Sorting and Hypergraph Orientation under Uncertainty with Predictions

Thomas Erlebach[1] , **Murilo de Lima**[2] , **Nicole Megow**[3] and **Jens Schlöter**[3]
[1]Department of Computer Science, Durham University, United Kingdom
[2]School of Informatics, University of Leicester, United Kingdom
[3]Faculty of Mathematics and Computer Science, University of Bremen, Germany
thomas.erlebach@durham.ac.uk, mslima@ic.unicamp.br, {nicole.megow, jschloet}@uni-bremen.de

Abstract

Learning-augmented algorithms have been attracting increasing interest, but have only recently been considered in the setting of explorable uncertainty where precise values of uncertain input elements can be obtained by a query and the goal is to minimize the number of queries needed to solve a problem. We study learning-augmented algorithms for sorting and hypergraph orientation under uncertainty, assuming access to untrusted predictions for the uncertain values. Our algorithms provide improved performance guarantees for accurate predictions while maintaining worst-case guarantees that are best possible without predictions. For sorting, our algorithm uses the optimal number of queries for accurate predictions and at most twice the optimal number for arbitrarily wrong predictions. For hypergraph orientation, for any $\gamma \geq 2$, we give an algorithm that uses at most $1 + 1/\gamma$ times the optimal number of queries for accurate predictions and at most γ times the optimal number for arbitrarily wrong predictions. These tradeoffs are the best possible. We also consider different error metrics and show that the performance of our algorithms degrades smoothly with the prediction error in all the cases where this is possible.

1 Introduction

The emerging research area of learning-augmented algorithm design has been attracting increasing attention in recent years. For *online* algorithms, it was initiated in [Lykouris and Vassilvtiskii, 2018] for caching and has fostered an overwhelming number of results for numerous problems, including online graph problems [Kumar *et al.*, 2019; Lindermayr *et al.*, 2022; Eberle *et al.*, 2022; Azar *et al.*, 2022b] and scheduling problems [Purohit *et al.*, 2018; Angelopoulos *et al.*, 2020; Mitzenmacher, 2020; Lattanzi *et al.*, 2020; Azar *et al.*, 2021; Azar *et al.*, 2022a; Lindermayr and Megow, 2022; Bampis *et al.*, 2022; Li and Xian, 2021]. In the research area of *explorable uncertainty* [Kahan, 1991; Erlebach and Hoffmann, 2015], however, learning-augmented algorithms have only recently been studied for the first time, namely for the minimum spanning tree problem (MST) [Erlebach *et al.*, 2022]. This area considers problems with uncertainty in the input data assuming that a *query* to an input element can be used to obtain the exact value of that element. Queries are costly, and hence the goal is to make as few queries as possible until sufficient information has been obtained to solve the given problem. Learning-augmented algorithms in this setting are given (untrusted) predictions of the uncertain input values. In this paper, we present and analyze learning-augmented algorithms for further fundamental problems with explorable uncertainty, namely sorting and hypergraph orientation.

In the *hypergraph orientation problem under uncertainty* [Bampis *et al.*, 2021], we are given a hypergraph $H = (V, E)$. Each vertex $v \in V$ is associated with an *uncertainty interval* $I_v = (L_v, U_v)$ and an, initially unknown, *precise weight* $w_v \in I_v$. We call L_v and U_v the *lower and upper limit* of v. A query of v reveals its precise weight w_v and reduces its uncertainty interval to $I_v = [w_v]$. Our task is to orient each hyperedge $S \in E$ towards the vertex of minimum precise weight in S. An adaptive algorithm can sequentially make queries to vertices to learn their weights until it has enough information to identify the minimum-weight vertex of each hyperedge. A set $Q \subseteq V$ is called *feasible* if querying Q reveals sufficient information to find the orientation. As queries are costly, the goal is to (adaptively) find a feasible query set of minimum size. An important special case is when the input graph is a simple graph that is exactly the interval graph induced by the uncertainty intervals $\mathcal{I}$. This special case corresponds to the problem of sorting a set of unknown values represented by uncertainty intervals and, therefore, we refer to it as *sorting under uncertainty*.

Since there exist input instances that are impossible to solve without querying all vertices, we evaluate our algorithms in an *instance-dependent* manner: For each input, we compare the number of queries made by an algorithm with the minimum possible number of queries *for that input*, using competitive analysis. For a given problem instance, let OPT denote an optimal query set. An algorithm is *ρ-competitive* if it executes, for any problem instance, at most $\rho \cdot |\mathrm{OPT}|$ queries. As the query results are uncertain and, to a large extent, are the deciding factor whether querying certain vertices is a good strategy or not, the problem has a clear online flavor. In particular, the uncertainty prevents 1-competitive algorithms, even without any running time restrictions.

