Online Interval Scheduling with Predictions *

Joan Boyar[1 [0000−0002−0725−8341]], Lene M. Favrholdt[1 [0000−0003−3054−2997]], Shahin Kamali[2 [0000−0003−1404−2212]], and Kim S. Larsen[1 [0000−0003−0560−3794]]

[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

* 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).

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 *profit* of the scheduler. This problem is also known as *fixed job scheduling* and *k-track assignment* [35].

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 [30] and outerplanar graphs by a combination of [26,34,29], but the problem is NP-complete for general graphs [28], and even on quite restricted graphs such as series-parallel graphs [42]. 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.

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. 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, $\textsc{Opt}$. The *competitive ratio* of an online algorithm $\textsc{Alg}$ is defined as $\inf_I \{\textsc{Alg}(I)/\,\textsc{Opt}(I)\}$, where $\textsc{Alg}(I)$ and $\textsc{Opt}(I)$, respectively, denote the profit of $\textsc{Alg}$ and $\textsc{Opt}$ on I (for randomized algorithms, $\textsc{Alg}(I)$ is the expected profit of $\textsc{Alg}$). 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 $\frac{1}{m}$ and $\frac{1}{\lceil \log m \rceil}$ [20]. These results suggest that the constraints on online algorithms must be relaxed to compete with $\textsc{Opt}$. Specifically, the problem has been considered in the *advice complexity model* for path graphs [16,31], trees [18], and grid graphs [19]. 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 [43,45], makespan scheduling [36], contract scheduling [6,7], and other variants of scheduling problems [11,39,14,13].

Other online problems studied under the prediction model include bin packing [3,4], knapsack [47,33,21], caching [40,44,46,8], matching problems [9,37,38], time series search [5], and various graph problems [25,27,24,12,15]. See also the survey by Mitzenmacher and Vassilvitskii [41] 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 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, $\eta(\hat{I}, I) = \textsc{Opt}(\mathrm{FP} \cup \mathrm{FN})$. 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 profit of an algorithm $\textsc{Alg}$ 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 $\textsc{Trust}$, which accepts a request only if it belongs to the set of predictions 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 $\textsc{Alg}$ and any positive integer p, there are worst-case input sequence I_w and prediction set $\hat{I}_w$ over a star graph, S_{8p}, 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, $\textsc{Trust}$ achieves an optimal competitive ratio in any graph class that contains S_8.

The above result demonstrates that even for trees, the problem is so hard that no algorithm can do better than the trivial $\textsc{Trust}$. 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 $\textsc{Alg}$ 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 $\textsc{Alg}(\hat{I}_w, I_w) \leq (1-\gamma(\hat{I}_w, I_w))\,\textsc{Opt}(I_w)$ (Theorem 3, setting $c = 2$).

  Next, we present a negative result for $\textsc{Trust}$. 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 $\textsc{Trust}(\hat{I}_w, I_w) = (1-2\gamma(\hat{I}_w, I_w))\,\textsc{Opt}(I_w)$. (Theorem 4). This suggests that there is room for improvement over $\textsc{Trust}$.

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 profit of the maintained solution. For any input I and prediction $\hat{I}$, we show that $\textsc{TrustGreedy}(\hat{I}, I) \geq (1 - \gamma(\hat{I}, I))\,\textsc{Opt}(I)$ (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) [40]. 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 $\frac{1}{10 \lceil \log m \rceil}$, 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 [23]. 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.

Omitted details and all omitted proofs can be found in the full paper [17].

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. Note that it is not interesting to consider alternative types of predictions that are very compact. 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, even if all predictions are correct [22].

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 $\eta = \textsc{Opt}(\text{FP} \cup \text{FN})$, and hence, the normalized error measure is $\gamma = \textsc{Opt}(\text{FP} \cup \text{FN}) / \textsc{Opt}(I)$. Our error measure satisfies the following desirable properties, the first two of which were strongly recommended in Im, et al. [32]: First, the *monotonicity* property requires that increasing the number of true positives or true negatives does not increase the error. Second, the *Lipschitz* property ensures that $\eta(\hat{I}, I) \geq |\textsc{Opt}(I) - \textsc{Opt}(\hat{I})|$. Finally, *Lipschitz completeness* requires $\eta(I, \hat{I}) \leq \textsc{Opt}(\text{FP} \cup \text{FN})$. The Lipschitz and Lipschitz completeness properties enforce a range for the error to avoid situations where the error is too small or too large. We refer to the full paper [17] for details on these properties. There, we also 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.

