives cause a deficit of at most $\textsc{Opt}(\mathrm{FN})$ in the schedule of Trust compared to the optimal schedule for I^*, (ii) false positives cause a deficit of at most $\textsc{Opt}(\mathrm{FP})$ in the optimal schedule of I^*, compared to the optimal schedule for I, and (iii) $\textsc{Opt}(\mathrm{FP}) + \textsc{Opt}(\mathrm{FN}) \leq 2\,\textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = 2\eta$.

Theorem 1. *For any graph G, any prediction $\hat{I}$, and input sequence I, we have* $\textsc{Trust}(\hat{I}, I) \geq (1 - 2\gamma(\hat{I}, I))\,\textsc{Opt}(I)$.

Proof. Since I^* is an optimal selection from $\mathrm{TP} \cup \mathrm{FP}$, the largest number of intervals 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 intervals that can be detracted from Trust is realized when intervals 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,

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

□

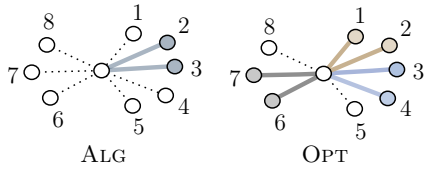
The following result shows that Theorem 1 is tight for star graphs of degree 8. One can conclude that Trust is optimal for any graph that contains stars of degree 8 as a subgraph, i.e., any graph of maximal degree at least 8.

Theorem 2. *Let* Alg *be any deterministic algorithm and p be any positive integer. On the star graph, S_{8p}, there exists a set of predicted intervals $\hat{I}_w$ and a request sequence I_w such that $\eta(\hat{I}_w, I_w) = p$ and* $\textsc{Alg}(\hat{I}_w, I_w) \leq (1 - 2\gamma(\hat{I}, I))\,\textsc{Opt}(I_w)$.

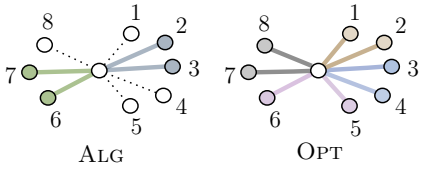
Proof. We consider the non-center vertices of S_{8p} in p groups of eight, and handle them all identically, one group at a time, treating each group independently. The prediction is fixed, but the input sequence depends on the algorithm's actions. For each group, we show that the error in the prediction is 1, and the payoff of Opt is at least 2 units more than that of Alg. Given that groups do not share edges between themselves, the total error and algorithms' payoffs are summed over all groups. Hence, the total error will be equal to $\eta(\hat{I}_w, I_w) = p$, and we can write $\textsc{Alg}(I_w) \leq \textsc{Opt}(I_w) - 2\eta(\hat{I}_w, I_w)$, that is, $\textsc{Alg}(I_w) \leq (1 - 2\gamma(\hat{I}_w, I_w))\,\textsc{Opt}(I_w)$.

Next, we explain how an adversary defines the input for each group. For group $0 \leq i \leq s - 1$, the non-center vertices are $8i + j$, where $1 \leq j \leq 8$, but we refer to these vertices by the value j. Let $\hat{I}_w = \{(1,2),(2,3),(3,4),(4,5),(6,7),(7,8)\}$ be the part of the prediction relevant for the current group of eight vertices. Both $(6,7)$ and $(7,8)$ are always included in the input sequence, with $(6,7)$ arriving immediately before $(7,8)$. Alg accepts at most one of them. This is discussed in the cases below. The first request in the input is always $(2,3)$, and Alg can either accept or reject it.

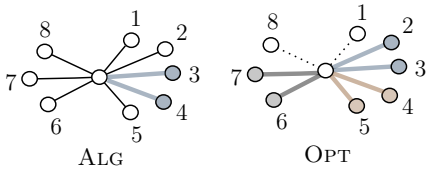
Case Alg accepts (2, 3): The next interval to arrive is $(6,7)$. If Alg rejects this interval, the next to arrive is $(7,8)$. If Alg also rejects this interval, then the intervals $(1,2)$ and $(3,4)$ also arrive, but $(4,5)$ is a false positive (see Figure 1a). Then, Opt accepts $\{(1,2),(3,4),(6,7)\}$, Alg only accepts $\{(2,3)\}$, and $\textsc{Opt}(\mathrm{FN} \cup \mathrm{FP}) = 1$. Thus, we may assume that Alg accepts at least one of $(6,7)$ and $(7,8)$, which we call $(7,x)$ where $x \in \{6,8\}$. We call the other of these two edges $(7,y)$. Then, the intervals



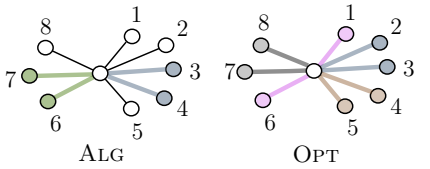
(a) ALG accepts $(2,3)$, rejects $(6,7)$ and $(7,8)$.