Variants of hypergraph orientation have been widely stud-

ied since the model of explorable uncertainty has been proposed [Kahan, 1991]. Sorting and hypergraph orientation are well known to admit efficient polynomial-time algorithms if precise input data is given, and they are well understood in the setting of explorable uncertainty: The best known deterministic algorithms are 2-competitive, and no deterministic algorithm can be better [Kahan, 1991; Halldórsson and de Lima, 2021; Bampis *et al.*, 2021]. For sorting, the competitive ratio can be improved to 1.5 using randomization [Halldórsson and de Lima, 2021]. In the stochastic setting, where the precise weights of the vertices are drawn according to known distributions over the intervals, there exists a 1.618-competitive algorithm for hypergraph orientation and a $4/3$-competitive algorithm for special cases [Bampis *et al.*, 2021]. For the stochastic sorting problem, the algorithm with the optimal expected query cost is known, but its competitive ratio remains unknown [Chaplick *et al.*, 2021]. Besides hypergraph orientation and other selection-type problems, there has been work on combinatorial optimization problems with explorable uncertainty, such as shortest path [Feder *et al.*, 2007], knapsack [Goerigk *et al.*, 2015], scheduling problems [Dürr *et al.*, 2020; Arantes *et al.*, 2018; Albers and Eckl, 2020; Albers and Eckl, 2021; Gong *et al.*, 2022], the minimum spanning tree (MST) problem and matroids [Erlebach *et al.*, 2008; Erlebach and Hoffmann, 2014; Megow *et al.*, 2017; Focke *et al.*, 2020; Merino and Soto, 2019].

In this paper, we consider a third model (to the adversarial and stochastic setting) and assume that the algorithm has, for each vertex v, access to a prediction $\overline{w}_v \in I_v$ for the unknown weight w_v. These predicted weights could for example be computed using machine learning (ML) methods or other statistical tools on past data. Given the emerging success of ML methods, it seems reasonable to expect predictions of high accuracy. However, there are no theoretical guarantees and the predictions might be arbitrarily wrong, which raises the question whether an ML algorithm performs sufficiently well in all circumstances. In the context of hypergraph orientation and explorable uncertainty, we answer this question affirmatively by designing learning-augmented algorithms that perform very well if the predictions are accurate and still match the adversarial lower bounds even if the predictions are arbitrarily wrong. To formalize these properties, we use the notions of α-consistency and β-robustness [Lykouris and Vassilvtiskii, 2018; Purohit *et al.*, 2018]: An algorithm is α-consistent if it is α-competitive when the predictions are correct, and it is β-robust if it is β-competitive no matter how wrong the predictions are. Additionally, we aim at guaranteeing a smooth transition between consistency and robustness by giving performance guarantees that gracefully degrade with the *amount* of prediction error. This raises interesting questions regarding appropriate ways of measuring prediction errors, and we explore several such measures. Analyzing algorithms in terms of error-dependent consistency and robustness allows us to still give worst-case guarantees (in contrast to the stochastic setting) that are more fine-grained than guarantees in the pure adversarial setting.

Research on explorable uncertainty with untrusted predictions is in its infancy. In the only previous work that we are aware of, Erlebach *et al.* 2022 studied the MST problem with uncertainty and untrusted predictions. They propose an error measure k_h called *hop-distance* (definition follows later) and obtain a 1.5-consistent and 2-robust algorithm as well as a parameterized consistency-robustness tradeoff with smooth degradation, $\min\{1 + \frac{1}{\gamma} + \frac{5 \cdot k_h}{|\mathrm{OPT}|}, \gamma + 1\}$, for any integral $\gamma \geq 2$ and error k_h. It remained open whether or how other problems with explorable uncertainty can leverage untrusted predictions, and what other prediction models or error metrics might be useful in explorable uncertainty settings.

Main results. We show how to utilize possibly erroneous predictions for hypergraph orientation and sorting with explorable uncertainty. For sorting, we present an algorithm that is 1-consistent and 2-robust, which is in a remarkable way best possible: the algorithm identifies an optimal query set if the predictions are accurate, while maintaining the best possible worst-case ratio of 2 for arbitrary predictions. For hypergraph orientation, we give a 1.5-consistent and 2-robust algorithm and show that this consistency is best possible when aiming for optimal robustness.

Our major focus lies on a more fine-grained performance analysis with guarantees that improve with the prediction accuracy. A key ingredient in this line of research is the choice of error measure quantifying this (in)accuracy. We propose and discuss three different error measures $k_\#, k_h$ and k_M: The number of inaccurate predictions $k_\#$ is natural and allows a smooth degradation result for sorting, but in general it is too crude to allow for improving upon lower bounds of 2 for the setting without predictions. We therefore also consider the *hop distance* k_h, proposed in [Erlebach *et al.*, 2022]. Further, we introduce a new error measure k_M called *mandatory query distance* which is tailored to problems with explorable uncertainty. It is defined in a more problem-specific way, and we show it to be more restrictive in the sense that $k_M \leq k_h$.

For the sorting problem, we obtain an algorithm with competitive ratio $\min\{1 + k/|\mathrm{OPT}|, 2\}$, where k can be any of the three error measures considered, which is best possible. For the hypergraph orientation problem, we provide an algorithm with competitive ratio $\min\{(1 + \frac{1}{\gamma - 1})(1 + \frac{k_M}{|\mathrm{OPT}|}), \gamma\}$, for any integral $\gamma \geq 2$. This is best possible for $k_M = 0$ and large k_M. With respect to the hop distance, we achieve the stronger bound $\min\{(1 + \frac{1}{\gamma})(1 + \frac{k_h}{|\mathrm{OPT}|}), \gamma\}$, for any integral $\gamma \geq 2$, which is also best possible for $k_h = 0$ and large k_h. While the consistency and robustness trade-off of the k_M-dependent algorithm is weaker, we remark that the corresponding algorithm requires less predicted information than the k_h-dependent algorithm and that the error dependency can be stronger as k_M can be significantly smaller than k_h. We note that parameter γ is in both results restricted to integral values as it determines sizes of query sets, but a generalization to reals $\gamma \geq 2$ is possible at a small loss in the guarantee.