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

3 Disjoint-path allocation

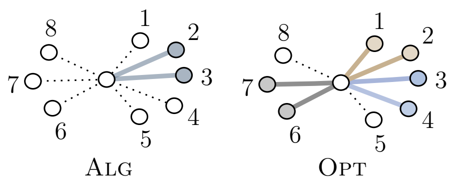
In this section, we show that a simple algorithm Trust for the disjoint path allocation problem has an optimal competitive ratio for any graph of maximal degree at least 8. Trust simply relies on the predictions being correct. Specifically, it computes an optimal solution I^* in $\hat{I}$ before processing the first request. Then, it accepts any 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 (i) false negatives cause a deficit of at most $\textsc{Opt}(\text{FN})$ in the schedule of Trust compared to the optimal schedule for I^*, (ii) false positives cause a deficit of at most $\textsc{Opt}(\text{FP})$ in the optimal schedule of I^*, compared to the optimal schedule for I, and (iii) $\textsc{Opt}(\text{FP}) + \textsc{Opt}(\text{FN}) \leq 2\,\textsc{Opt}(\text{FP} \cup \text{FN}) = 2\eta$.

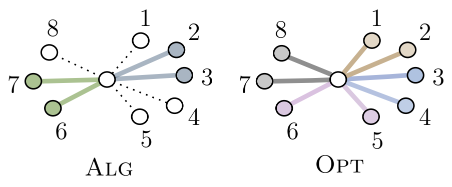
Theorem 1. *For any graph G, any prediction $\hat{I}$, and input sequence I, we have* $\textsc{Trust}(\hat{I}, I) \geq (1 - 2\gamma(\hat{I}, I))\,\textsc{Opt}(I)$.

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 class that contains stars of degree 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 requests $\hat{I}_w$ and a request sequence I_w such that* $\eta(\hat{I}_w, I_w) = p$ *and* $\textsc{Alg}(\hat{I}_w, I_w) \leq (1 - 2\gamma(\hat{I}, I))\,\textsc{Opt}(I_w)$.



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



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

Fig. 1: Illustration of the proof of Theorem 2 for the case that ALG accepts $(2, 3)$. Highlighted edges indicate paths between accepted pairs.

Proof (sketch). 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 profit 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 algorithm's profits are summed over all groups. Hence, the total error will be equal to $\eta(\hat{I}_w, I_w) = p$, and we can write $\text{ALG}(I_w) \leq \text{OPT}(I_w) - 2\eta(\hat{I}_w, I_w)$, that is, $\text{ALG}(I_w) \leq (1 - 2\gamma(\hat{I}_w, I_w))\, \text{OPT}(I_w)$.

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 $(2, 3)$. Here we discuss the case when ALG accepts $(2, 3)$; the other case when it rejects $(2, 3)$ follows with a similar case analysis (see the full paper [17]).

Case ALG accepts $(2, 3)$: The next request to arrive is $(6, 7)$. If ALG rejects this request, the next to arrive is $(7, 8)$. If ALG also rejects this request, then the requests $(1, 2)$ 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 $\text{OPT}(\text{FN} \cup \text{FP}) = 1$. 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 requests $(1, 2)$ and $(3, 4)$ also arrive, along with a false negative $(5, x)$. The request $(4, 5)$ 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. □

4 Interval Scheduling

In this section, we show tight upper and lower bounds for 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 $\textsc{Alg}$, there exists an input sequence I_w and a set of predictions $\hat{I}_w$ such that $\textsc{Alg}(\hat{I}_w, I_w) = \textsc{Opt}(I_w) - \eta(\hat{I}_w, I_w)$, and that this can be established for any integer error of at least 2..