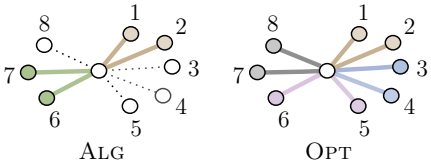
(b) ALG accepts $(2,3),(7,x)$; here $x=6$, $y=8$.



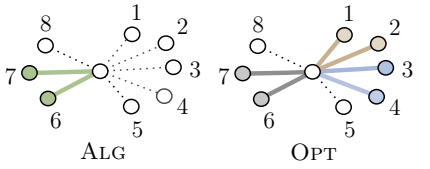
(c) ALG rejects $(2,3)$, accepts $(3,4)$, and rejects both $(6,7)$ and $(7,8)$.



(d) ALG rejects $(2,3)$, accepts $(3,4)$ and $(7,x)$; here $x=6$ and $y=8$.



(e) ALG rejects $(2,3)$, $(3,4)$, accepts $(7,x)$ and $(1,2)$; here $x=6$ and $y=8$.



(f) ALG rejects $(2,3)$, $(3,4)$, accepts $(7,x)$, and rejects $(1,2)$; here $x=6$ and $y=8$.

Fig. 1: Illustration of the proof of Theorem 2. Highlighted edges indicate paths between accepted pairs.

$(1,2)$ and $(3,4)$ also arrive, along with a false negative $(5,x)$. The interval $(4,5)$ is a false positive and is not in the input (see Figure 1b). Since $(4,5)$ and $(5,x)$ share an edge, $\text{OPT}(\text{FN}\cup\text{FP})=1$. ALG accepts $\{(2,3),(7,x)\}$, and OPT accepts $\{(1,2),(3,4),(5,x),(7,y)\}$. To conclude, the error increases by 1, and ALG's deficit to OPT increases by 2.

Case ALG rejects $(2,3)$: The next interval to arrive is $(3,4)$.

Subcase ALG accepts $(3,4)$: As in the previous case, we consider which of $(6,7)$ and $(7,8)$ ALG accepts. If neither is accepted, in addition to $(2,3)$, $(4,5)$ arrives, but $(1,2)$ is a false positive (Figure 1c). Again, payoffs of ALG and OPT are respectively 1 and 3, and $\text{OPT}(\text{FN}\cup\text{FP})=1$. The error is increased by 1, and the net advantage of OPT over ALG is increased by at least 2.

Next, we assume that ALG accepts $(7,x)$ and rejects $(7,y)$. Then, in addition to the intervals $(2,3)$ and $(3,4)$, $(4,5)$ arrives, along with a false negative $(1,x)$ (Figure 1d). The interval $(1,2)$ is a false positive and is not in the input. Since $(1,2)$ and $(1,x)$ share an edge, $\text{OPT}(\text{FN}\cup\text{FP})=1$. ALG accepts $\{(3,4),(7,x)\}$, and OPT accepts $\{(1,x),(2,3),(4,5),(7,y)\}$. Again, the error is increased by 1, and the net advantage of OPT over ALG is increased by 2.

Subcase ALG rejects $(3,4)$: The next interval to arrive is $(1,2)$.

Regardless of whether ALG accepts or rejects $(1,2)$, as in the previous cases, we consider which of $(6,7)$ and $(7,8)$ ALG accepts. If neither is accepted, then $(2,3)$ and $(3,4)$ have already arrived, but $(4,5)$ is a false positive. The payoff of ALG is at most 1 if it accepts $(1,2)$ and 0 otherwise, while OPT accepts $\{(1,2),(2,3),(3,4)\}$, and $\text{OPT}(\text{FN}\cup\text{FP})=1$. Thus, the error is increased by 1, and the net advantage of OPT over ALG is increased by 2. In what follows, we assume ALG accepts $(7,x)$ for $x\in\{6,8\}$.

Subsubcase Alg accepts (1, 2): Then, in addition to the intervals $(2, 3)$ and $(3, 4)$, a false negative, $(5, x)$, arrives. The interval $(4, 5)$ is a false positive and is not in the input. 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 $\{(1, 2), (3, 4), (5, x), (7, y)\}$ (Figure 1e). As before, the error is increased by 1, and the net advantage of Opt over Alg is increased by 2.

Subsubcase Alg rejects (1, 2): In this case, the interval $(4, 5)$ is a false positive, and there are no false negatives. Thus, the payoffs of Alg and Opt are respectively 1 and 3, and $|\,\text{Opt}(\text{FN} \cup \text{FP})| = 1$ (Figure 1f). That is, the error is increased by 1, and Alg's deficit compared to Opt is increased by 2.