While our algorithm for sorting has polynomial running time, the algorithms for the hypergraph orientation problem may involve solving an NP-hard vertex cover problem. We justify this increased complexity by showing that even the offline version of the problem (determining the optimal query set if the precise weights are known) is NP-hard.

All missing proofs are provided in the full version [Erlebach *et al.*, 2023] of the paper.

2 Preliminaries and Error Measures

An algorithm for the hypergraph orientation problem with uncertainty that assumes that the predicted weights are accurate can exploit the characterization of optimal solutions given in [Bampis *et al.*, 2021] to compute a query set that is optimal if the predictions are indeed accurate. Querying this set leads to 1-consistency but may perform arbitrarily bad in case of incorrect predictions. On the other hand, known 2-competitive algorithms for the adversarial problems without predictions [Kahan, 1991; Halldórsson and de Lima, 2021] are not better than 2-consistent, and the algorithms for the stochastic setting [Bampis *et al.*, 2021] do not guarantee any robustness at all. The known lower bounds of 2 rule out any robustness factor less than 2 for our model. They build on the simple example with a single edge $\{u, v\}$ and intersecting intervals I_v, I_u. No matter which vertex a deterministic algorithm queries first, say v, the realized weight could be $w_v \in I_v \cap I_u$, which requires a second query. If the adversary chooses $w_u \notin I_v \cap I_u$, querying just u would have been sufficient to identify the vertex of minimum weight.

The following bound on the best achievable tradeoff between consistency and robustness translates from the lower bound in [Erlebach *et al.*, 2022] for MST under uncertainty with predictions. Later in this paper, we provide algorithms with matching performance guarantees. Note that this lower bound does not hold for the sorting problem.

Theorem 2.1. *Let $\beta \geq 2$ be a fixed integer. For hypergraph orientation under uncertainty, there is no deterministic β-robust algorithm that is α-consistent for $\alpha < 1 + \frac{1}{\beta}$. And vice versa, no deterministic α-consistent algorithm, with $\alpha > 1$, is β-robust for $\beta < \max\{\frac{1}{\alpha-1}, 2\}$. The result holds even for orienting a single hyperedge or a simple (non-hyper) graph.*

2.1 Preliminaries

The crucial structure and unifying concept in hypergraph orientation and sorting under explorable uncertainty without predictions are *witness sets* [Bruce *et al.*, 2005]. Witness sets are the key to any comparison with an optimal solution. A "classical" witness set is a set of vertices for which we can guarantee that any feasible solution must query at least *one* of these vertices. In the classical setting without access to predictions, sorting and hypergraph orientation admit 2-competitive online algorithms that rely essentially on identifying and querying disjoint witness sets of size two. We refer to witness sets of size two also as *witness pairs*. The following lemma characterizes witness pairs for our problems. We call a vertex v *leftmost* in a hyperedge S if it is a vertex with minimum lower limit L_v in S.

Lemma 2.2 (Kahan, 1991)**.** *Given (hyper)graph $H = (V, E)$. Consider some $S \in E$. A set $\{v, u\} \subseteq S$ with $I_v \cap I_u \neq \emptyset$, and v or u leftmost in S, is a witness set.*

In terms of learning-augmented algorithms, completely relying on querying witness pairs ensures 2-robustness, but it does not lead to any improvements in terms of consistency. In order to obtain an improved consistency, we need stronger local guarantees. To that end, we call a vertex *mandatory*, if it is part of every feasible query set. Identifying mandatory vertices based on the interval structure alone is not always possible, otherwise there would be a 1-competitive algorithm. Therefore, we want to identify vertices that are mandatory under the assumption that the predictions are correct. We call such vertices *prediction mandatory*. The following lemma gives a characterization of (prediction) mandatory vertices.

Lemma 2.3. *A vertex $v \in V$ is mandatory if and only if there is a hyperedge $S \in E$ with $v \in S$ such that either (i) v is a minimum-weight vertex of S and $w_u \in I_v$ for some $u \in S \setminus \{v\}$, or (ii) v is not a minimum-weight vertex of S and $w_u \in I_v$ for the minimum-weight vertex u of S.*

Proof. If v is a minimum-weight vertex of hyperedge S and I_v contains w_u of another vertex $u \in S \setminus \{v\}$, then S cannot be oriented even if we query all vertices in $S \setminus \{v\}$ as we cannot prove $w_v \leq w_u$ without querying v. If v is not a minimum-weight vertex of a hyperedge S with $v \in S$ and I_v contains the minimum weight w^* of S, then S cannot be solved even if we query all vertices in $S \setminus \{v\}$, as we cannot prove that $w^* \leq w_v$ without querying v.

If v is a minimum-weight vertex of hyperedge S, but $w_u \notin I_v$ for every $u \in S \setminus \{v\}$, then $S \setminus \{v\}$ is a feasible solution for orienting S. If v is not a minimum-weight vertex of hyperedge S and I_v does not contain the minimum weight of S, then again $S \setminus \{v\}$ is a feasible solution for S. If every hyperedge S that contains v falls into one of these two cases, then querying all vertices except v is a feasible query set for the whole instance. □

By using the lemma with the predicted weights instead of the precise weights, we can identify prediction mandatory vertices. Furthermore, the lemma does not only enable us to identify mandatory vertices given full knowledge of the precise weights, but also implies criteria to identify *known mandatory* vertices, i.e., vertices that are known to be mandatory given only the hypergraph, the intervals, and precise weights revealed by previous queries. Every algorithm can query such vertices without worsening its competitive ratio.