Theorem 3. *Let* $\textsc{Alg}$ *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* $\textsc{Alg}(\hat{I}_w, I_w) = (1 - \gamma(\hat{I}_w, I_w)) \, \textsc{Opt}(I_w) \leq \frac{1}{c} \, \textsc{Opt}(I_w)\,.$

Proof. $\textsc{Alg}$ 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: $\hat{I} = \bigcup_{i=0}^{p-1} \left\{(ci, c(i+1)), (ci, ci+1)\right\}.$

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

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

For $c = 2$, we get $\eta(\hat{I}_w, I) = p$ and $\textsc{Alg}(\hat{I}_w, I_w) = (1 - \gamma(\hat{I}_w, I_w)) \, \textsc{Opt}(\hat{I}_w)$. The next theorem shows that the competitive ratio of $\textsc{Trust}$ 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 profit of $\textsc{Opt}$ and η grow in phases while the profit of $\textsc{Trust}$ 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* $\textsc{Trust}(\hat{I}_w, I_w) = (1-2\gamma(\hat{I}_w, I_w))\,\textsc{Opt}(I_w)$.

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, 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. TG 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. During the online processing after this initialization, 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 $I^* \cap \mathrm{FP}$ 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:

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

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. 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 $\textsc{Opt}^{\mathrm{FN}}$ 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 $r \in U \cup \text{TG}^{\text{TP}}$, 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 $\text{O}\textsc{pt}^{\text{FN}}$.
   (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 $i \in \text{O}\textsc{pt}$ 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 $\text{O}\textsc{pt}$ 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 $\text{O}\textsc{pt}$, 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 $\text{O}\textsc{pt}$ 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. $\square$

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 $\textsc{Opt}$. Thus, we 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, we need 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. □

The following theorem follows from Lemmas 2 and 3.

Theorem 5. *For any prediction $\hat{I}$ and any input sequence I, we have* $\textsc{TrustGreedy}(\hat{I}, I) \geq (1 - \gamma(\hat{I}, I))\, \textsc{Opt}(I)$.

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 $\beta \leq \frac{1}{m}$.*

The more interesting case is randomized algorithms. The proof of the following was inspired by the proof of Theorem 13.8 in [20] for the online case without predictions, and that $\Omega(\log m)$ result was originally proven in [10].

Theorem 6. *If a (possibly randomized) algorithm* $\textsc{Alg}$ *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 $\text{OPT}(\sigma_0) = 1$ and ALG is β-robust, $E[\text{ALG}(\sigma_0)] \geq \beta$. For σ_i with $i \geq 1$, OPT accepts all intervals in $\hat{I}_i$, so $\text{OPT}(\sigma_i) = 2^i$. To be β-robust, the expected number of intervals of length at most $m'/2^i$ that 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 ALG must have accepted is at least $2^r\beta$. If σ is the actual input sequence, then the predictions are correct, so for ALG to be α-consistent, we must have $E[\text{ALG}(\sigma')] \geq m'\alpha$. Since also $2^r\beta + (m' - \frac{m'(r+1)}{2}\beta) \geq E[\text{ALG}(\sigma')]$, 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, $\beta \leq \frac{2}{r}(1-\alpha)$. □

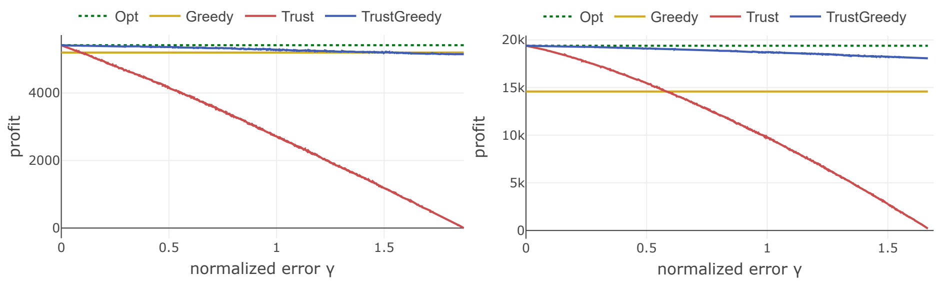
Note that as α approaches 1 (optimal consistency), β goes to 0 (worst-case robustness) and vice-versa. Next, we present a family of algorithms, ROBUSTTRUST, which has a parameter $0 \leq \alpha \leq 1$ and works as follows. With a probability of α, ROBUSTTRUST applies TG. (Applying TRUST, instead of 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 [20]. For completeness, we include the CRS algorithm in the full paper [17]. 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 [10].