This completes the proof for one group of eight vertices. Repeating it independently for each of the s groups of eight vertices gives the claimed result. □

4 Interval Scheduling

In this section, we show tight upper and lower bounds on the competitive ratio of a deterministic algorithm for interval scheduling. As an introduction to the difficulties in designing algorithms for the problem, we start by proving a general lower bound. We show that for any deterministic algorithm Alg, there exists an input sequence I_w and a set of predictions $\hat{I}_w$ such that $\text{Alg}(\hat{I}_w, I_w) = \text{Opt}(I_w) - \eta(\hat{I}_w, I_w)$, and that this can be established for any positive integer error. We also show that the competitive ratio of Alg is arbitrarily small.

Theorem 3. *Let* Alg *be any deterministic algorithm. For any positive integers p and $c \in [2, m]$, there are instances I_w and predictions $\hat{I}_w$ such that $p \leq \eta(\hat{I}_w, I_w) \leq (c-1)p$ and $\text{Alg}(\hat{I}_w, I_w) = (1 - \gamma(\hat{I}_w, I_w))\,\text{Opt}(I_w) \leq \frac{1}{c}\,\text{Opt}(I_w)$.*

Proof. Alg will be presented with p intervals of length c, and the remainder of the sequence will depend on which of these it accepts. The prediction, however, will include the following $2p$ requests: $\hat{I} = \bigcup_{i=0}^{p-1} \{(ci, c(i+1)), (ci, ci+1)\}$.

The input I_w is formed by p phases, $i \in [0, p-1]$. The ith phase starts with the true positive $(ci, c(i+1))$. There are two cases to consider:

- If Alg accepts $(ci, c(i+1))$, then the phase continues with $\{(ci+j, ci+(j+1)) \mid 0 \leq j \leq c-1\}$. The first of these requests is a true positive, and the other $c-1$ are false negatives. Note that 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.
- If Alg rejects $(ci, c(i+1))$, the phase ends with no further requests. In this case, $(ci, ci+1)$ is a false positive.

The contribution, η_i, of phase i to $|\,\text{FP} \cup \text{FN}\,|$ is $\eta_i = c - 1$ in the first case and $\eta_i = 1$ in the second. Since the intervals in $\text{FP} \cup \text{FN}$ are disjoint, we can write $\text{Opt}(\text{FP} \cup \text{FN}) = \sum_{i=0}^{p-1} \eta_i$ and it follows that $p \leq \text{Opt}(\text{FP} \cup \text{FN}) \leq (c-1)p$. Moreover, the net advantage of Opt over Alg in phase i is at least η_i: in the first case, Opt accepts $\eta_i + 1$ and Alg accepts one request, and in the second case, Opt accepts $\eta_i = 1$ and Alg accepts no requests. Given that there are p phases, we can write $\text{Alg}(\hat{I}_w, I_w) \leq \text{Opt}(I_w) - \sum_{i=0}^{p-1} \eta_i = \text{Opt}(I_w) - \text{Opt}(\text{FP} \cup \text{FN}) = (1 - \gamma(\hat{I}_w, I_w))\,\text{Opt}(I_w)$.

In phases where Alg accepts the first request, Opt accepts c times as many requests as Alg. In phases where Alg rejects the first request, Opt accepts one interval, and Alg accepts no intervals. Thus, $\text{Opt}(I_w) \geq c \cdot \text{Alg}(\hat{I}_w, I_w)$. □

For $c = 2$, we get $\eta(\hat{I}_w, I) = p$ and $\text{Alg}(\hat{I}_w, I_w) = (1 - \gamma(\hat{I}_w, I_w))\,\text{Opt}(\hat{I}_w)$. The next theorem shows that the competitive ratio of Trust compared to the lower bound of Theorem 3 is not tight. The

proof follows from an adversarial sequence similar to that of Theorem 3 in which the payoff of Opt and η grow in phases while the payoff of Trust stays 0.

Theorem 4. *For any integer $p \geq 1$, there exists a prediction $\hat{I}_w$ and an input sequence I_w so that $\eta(\hat{I}_w, I_w) = p$ and $\textsc{Trust}(\hat{I}_w, I_w) = (1 - 2\gamma(\hat{I}_w, I_w))\,\textsc{Opt}(I_w)$.*

Proof. Let the prediction be

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

Trust chooses an optimal solution I^* from $\hat{I}_w$. For each i, I^* will contain either $(3i, 3i+2)$ or $(3i+1, 3i+3)$. 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 in I_w. 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 in I_w. Further, I_w will then contain the FN-interval, $(3i+2, 3i+3)$.