Corollary 2.4. *If the interval I_v of the leftmost vertex v in a hyperedge S contains the precise weight of another vertex in S, then v is mandatory. In particular, if v is leftmost in S and $I_u \subseteq I_v$ for some $u \in S \setminus \{v\}$, then v is mandatory.*

We use Lemma 2.3 to define an offline algorithm, i.e., we assume full access to the precise weights but still want to compute a feasible query set, that follows a two-stage structure: First, we iteratively query all mandatory vertices computed using Lemma 2.3. After that, each not yet oriented hyperedge S has the following configuration: The leftmost vertex v has a precise weight outside I_u for all $u \in S \setminus \{v\}$, and each other vertex in S has precise weight outside I_v. Thus we can either query v or all other vertices $u \in S \setminus \{v\}$ with $I_u \cap I_v \neq \emptyset$ to determine the orientation. The optimum solution is to query a minimum vertex cover in the following auxiliary graph as introduced in [Bampis *et al.*, 2021]:

Definition 2.5. *Given a hypergraph $H = (V, E)$, the* vertex cover instance *of H is the graph $G = (V, \bar{E})$ with $\{v, u\} \in \bar{E}$ if and only if there is a hyperedge $F \in E$ such that $v, u \in F$,*

v is leftmost in F and $I_v \cap I_u \neq \emptyset$. For the sorting problem, it holds that $\bar{G} = G$.

The Lemmas 2.2 and 2.3 directly imply that the offline algorithm is optimal. The algorithm may require exponential time, but this is not surprising as we also show that the offline version of the hypergraph orientation problem is NP-hard.

A key idea of our algorithms is to emulate the offline algorithm using the predicted information. Since blindly following the offline algorithm might lead to competitive ratio of n for faulty predictions, we have to augment the algorithm with additional, carefully selected queries.

The next lemma formulates a useful property of vertex cover instances without known mandatory vertices.

Lemma 2.6. *Given a hypergraph $H = (V, E)$ without known mandatory vertices (by Corollary 2.4), let Q be an arbitrary vertex cover of $\bar{G}$. After querying Q, for each hyperedge $F \in E$, we either know the orientation of F or can determine it by exhaustively querying according to Corollary 2.4.*

2.2 Accuracy of Predictions

While consistency and robustness only consider the extremes in terms of prediction quality, we aim for a more fine-grained analysis that relies on error metrics to measure the quality of the predictions. As was discussed in [Erlebach *et al.*, 2022], for the MST problem, simple error measures like the number of inaccurate predictions $k_{\#} = |\{v \in V \mid w_v \neq \overline{w}_v\}|$ or an ℓ_1 error metric such as $\sum_{e \in E} |w_e - \overline{w}_e|$ are not meaningful; this is also true for the hypergraph orientation problem. In particular, we show that even for $k_{\#} = 1$ the competitive ratio cannot be better than the known bound of 2 (cf. Appendix) for general hypergraph orientation. For the sorting problem, however, we show that $k_{\#}$-dependent guarantees are indeed possible. In general we need more refined measures that take the interleaving structure of intervals into account.

As a first refined measure, we consider the *hop distance* as proposed in [Erlebach *et al.*, 2022]. For a vertex v and any vertex $u \in V \setminus \{v\}$, we define the function $k_u(v)$ that indicates whether the relation of w_v to interval I_u changes compared to the relation of $\overline{w}_v$ and I_u. To be more precise, $k_u(v) = 1$ if $\overline{w}_v \leq L_u < w_v$, $w_v \leq L_u < \overline{w}_v$, $w_v < U_u \leq \overline{w}_v$ or $\overline{w}_v < U_u \leq w_v$, and $k_u(v) = 0$ otherwise. Based on this function, we define the hop distance of a single vertex as $k^+(v) = \sum_{u \in V \setminus \{v\}} k_u(v)$. Intuitively $k^+(v)$ for a single $v \in V$ counts the number of relations between w_v and intervals I_u with $u \in V \setminus \{v\}$ that are not accurately predicted. For a set of vertices $V' \subseteq V$, we define $k^+(V') = \sum_{v \in V'} k^+(v)$. Finally, we define the hop distance by $k_h = k^+(V)$. For an example see Figure 1.

Note that $k_{\#} = 0$ implies $k_h = 0$, so Theorem 2.1 implies that no algorithm can simultaneously have competitive ratio better than $1 + \frac{1}{\beta}$ if $k_h = 0$ and β for arbitrary k_h.

While the hop distance takes the interval structure into account, it does not distinguish whether a "hop" affects a feasible solution. We introduce a third and strongest error measure based on the sets of (prediction) mandatory elements.