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 $\frac{1-\alpha}{\lceil \log m \rceil}$.*

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 [23]. Each benchmark from [23] specifies the start and finish times of tasks as scheduled on parallel machines with several processors (see also the full paper [17]). We use these tasks to generate inputs to the interval scheduling problem. For each



(a) LLNL-uBGL-2006-2 (b) SDSC-DS-2004-2.1-cln

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

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 $\gamma(\hat{I}, I) = \frac{\textsc{Opt}(\mathrm{FN} \cup \mathrm{FP})}{\textsc{Opt}(I)}$, and report the profit of Trust and TrustGreedy as a function of γ.

Figure 2 shows the results for two representative benchmarks from [23], namely, LLNL (the workload of the BlueGene/L system installed at Lawrence Livermore National Lab) and SDSC (the workload log from San Diego Supercomputer Center). 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 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.

We present results for more benchmarks and situations where false negatives and false positives contribute differently to the error set in the full paper [17]. Our code and results are available at [2].

7 Concluding Remarks

In [30], 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 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 $(1 - 2\gamma(\hat{E}, E))\,\textsc{Opt}(G)$ 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 the independent set in the line graph of G, we can get a similar result for the independent set under the vertex-arrival model. We refer to the full paper [17] for details.

References

1. Algorithms with predictions. `https://algorithms-with-predictions.github.io/`, accessed: 2023-02-19
2. Interval scheduling with prediction. `https://github.com/shahink84/IntervalSchedulingWithPrediction`, accessed: 2023-02-19
3. Angelopoulos, S., Dürr, C., Jin, S., Kamali, S., Renault, M.: Online computation with untrusted advice. In: Proc. ITCS. LIPIcs, vol. 151, pp. 52:1–52:15 (2020)
4. Angelopoulos, S., Kamali, S., Shadkami, K.: Online bin packing with predictions. In: Proc. IJCAI. pp. 4574–4580 (2022)
5. Angelopoulos, S., Kamali, S., Zhang, D.: Online search with best-price and query-based predictions. In: Proc. AAAI. pp. 9652–9660 (2023)
6. Angelopoulos, S., Kamali, S.: Contract scheduling with predictions. In: Proc. AAAI. pp. 11726–11733. AAAI Press (2021)
7. Angelopoulos, S., Arsénio, D., Kamali, S.: Competitive sequencing with noisy advice. CoRR **abs/2111.05281** (2021)
8. Antoniadis, A., Boyar, J., Eliás, M., Favrholdt, L.M., Hoeksma, R., Larsen, K.S., Polak, A., Simon, B.: Paging with succinct predictions. In: Proc. ICML (2023), to appear.
9. Antoniadis, A., Gouleakis, T., Kleer, P., Kolev, P.: Secretary and online matching problems with machine learned advice. In: Proc. NeurIPS (2020)
10. Awerbuch, B., Bartal, Y., Fiat, A., Rosén, A.: Competitive non-preemptive call control. In: Proc. SODA. pp. 312–320 (1994)