Thus, $\textsc{Opt}(I_w) = 2p$, and for each i, the interval in FP and the interval in FN overlap, that so $\textsc{Opt}(\text{FN} \cup \text{FP}) = p$. Since $I^* = \text{FP}$, Trust does not accept any intervals, so

$$\textsc{Trust}(\hat{I}, I_w) = 0 = \textsc{Opt}(I) - 2\,\textsc{Opt}(\text{FN} \cup \text{FP}) = (1 - 2\gamma(\hat{I}_w, I_w))\,\textsc{Opt}(I_w).$$

□

4.1 TrustGreedy

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

The algorithm. TG starts by choosing an optimal solution offline set I^* of the schedules in $\hat{I}$, and plans to accept those intervals in I^* and reject all others, and it just follows its plan, except possibly when the next request is in FN. TG maintains an updated plan, A. Initially, A is I^*. When a request, r, is in FN, TG 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 TG, but they are not said to "replace" it. We let U denote the set of intervals in $I^* \cap \text{FP}$ that are not replaced during the execution of TG.

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

- $\text{TG}^{\text{FN}} = \text{TG} \cap \text{FN}$ and $\textsc{Opt}^{\text{FN}} = \textsc{Opt} \cap \text{FN}$
- $\text{TG}^{\text{TP}} = \text{TG} \cap \hat{I} = \text{TG} \cap \text{TP}$ and $\textsc{Opt}^{\text{TP}} = \textsc{Opt} \cap \hat{I} = \textsc{Opt} \cap \text{TP}$

Lemma 1. *Each interval $i \in \textsc{Opt}^{\text{TP}}$ 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. This contradicts the construction of I^*, since i would have been in I^* instead of r. □

We define a set O^{FN} consisting of a copy of each interval in OPT^{FN} and let $\mathcal{F} = O^{\text{FN}} \cup U$. We define a mapping $f\colon \text{OPT} \to \text{TG} \cup \mathcal{F}$ as follows. For each $i \in \text{OPT}$:

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 $r \in U \cup \text{TG}^{\text{TP}}$, then $f(i) = r$.
   (b) Otherwise, r has been replaced by some interval t. 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, $f(i) = o_i$.

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 $F \cap O^{\text{FN}}$.

Lemma 2. *The mapping f is an injection.*

Proof. Intervals in $U \cup \text{TG}^{\text{TP}}$ are only mapped to in step 1a. Note that U and TG are disjoint. If an interval $i \in \text{OPT}$ is mapped to an interval $r \in U \cup \text{TG}$ in this step, 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, 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 $F \cap O^{\text{FN}}$ 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, $F \cap O^{\text{FN}}$ is feasible since the intervals of $F \cap O^{\text{FN}}$ are identical to the corresponding subsets of OPT. Thus, we need to show that no interval in $F \cap U$ overlaps any interval in $F \cap O^{\text{FN}}$.

Consider an interval $u \in F \cap U$ mapped to from an interval $i \in \text{OPT}$. Since i is not mapped to its own copy in $\mathcal{F}$, its copy does not belong to F. Since $i \in \text{OPT}$, no interval in $F \cap O^{\text{FN}}$ overlaps i. Thus, we need to argue that $F \cap O^{\text{FN}}$ 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. □

The following theorem follows from Lemmas 2 and 3.

Theorem 5. *For any prediction $\hat{I}$ and any input sequence I, we have*

$$\textsc{TrustGreedy}(\hat{I}, I) \geq (1 - \gamma(\hat{I}, I))\,\textsc{Opt}(I)\,.$$

Proof. We show that

$$\mathrm{TG}(\hat{I}, I) \geq \textsc{Opt}(I) - \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN}) = (1 - \gamma(\hat{I}, I))\,\textsc{Opt}(I)\,:$$

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

□

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

Proposition 2. *If a deterministic algorithm has non-zero consistency, α, it has robustness $\beta \leq \frac{1}{m}$.*

Proof. Consider a prediction that indicates the input to be one long interval, $\hat{I} = (0, m)$. In order to have non-zero consistency, α, the algorithm must accept this interval, if it is first in some sequence because it might be the only interval in that sequence.

Suppose an input σ is $(0, m), (0, 1), (1, 2), (2, 3), \ldots, (m-1, m)$. Clearly, Opt accepts the m intervals of length 1, giving robustness $\frac{1}{m}$. □

The more interesting case is randomized algorithms. The proof of the following was inspired by the proof of Theorem 13.8 in [16] for the online case without predictions, and that $\Omega(\log m)$ result was originally proven in [7].

Theorem 6. *If a (possibly randomized) algorithm* Alg *is both α-consistent and β-robust, then $\alpha \leq 1 - \frac{\lfloor \log m \rfloor - 1}{2}\beta$ and $\beta \leq \frac{2}{\lfloor \log m \rfloor - 1} \cdot (1 - \alpha)$.*

Proof. Let $r = \lfloor \log m \rfloor - 1$ and let $m' = 2^{r+1}$. Consider a prediction $\sigma = \langle \hat{I}_0, \hat{I}_1, \ldots, \hat{I}_r, \hat{I}' \rangle$, where $\hat{I}' = \langle (0,1), (1,2), \ldots, (m'-1, m') \rangle$ and, for $0 \leq i \leq r$, $\hat{I}_i = \langle (0, m'/2^i), (m'/2^i, 2m'/2^i), \ldots, (m' - m'/2^i, m') \rangle$. Note that $\hat{I}_i$ consists of 2^i disjoint intervals of length $m'/2^i$. For $0 \leq i \leq r$, let $\sigma_i = \langle \hat{I}_0, \hat{I}_1, \ldots, \hat{I}_i \rangle$.