Let $\mathcal{I}_P$ be the set of prediction mandatory elements, and let $\mathcal{I}_R$ be the set of really mandatory elements. The *mandatory query distance* is the size of the symmetric difference of $\mathcal{I}_P$ and $\mathcal{I}_R$, i.e., $k_M = |\mathcal{I}_P \Delta \mathcal{I}_R| = |(\mathcal{I}_P \cup \mathcal{I}_R) \setminus (\mathcal{I}_P \cap \mathcal{I}_R)| = |(\mathcal{I}_P \setminus \mathcal{I}_R) \cup (\mathcal{I}_R \setminus \mathcal{I}_P)|$. Figure 1 (right) shows an example with $k_M = 1$. Considering the precise weights in the example, both $\{v_1\}$ and $\{v_2, v_3, v_4\}$ are feasible solutions. Thus, no element is part of every feasible solution and $\mathcal{I}_R = \emptyset$. Assuming correct predicted weights, we have that v_1 has to be queried even if all other vertices have already been queried and, therefore, $\mathcal{I}_P = \{v_1\}$. It follows $k_M = |\mathcal{I}_P \Delta \mathcal{I}_R| = 1$.

Obviously, k_M is a problem-specific error measure as, in a given set of uncertainty intervals, different intervals may be mandatory for different problems. We can relate k_M to k_h.

Theorem 2.7. *For any instance of hypergraph orientation under uncertainty with predictions, the hop distance is at least as large as the mandatory query distance, i.e., $k_M \leq k_h$.*

The learnability results of [Erlebach *et al.*, 2022] translate to the hypergraph orientation problem and show that the predictions are PAC-learnable (cf. [Valiant, 1984]) w.r.t. k_h with polynomial running time and sample complexity. In the full version, we show PAC-learnability w.r.t. k_M. While this might require exponential running time, we remark that the set $\mathcal{I}_P$ can be predicted with polynomial running time and that access to that set is sufficient information to execute our k_M-dependent algorithm for hypergraph orientation. For sorting, our algorithm relies on access to the predicted weights. Note that, under the PAC-learning assumption that we can sample the precise weights, it is possible to apply the algorithm for the stochastic problem variant given in [Bampis *et al.*, 2021]. However, in contrast to the results of this paper, the guarantee of the algorithm given in [Bampis *et al.*, 2021] holds in expectation instead of in the worst-case and admits no robustness. Furthermore, our algorithms work as a black-box independent of how the predictions are obtained.

3 Hypergraph Orientation

We consider the general hypergraph orientation problem, and give error-sensitive algorithms w.r.t. the measures k_h and k_M. For k_h, we show that we can use an adjusted variant of the algorithm given in [Erlebach *et al.*, 2022] for the MST problem.

To achieve error-sensitive guarantees w.r.t k_M, we show that the k_h-dependent guarantee does not directly translate to k_M. Instead, we give an even simpler algorithm that emulates the offline algorithm and augments it with additional queries, and show that this algorithm achieves the optimal trade-off w.r.t k_M. We remark that this algorithm only requires access to the set of prediction mandatory vertices, which is a weaker type of prediction than access to the predicted weights $\overline{w}$.

3.1 Error-Sensitive Algorithm w.r.t. Hop Distance

In the full version, we show how to adjust the algorithm and analysis given in [Erlebach *et al.*, 2022] for the MST problem to prove the following theorem. In contrast to the result of [Erlebach *et al.*, 2022], we achieve the optimal consistency-robustness tradeoff by essentially exploiting Lemma 2.6. The corresponding analysis requires additional work specific to the hypergraph orientation problem.

Theorem 3.1. *There is an algorithm for hypergraph orientation under uncertainty that, given $\gamma \in \mathbb{N}_{\geq 2}$, achieves a competitive ratio of $\min\{(1 + \frac{1}{\gamma})(1 + k_h/|\mathrm{OPT}|), \gamma\}$.*

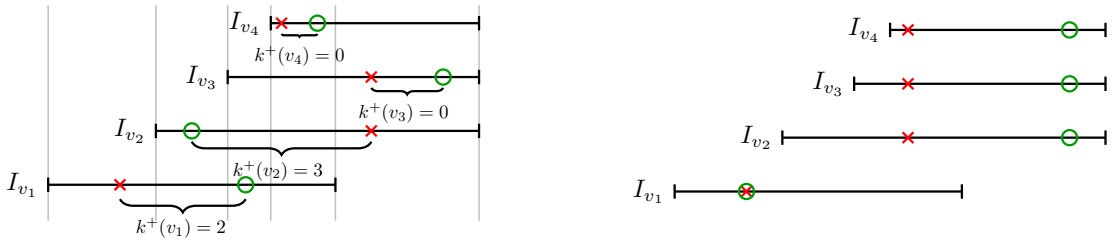

Figure 1: Example from [Erlebach *et al.*, 2022] interpreted for the hypergraph orientation problem with a hyperedge $S = \{v_1, v_2, v_3, v_4\}$. Circles illustrate precise weights and crosses illustrate the predicted weights. Predictions and precise weights with a total hop distance of $k_h = 5$ and mandatory query distance of $k_M = 1$ (left) and $k_h = 3$ and $k_M = 1$ (right).

3.2 An Error-Sensitive Algorithm w.r.t. the Mandatory-Query Distance

We start by providing the following lower bound.

Theorem 3.2. *Let $\gamma \in \mathbb{R}_{\geq 2}$ be fixed. If a deterministic algorithm for hypergraph orientation under uncertainty is γ-robust, then it cannot have competitive ratio better than $1 + \frac{1}{\gamma-1}$ for $k_M = 0$. If an algorithm has competitive ratio $1 + \frac{1}{\gamma-1}$ for $k_M = 0$, then it cannot be better than γ-robust.*

This lower bound shows that the guarantee of Theorem 3.1 cannot translate to the error measure k_M.