11. Azar, Y., Leonardi, S., Touitou, N.: Flow time scheduling with uncertain processing time. In: Proc. STOC. pp. 1070–1080 (2021)
12. Azar, Y., Panigrahi, D., Touitou, N.: Online graph algorithms with predictions. In: Proc. SODA. pp. 35–66 (2022)
13. Balkanski, E., Gkatzelis, V., Tan, X.: Strategyproof scheduling with predictions. In: Proc. ITCS. LIPIcs, vol. 251, pp. 11:1–11:22 (2023)
14. Bampis, E., Dogeas, K., Kononov, A.V., Lucarelli, G., Pascual, F.: Scheduling with untrusted predictions. In: Proc. IJCAI. pp. 4581–4587 (2022)
15. Banerjee, S., Cohen-Addad, V., A., Li, Z.: Graph searching with predictions. In: Proc. ITCS. LIPIcs, vol. 251, pp. 12:1–12:24 (2023)
16. 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. LNCS, vol. 8327, pp. 89–101 (2014)
17. Berg, M., Boyar, J., Favrholdt, L.M., Larsen, K.S.: Online interval scheduling with predictions. ArXiv (2023), http://arxiv.org/abs/2302.13701, arXiv:2302.13701. To appear in 18th WADS, 2023.
18. Böckenhauer, H., Benz, N.C., Komm, D.: Call admission problems on trees. Theor. Comput. Sci. **922**, 410–423 (2022)
19. Böckenhauer, H., Komm, D., Wegner, R.: Call admission problems on grids with advice. Theor. Comput. Sci. **918**, 77–93 (2022)
20. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press (1998)
21. Boyar, J., Favrholdt, L.M., Larsen, K.S.: Online unit profit knapsack with untrusted predictions. In: Proc. SWAT. LIPIcs, vol. 227, pp. 20:1–20:17 (2022)
22. 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)
23. 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. In: Proc. IPPS/SPDP. LNCS, vol. 1659, pp. 67–90 (1999)
24. 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)
25. Chen, J.Y., Silwal, S., Vakilian, A., Zhang, F.: Faster fundamental graph algorithms via learned predictions. In: Proc. ICML. PLMR, vol. 162, pp. 3583–3602 (2022)
26. D. Wagner, K.W.: A linear-time algorithm for edge-disjoint paths in planar graphs. Combinatorica **15**, 135–150 (1995)
27. Eberle, F., Lindermayr, A., Megow, N., Nölke, L., Schlöter, J.: Robustification of online graph exploration methods. In: Proc. AAAI. pp. 9732–9740 (2022)
28. Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. **5**(4), 691–703 (1976)
29. Frank, A.: Edge-disjoint paths in planar graphs. J. Combin. Theory Ser. B **39**, 164–178 (1985)
30. Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica **18**, 3–20 (1977)
31. Gebauer, H., Komm, D., Královic, R., Královic, R., Smula, J.: Disjoint path allocation with sublinear advice. In: Proc. COCOON. LNCS, vol. 9198, pp. 417–429 (2015)
32. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Non-clairvoyant scheduling with predictions. In: Proc. SPAA. pp. 285–294 (2021)

33. Im, S., Kumar, R., Qaem, M.M., Purohit, M.: Online knapsack with frequency predictions. In: Proc. NeurIPS. pp. 2733–2743 (2021)
34. 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)
35. Kolen, A.W., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C.: Interval scheduling: A survey. Naval Research Logistics **54**(5), 530–543 (2007)
36. Lattanzi, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Online scheduling via learned weights. In: Proc. SODA. pp. 1859–1877 (2020)
37. Lavastida, T., Moseley, B., Ravi, R., Xu, C.: Learnable and instance-robust predictions for online matching, flows and load balancing. In: Proc. ESA. LIPIcs, vol. 204, pp. 59:1–59:17 (2021)
38. Lavastida, T., Moseley, B., Ravi, R., Xu, C.: Using predicted weights for ad delivery. In: Proc. ACDA. pp. 21–31 (2021)
39. 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)
40. Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. In: Proc. ICML. PMLR, vol. 80, pp. 3302–3311 (2018)
41. 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)
42. Nishizeki, T., Vygen, J., Zhou, X.: The edge-disjoint paths problem is NP-complete for series–parallel graphs. Discret. Appl. Math. **115**(1-3), 177–186 (2001)
43. Purohit, M., Svitkina, Z., Kumar, R.: Improving online algorithms via ML predictions. In: Proc. NeurIPS. pp. 9661–9670 (2018)
44. Rohatgi, D.: Near-optimal bounds for online caching with machine learned advice. In: Proc. SODA. pp. 1834–1845 (2020)
45. Wei, A., Zhang, F.: Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In: Proc. NeurIPS (2020)
46. Wei, A.: Better and simpler learning-augmented online caching. In: Proc. APPROX/RANDOM. LIPIcs, vol. 176, pp. 60:1–60:17 (2020)
47. 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)