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. Since $\textsc{Opt}(\sigma_0) = 1$ and $\textsc{Alg}$ is β-robust, $E[\textsc{Alg}(\sigma_0)] \geq \beta$. For σ_i with $i \geq 1$, $\textsc{Opt}$ accepts all intervals in $\hat{I}_i$, so $\textsc{Opt}(\sigma_i) = 2^i$. To be β-robust, the expected number of intervals of length at most $m'/2^i$ that $\textsc{Alg}$ accepts is at least $2^i\beta$. Inductively, for $i \geq 1$, by the linearity of expectations, this is at least $2^{i-1}\beta$ intervals of length $m'/2^i$, and these intervals have a total expected size of at least $2^{i-1}\beta \times m'/2^i = \frac{m'}{2}\beta$. Again, by the linearity of expectations, for σ_r, the expected sum of the lengths of the accepted intervals is at least $\sum_{i=0}^{r} \frac{m'}{2}\beta = \frac{m'(r+1)}{2}\beta$.

From σ_r, the expected number of intervals $\textsc{Alg}$ must have accepted is at least $2^r\beta$. If σ is the actual input sequence, then the predictions are correct, so for $\textsc{Alg}$ to be α-consistent, we must have $E[\textsc{Alg}(\sigma')] \geq m'\alpha$. Since also $2^r\beta + (m' - \frac{m'(r+1)}{2}\beta) \geq E[\textsc{Alg}(\sigma')]$, we can combine these two inequalities and obtain $\frac{2^r}{m'}\beta + 1 - \frac{r+1}{2}\beta \geq \alpha$. Since $\frac{2^r}{m'} = \frac{1}{2}$, this reduces to $\alpha \leq 1 - \frac{r}{2}\beta$. Solving for β, $\beta \leq \frac{2}{r}(1-\alpha)$. □

Note that as α approaches 1 (optimal consistency), β goes to 0 (worst-case robustness) and vice-versa. Next, we present a family of algorithms, RobustTrust, which has a parameter $0 \leq \alpha \leq 1$ and works as follows. With a probability of α, RobustTrust applies TG. (Applying Trust, instead of TG, 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 [16]. Crs is strictly $\lceil \log m \rceil$-competitive (they use ratios at least one). A similar algorithm was originally proven $O(\log m)$-competitive in [7].

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

We define a set of $\lceil \log m \rceil$ 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. The set $E_1 = \{e_1\}$ and Level 1 consists 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^{j-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 j, each with probability $\frac{1}{j}$. 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.

When RobustTrust applies TG and the predictions are correct, it accepts exactly as many intervals as there are in the optimal solution. From these observations, we can get the following results.

Theorem 7. RobustTrust *(*Rt*) with parameter α has consistency at least α and robustness at least $\frac{1-\alpha}{\lceil \log m \rceil}$.*

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

Suppose all predictions are correct. RobustTrust applies TG with probability α. Since TG is optimal when all predictions are correct, the expected payoff of RobustTrust is at least $\alpha \cdot \textsc{Opt}$. 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, Crs accepts as many intervals as $\textsc{Opt}$ 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 from OPT's configuration that CRS accepts on any given level is $\frac{1}{\lceil \log m \rceil}$ 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 most $\frac{1-\alpha}{\lceil \log m \rceil}$. □

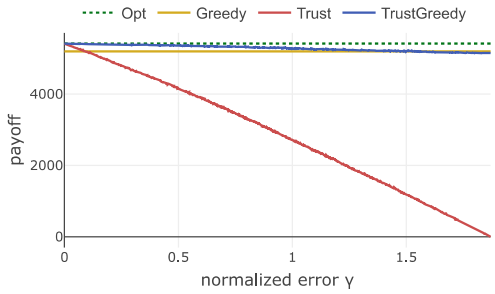
6 Experimental Results

We present an experimental evaluation of TRUST and TRUSTGREEDY in comparison with the GREEDY algorithm, which serves as a baseline online algorithm, and OPT, which serves as the performance upper bound. We evaluate our algorithms using real-world scheduling data for parallel machines [19]. Each benchmark from [19] 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 [19]. 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 remaining $N - n$ 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 $\gamma(\hat{I}, I) = \frac{\mathrm{OPT}(\mathrm{FN} \cup \mathrm{FP})}{\mathrm{OPT}(I)}$, and report the payoff of TRUST and TRUSTGREEDY as a function of γ.