Further, we prove the following tight bound by presenting the new Algorithm 1 with dependency on k_M. Note that this algorithm only uses the initial set of prediction mandatory vertices, and otherwise ignores the predicted weights. Access to this set is sufficient to execute the algorithm.

Theorem 3.3. *There is an algorithm for hypergraph orientation under uncertainty that, given an integer parameter $\gamma \geq 2$, has a competitive ratio of $\min\{(1+\frac{1}{\gamma-1})\cdot(1+\frac{k_M}{|\mathrm{OPT}|}), \gamma\}$.*

The full proof is in the Appendix; here we only give a proof sketch. The algorithm emulates the two-stage structure of the offline algorithm. Recall that the offline algorithm in a first stage queries all mandatory vertices and in a second stage queries a minimum vertex cover in the remaining vertex cover instance. Since blindly following the offline algorithm based on the predicted weights would lead to a competitive ratio of n, the algorithm augments both stages with additional queries. Algorithm 1 implements the augmented first stage in Lines 2 to 7 and afterwards executes the second stage.

To start the first phase, the algorithm computes the set P of initial prediction mandatory vertices (Lemma 2.3). Then it tries to find a vertex $p \in P$ that is part of a witness set $\{p, b\}$. If $|P| \geq \gamma - 1$, we query a set $P' \subseteq P$ of size $\gamma - 1$ that includes p, plus b (we allow $b \in P'$). This is clearly a witness set of size at most γ, which ensures that the queries do not violate the γ-robustness. Also, at least a $\frac{\gamma-1}{\gamma}$ fraction of the queried vertices are in P, and every vertex in $P \setminus \mathrm{OPT}$ is in $\mathcal{I}_P \setminus \mathcal{I}_R$. This ensures, at least locally, that the queried vertices do not violate the error-dependent consistency. We then repeatedly query known mandatory vertices, remove them from P and repeat without recomputing P, until P is empty or no vertex in P is part of a witness set.

We may have one last iteration of the loop where $|P| < \gamma - 1$. After that, the algorithm will proceed to the second phase, querying a minimum vertex cover and intervals that become known mandatory. For the second phase itself, we can use that a minimum vertex cover of the vertex cover instance (cf. Definition 2.5) is a lower bound on the optimal solution for the remaining instance by Lemma 2.2. Since all queries of Line 9 are mandatory, the queries of the Lines 8 and 9 are 2-robust for the remaining instance. Even in combination with the additional at most $\gamma - 2$ queries of the last iteration of the loop, this is still γ-robust. It is not hard to show that each query of Line 9 contributes an error to k_M, which completes the argument.

Algorithm 1: Algorithm for hypergraph orientation under uncertainty w.r.t. error measure k_M

Input: Hypergraph $H = (V, E)$, intervals I_v and predictions $\overline{w}_v$ for all $v \in V$

1 $P \leftarrow$ set of initial prediction mandatory vertices (characterized in Lemma 2.3);
2 **while** $\exists p \in P$ *and an unqueried vertex* b *where* $\{p, b\}$ *is a witness set for the current instance by Lemma 2.2* **do**
3 **if** $|P| \geq \gamma - 1$ **then**
4 pick $P' \subseteq P$ with $p \in P'$ and $|P'| = \gamma - 1$;
5 query $P' \cup \{b\}$, $P \leftarrow P \setminus (P' \cup \{b\})$;
6 **while** *there is a known mandatory vertex* v **do** query v, $P \leftarrow P \setminus \{v\}$;
7 **else** query P, $P \leftarrow \emptyset$;
8 Compute and query a minimum vertex cover Q' on the current vertex cover instance;
9 Exhaustively apply Corollary 2.4 ;

4 Sorting under Uncertainty

We consider the special case of the hypergraph orientation problem, where the input graph is a simple graph $G = (V, E)$ that satisfies $\{u, v\} \in E$ if and only if $I_v \cap I_u \neq \emptyset$. That is, G corresponds to the interval graph induced by the uncertainty intervals $\mathcal{I} = \{I_v \mid v \in V\}$. To orient such a graph, we have to, for each pair of intersecting intervals, decide which one has the smaller precise weight. An orientation of the graph defines an order of the intervals according to their precise weights (and vice versa). Thus, the problem corresponds to the problem of sorting a single set of uncertainty intervals. Note that, by querying vertices, the uncertainty in-

tervals change and, thus, the graph induced by the intervals also changes. When we speak of the *current interval graph*, we refer to the interval graph induced by the uncertainty intervals after all previous queries. We show in the full version:

Theorem 4.1. *Any deterministic algorithm for sorting or hypergraph orientation with predictions (even for pairwise disjoint hyperedges) has a competitive ratio* $\rho \geq \min\{1 + \frac{k}{|\mathrm{OPT}|}, 2\}$*, for any error measure* $k \in \{k_\#, k_M, k_h\}$.

As a main result, we show a matching upper bound.

Theorem 4.2. *There exists a single polynomial-time algorithm for sorting under uncertainty with predictions that is* $\min\{1 + \frac{k}{|\mathrm{OPT}|}, 2\}$*-competitive for any* $k \in \{k_\#, k_M, k_h\}$.

The key observation that allows us to achieve improved results for interval graphs is the simple characterization of mandatory vertices: any vertex with an interval that contains the precise weight of another vertex is mandatory [Halldórsson and de Lima, 2021]. This observation is a direct consequence of Lemma 2.3 and the structure of interval graphs. Analogously, each vertex with an interval that contains the predicted weight of another vertex is prediction mandatory. Furthermore, Lemma 2.2 implies that any two vertices with intersecting intervals constitute a witness set.

To obtain a guarantee of $|\mathrm{OPT}| + k$ for any measure k, our algorithm (cf. Algorithm 2) must trust the predictions as much as possible. That is, the algorithm must behave very close to the offline algorithm under the assumption that the predictions are correct. Recall that the offline algorithm in a first stage queries all mandatory vertices and in a second stage queries a minimum vertex cover in the remaining vertex cover instance. Algorithm 2 emulates again these two stages. In contrast to the algorithms for general hypergraph orientation, we cannot afford to augment the stages with additional queries as we aim at achieving 1-consistency. Thus, we need a new algorithm and cannot apply existing results.

In the emulated first phase, our algorithm queries all prediction mandatory intervals (cf. Line 4) and all intervals that are mandatory based on the already obtained information (cf. Lines 2 and 5). This phase clearly does not violate the $|\mathrm{OPT}| + k$ guarantee for $k \in \{k_M, k_h\}$, as all queried known mandatory vertices (cf. Lines 2 and 5) are contained in OPT and all queried prediction mandatory vertices (cf. Line 4) are either in OPT or contribute one to $k_M \leq k_h$. We will show that the same holds for $k = k_\#$. However, the main challenge is to guarantee 2-robustness. Our key ingredient for ensuring this is the following lemma, which we show in the full version.

Lemma 4.3. *For an instance of the sorting problem, let* $\mathcal{I}_P$ *be the set of prediction mandatory vertices and* M *the set of known mandatory vertices after querying* $\mathcal{I}_P$ *(by exhaustively applying Corollary 2.4). Then, we can partition* $\mathcal{I}_P \cup M$ *into a set of disjoint cliques* $\mathcal{C}$ *such that each* v *with* $\{v\} \in \mathcal{C}$ *either satisfies* $v \in M$ *or* $I_v \cap I_u \neq \emptyset$ *for a distinct* $u \notin \mathcal{I}_P \cup M$*. The partition can be computed in polynomial time.*

We can apply the lemma to the queries of the first phase of the algorithm (cf. Line 7). Given the partition $\mathcal{C}$ of the lemma, we know that queries to vertices v that are part of some $C \in \mathcal{C}$ with $|C| \geq 2$ (or mandatory, i.e., $v \in M$) do not violate the 2-robustness as even the optimal solution can avoid at most one query per clique [Halldórsson and de Lima, 2021]. Thus, we only have to worry about vertices $v \notin M$ with $\{v\} \in \mathcal{C}$. We call such vertices *critical* isolated vertices. But even for critical isolated vertices v, the lemma gives us a distinct not yet queried u with $I_v \cap I_u \neq \emptyset$, i.e., $\{v, u\}$ is a witness set.

Algorithm 2: A nicely degrading algorithm for sorting with uncertainty and predictions

Input: Interval graph $G = (V, E)$, intervals I_v and predictions $\overline{w}_v$ for all $v \in V$

1 $\mathcal{I}_P \leftarrow$ set of prediction mandatory vertices;
2 **while** $\exists v \neq u$ *with* $I_u \subseteq I_v$*, or* u *was queried and* $w_u \in I_v$ **do** query v;
3 $M_1 \leftarrow$ vertices queried in Line 2; $\mathcal{S} \leftarrow \mathcal{I}_P \setminus M_1$;
4 Query $\mathcal{S}$;
5 Exhaustively apply Corollary 2.4 ;
6 $M_2 \leftarrow$ set of vertices queried in Line 5;
7 $\mathcal{C} \leftarrow$ Clique partition of $\mathcal{S} \cup M_1 \cup M_2$ such that all isolated vertices v satisfy either $v \in M_1 \cup M_2$ or $I_u \cap I_v \neq \emptyset$ for a distinct $u \notin \mathcal{S} \cup M_1 \cup M_2$ (computed using Lemma 4.3);
8 **while** *the problem is unsolved* **do**
9 **let** $P = x_1 x_2 \cdots x_p$ be a path component of the current interval graph with $p \geq 2$ in direction of non-increasing lower limits L_{x_i};
10 **if** p *is odd* **then** query $\{x_2, x_4, \ldots, x_{p-1}\}$;
11 **else**
12 **if** x_1 *is the distinct partner of a critical isolated vertex* v *(*$I_{x_1} \cap I_v \neq \emptyset$ *and* $v \notin M_1 \cup M_2$*; cf. Lemma 4.3)* **then** query $\{x_1, x_3, \ldots, x_{p-1}\}$;
13 **else** query $\{x_2, x_4, \ldots, x_p\}$;
14 Exhaustively apply Corollary 2.4;

In line with the offline algorithm, the second phase of the algorithm (cf. Lines 8 to 14) queries a minimum vertex cover of the remaining instance (the interval graph defined by the intervals of non-queried vertices). However, to guarantee 2-robustness, we have to take the witness sets of the critical isolated vertices into account when deciding which vertex cover to query. To see this, consider the example of Figure 2: only v_1 is prediction mandatory, so the first phase of the algorithm just queries v_1. After querying v_1, there are no prediction mandatory (or mandatory) intervals left. One possible minimum vertex cover of the remaining instance is $\{v_3, v_5\}$, but querying this vertex cover renders v_2 and v_4 mandatory. Thus, the algorithm would query all five intervals, which violates the 2-robustness as the optimal solution just queries $\{v_2, v_4\}$. The example shows that the selection of the minimum vertex cover in the second phase is important.