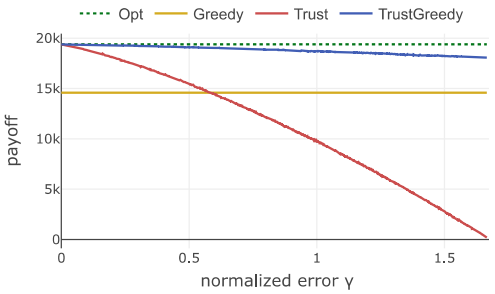
Figure 2 shows the results for two representative benchmarks from [19], namely, LLNL (the workload of the BlueGene/L system installed at Lawrence Livermore National Lab), SDSC (the workload log from San Diego Supercomputer Center), NASA-iPSC (scheduling log from Numerical Aerodynamic Simulation -NAS- 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. 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 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 payoff 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 $\gamma = \frac{\mathrm{OPT}(\mathrm{FP} \cup \mathrm{FN})}{\mathrm{OPT}(I)}$ takes values in $[1.5, 2]$.

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

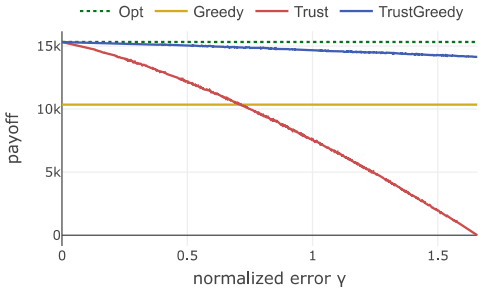
Table 1: Details on the benchmarks from [19] used in our experiments.



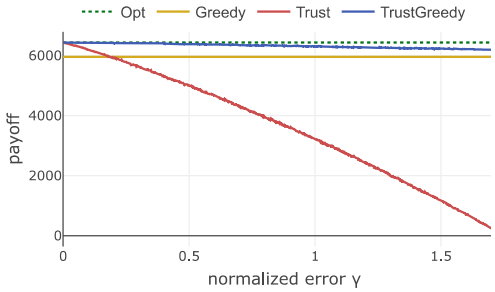
(a) LLNL-uBGL-2006-2



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

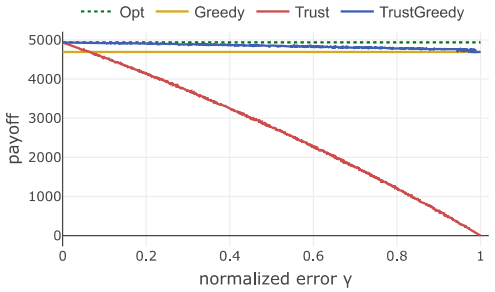


(c) CTC-SP2-1996-3.1-cln



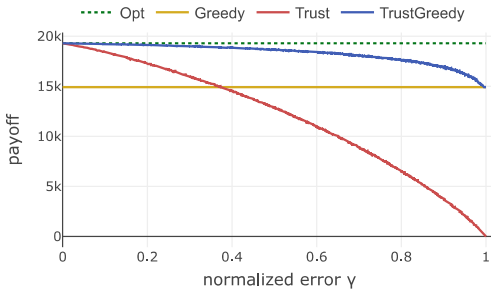
(d) NASA-iPSC-1993

Fig. 2: Payoff as a function of normalized error value

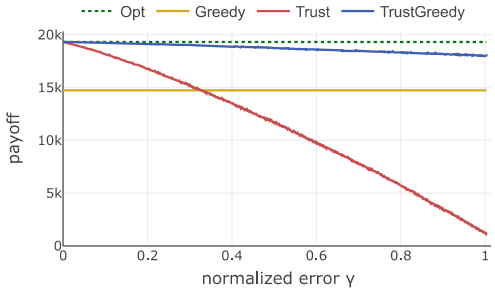


(a) LLNL-uBGL-2006-2, no FP

(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

Fig. 3: Payoff as a function of normalized error value in the absence of false positives (a), (c) and false negatives (b), (d).

We also experiment in a setting where false positives and negatives contribute differently to the error set. We generate the input sequences in the 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 3a and 3c illustrate the payoff 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 3a and 3c illustrate the payoff of Trust and TrustGreedy in this case. In this case, the payoff 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.

7 Related Problems: Matching and Independent Set

In [27], 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 selection 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 $\gamma(\hat{E}, E) = \frac{\textsc{Opt}(\text{FP} \cup \text{FN})}{\textsc{Opt}(E)}$, where $\textsc{Opt}(S)$ indicates the size of the matching for graph $G = (V, S)$.

The correspondence between the two problems is as follows: Consider a set of intervals on a star. Each such interval is a pair of vertices. We can assume no pair contains the star's center since all such intervals 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 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, we get the following graph for matching: $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and

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

See also Figure 4. Note that the edges in this graph correspond to the intervals that are used in the proof of Theorem 2. The proof can be simulated in this new setting so that the number of intervals 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.

Proposition 3. *For any instance $G = (V, E)$ of the online matching problem under the edge-arrival model and a prediction set $\hat{E}$, there is an algorithm* Trust *that matches at least* $(1-2\gamma(\hat{E}, E))\,\textsc{Opt}(G)$

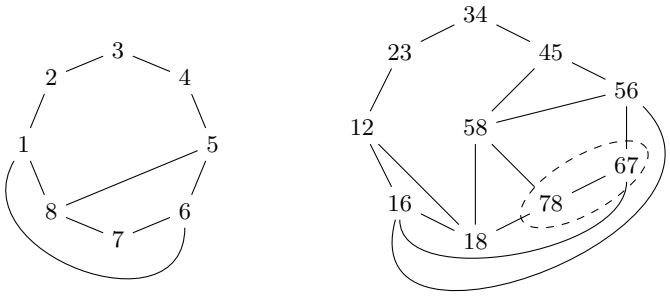


Fig. 4: Left: The graph, G, giving a correspondence between disjoint paths on a star and matchings in general graphs. Right: The line graph, G', corresponding to G, where the two digits in a vertex name in G' indicate the edge (given by its two endpoints) from G that the vertex corresponds to; the dashed ellipse indicates the contraction of the two vertices.

edges. Moreover, there are instances $G_w = (V, E_w)$ *of the matching problem, along with predictions* $\hat{E}_w$ *for which any deterministic algorithm matches at most* $(1 - 2\gamma(\hat{E}, E)_w)\,\textsc{Opt}(G_w)$ *edges.*

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. 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

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

Intervals from the proof in Theorem 2 correspond to vertices here. See also Figure 4.

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 [20,29]. Also, it is not chordal. However, the lower bound from Theorem 3, that for any deterministic algorithm Alg, there are instances I and predictions $\hat{I}$ such that $\textsc{Alg}(\hat{I}, I) = \textsc{Opt}(I) - \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$ clearly holds for independent sets in interval graphs, too, by considering the interval graph corresponding to a set of intervals on the line.

Using the correspondence between matchings in a graph, G, and the independent set in the line graph of G, we can get a similar result for the independent set under the vertex-arrival model.

References

1. Algorithms with predictions. https://algorithms-with-predictions.github.io/, accessed: 2023-02-19
2. Angelopoulos, S., Dürr, C., Jin, S., Kamali, S., Renault, M.: Online computation with untrusted advice. In: Proc. ITCS. pp. 52:1–52:15 (2020)
3. Angelopoulos, S., Kamali, S., Shadkami, K.: Online bin packing with predictions. In: Proc. IJCAI. pp. 4574–4580 (2022)
4. Angelopoulos, S., Kamali, S.: Contract scheduling with predictions. In: 35th AAAI Conference on Artificial Intelligence (AAAI), 33rd Conference on Innovative Applications of Artificial Intelligence (IAAI), 11th Symposium on Educational Advances in Artificial Intelligence (EAAI). pp. 11726–11733. AAAI Press (2021)

5. Angelopoulos, S., Arsénio, D., Kamali, S.: Competitive sequencing with noisy advice. CoRR **abs/2111.05281** (2021)
6. Antoniadis, A., Gouleakis, T., Kleer, P., Kolev, P.: Secretary and online matching problems with machine learned advice. In: Proc. NeurIPS (2020)
7. Awerbuch, B., Bartal, Y., Fiat, A., Rosén, A.: Competitive non-preemptive call control. In: Proc. (SODA). pp. 312–320 (1994)
8. Azar, Y., Leonardi, S., Touitou, N.: Flow time scheduling with uncertain processing time. In: Proc. STOC (2021)
9. Azar, Y., Panigrahi, D., Touitou, N.: Online graph algorithms with predictions. In: Proc. SODA. pp. 35–66 (2022)
10. Balkanski, E., Gkatzelis, V., Tan, X.: Strategyproof scheduling with predictions. In: Proc. ITCS. vol. 251, pp. 11:1–11:22 (2023)
11. Bampis, E., Dogeas, K., Kononov, A.V., Lucarelli, G., Pascual, F.: Scheduling with untrusted predictions. In: Proc. IJCAI. pp. 4581–4587 (2022)
12. Banerjee, S., Cohen-Addad, V., A., Li, Z.: Graph searching with predictions. In: Proc. ITCS. vol. 251, pp. 12:1–12:24 (2023)
13. Barhum, K., Böckenhauer, H., Forisek, M., Gebauer, H., Hromkovič, J., Krug, S., Smula, J., Steffen, B.: On the power of advice and randomization for the disjoint path allocation problem. In: Proc. SOFSEM. pp. 89–101 (2014)
14. Böckenhauer, H., Benz, N.C., Komm, D.: Call admission problems on trees. Theor. Comput. Sci. **922**, 410–423 (2022)
15. Böckenhauer, H., Komm, D., Wegner, R.: Call admission problems on grids with advice. Theor. Comput. Sci. **918**, 77–93 (2022)
16. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press (1998)
17. Boyar, J., Favrholdt, L.M., Larsen, K.S.: Online unit profit knapsack with untrusted predictions. In: Proc. SWAT. pp. 20:1–20:17 (2022)
18. Boyar, J., Favrholdt, L.M., Kudahl, C., Mikkelsen, J.W.: The advice complexity of a class of hard online problems. Theory Comput. Syst. **61**(4), 1128–1177 (2017)
19. Chapin, S.J., Cirne, W., Feitelson, D.G., Jones, J.P., Leutenegger, S.T., Schwiegelshohn, U., Smith, W., Talby, D.: Benchmarks and standards for the evaluation of parallel job schedulers (https://www.cs.huji.ac.il/labs/parallel/workload/). In: Proc. IPPS/SPDP. vol. 1659, pp. 67–90 (1999)
20. Chartrand, G., Harary, F.: Planar permutation graphs. Annales de l'Institut Henri Poincaré, série B – Probabil ités et Statistiques **3**(4), 433–438 (1967)
21. Chen, J.Y., Eden, T., Indyk, P., Lin, H., Narayanan, S., Rubinfeld, R., Silwal, S., Wagner, T., Woodruff, D.P., Zhang, M.: Triangle and four cycle counting with predictions in graph streams. In: Proc. ICLR (2022)
22. Chen, J.Y., Silwal, S., Vakilian, A., Zhang, F.: Faster fundamental graph algorithms via learned predictions. In: Proc. ICML. vol. 162, pp. 3583–3602 (2022)
23. D. Wagner, K.W.: A linear-time algorithm for edge-disjoint paths in planar graphs. Combinatorica **15**, 135–150 (1995)
24. Eberle, F., Lindermayr, A., Megow, N., Nölke, L., Schlöter, J.: Robustification of online graph exploration methods. In: Proc. AAAI. pp. 9732–9740 (2022)
25. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. **5**(4), 691–703 (1976)
26. Frank, A.: Edge-disjoint paths in planar graphs. J. Combin. Theory Ser. B **39**, 164–178 (1985)
27. Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica pp. 3–20 (1977)
28. Gebauer, H., Komm, D., Královic, R., Královic, R., Smula, J.: Disjoint path allocation with sublinear advice. In: Proc. COCOON. vol. 9198, pp. 417–429 (2015)
29. Harary, F.: Graph Theory. Addison-Wesley, Reading, Massachusetts (1969)
30. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Non-clairvoyant scheduling with predictions. In: Agrawal, K., Azar, Y. (eds.) Proc. SPAA. pp. 285–294. ACM (2021)
31. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Online knapsack with frequency predictions. In: Proc. NuerIPS. pp. 2733–2743 (2021)
32. K. Matsumoto, T. Nishizeki, N.S.: An efficient algorithm for finding multi-commodity flows in planar networks. SIAM J. Comput. **14**(2), 289–302 (1985)

33. Kolen, A.W., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C.: Interval scheduling: A survey. Naval Research Logistics (NRL) **54**(5), 530–543 (2007)
34. Lattanzi, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Online scheduling via learned weights. In: Proc. SODA. pp. 1859–1877 (2020)
35. Lavastida, T., Moseley, B., Ravi, R., Xu, C.: Learnable and instance-robust predictions for online matching, flows and load balancing. In: Proc. ESA. vol. 204, pp. 59:1–59:17 (2021)
36. Lavastida, T., Moseley, B., Ravi, R., Xu, C.: Using predicted weights for ad delivery. In: Proc. ACDA. pp. 21–31 (2021)
37. Lee, R., Maghakian, J., Hajiesmaili, M., Li, J., Sitaraman, R.K., Liu, Z.: Online peak-aware energy scheduling with untrusted advice. In: Proc. e-Energy. pp. 107–123 (2021)
38. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. In: Proc. ICML. pp. 3302–3311 (2018)
39. Mitzenmacher, M., Vassilvitskii, S.: Algorithms with predictions. In: Roughgarden, T. (ed.) Beyond the Worst-Case Analysis of Algorithms, pp. 646–662. Cambridge University Press (2020)
40. Nishizeki, T., Vygen, J., Zhou, X.: The edge-disjoint paths problem is NP-complete for series–parallel graphs. Discrete Applied Mathematics **115**(1-3), 177–186 (2001)
41. Purohit, M., Svitkina, Z., Kumar, R.: Improving online algorithms via ML predictions. In: Proc. NeurIPS. pp. 9661–9670 (2018)
42. Rohatgi, D.: Near-optimal bounds for online caching with machine learned advice. In: Proc. SODA. pp. 1834–1845 (2020)
43. Wei, A., Zhang, F.: Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In: Proc. NeurIPS (2020)
44. Zeynali, A., Sun, B., Hajiesmaili, M., Wierman, A.: Data-driven competitive algorithms for online knapsack and set cover. In: Proc. AAAI. pp. 10833–10841 (2021)