We show in the Appendix that instances occurring in the second phase have a structure similar to the example by using that such instances have no prediction mandatory vertices.

Lemma 4.4. *Each connected component of an interval graph without prediction mandatory vertices is either a path or a single vertex.*

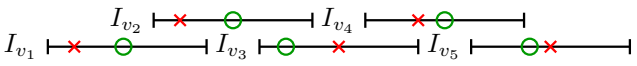

Figure 2: Example showing that it is necessary to query a specific vertex cover in the second phase to ensure 2-robustness. Circles illustrate precise values and crosses illustrate the predicted values.

Further, we observe that if the intervals of critical isolated vertices intersect intervals of vertices on such a path component, they intersect the interval of an endpoint of the component. Otherwise, the predicted weight $\overline{w}_v$ of the critical isolated vertex v would be contained in the interval of at least one vertex on the path component, which contradicts the vertices on the path not being prediction mandatory. The distinct partner u of a critical isolated vertex v that exists by Lemma 4.3 is an endpoint of such a path component.

The second phase of our algorithm iterates through all such connected components and, for each component, queries a minimum vertex cover (cf. Lines 10, 12 and 13) and all resulting mandatory intervals (cf. Line 14). If the path is of odd length, then the minimum vertex cover is unique. Otherwise, the algorithm selects the minimum vertex cover based on whether the interval of a critical isolated vertex intersects the interval of the first path endpoint. Lemma 2.6 guarantees that the algorithm indeed queries a feasible query set. The following lemma shows that this strategy indeed ensures 2-robustness (using Lemma 4.3).

Lemma 4.5. *Algorithm 2 is 2-robust.*

Proof. Fix an optimum solution OPT. Let M_1, M_2 and $\mathcal{S}$ denote the phase one queries of the algorithm as defined in the pseudocode. Consider the clique partition $\mathcal{C}$ as computed in Line 7, then all $C \in \mathcal{C}$ with $|C| \geq 2$ satisfy $|C| \leq 2 \cdot |C \cap \mathrm{OPT}|$ and all $C \in \mathcal{C}$ with $C \subseteq M_1 \cup M_2$ satisfy $|C| \leq |C \cap \mathrm{OPT}|$. The latter holds as all members of $M_1 \cup M_2$ are mandatory by Lemma 2.3. Queries to vertices that are covered by such cliques do not violate the 2-robustness. This leaves members of $\mathcal{S}$ that are critical isolated vertices in $\mathcal{C}$ and queries of the second phase. We partition such queries (and some non-queried vertices) into a collection $\mathcal{W}$ such that, for each $W \in \mathcal{W}$, the algorithm queries at most $2 \cdot |W \cap \mathrm{OPT}|$ intervals in W. If we have such a partition, then it is clear that we spend at most $2 \cdot |\mathrm{OPT}|$ queries and are 2-robust.

By Lemma 4.3, there is a distinct vertex $u \notin M_1 \cup M_2 \cup \mathcal{S}$ for each critical isolated vertex v with $I_v \cap I_u \neq \emptyset$; as argued above, u is the endpoint of a path component of the current instance before line 8. We create the partition $\mathcal{W}$ as follows: Iteratively consider all connected (path) components P of the current instance before line 8. Let W be the union of P and the critical isolated vertices that are the distinct partner of at least one endpoint of P. If $|W| \geq 2$, add W to $\mathcal{W}$. Clearly, $\mathcal{W}$ contains all critical isolated vertices of $\mathcal{C}$ and all intervals that are queried in the Lines 10, 12, 13 and 14.

We conclude the proof by arguing that each $W \in \mathcal{W}$ satisfies that the algorithm queries at most $2 \cdot |W \cap \mathrm{OPT}|$ vertices in W. By construction, W contains a path component P and up-to two critical isolated vertices. Furthermore, W itself is a path in the initial interval graph (in addition to the edges of the path, there may be an additional edge between each critical isolated vertex of $\mathcal{C}$ in W and the second or penultimate vertex of P, but this does not affect the argument that follows). Consider an arbitrary $W \in \mathcal{W}$. If $|W|$ is even, then $|W| \leq 2 \cdot |W \cap \mathrm{OPT}|$ as all pairs of neighboring vertices in path W are witness pairs. Thus, assume that $|W|$ is odd. As each critical isolated vertex has a distinct partner by Lemma 4.3 and this partner is an endpoint of a path component, W contains at most one critical isolated vertex per distinct endpoint of P and we have $|P| \geq 2$.

We divide the analysis in two cases. First assume that $|P| = p$ is odd. Then the algorithm queries $\{x_2, x_4, \ldots, x_{p-1}\}$ in Line 10. As P is a path, the precise weight of each queried vertex can be contained in the interval of at most one other vertex of P and, therefore, force at most one query in Line 14. This leaves at least one vertex in $P \subseteq W$ that is never queried by the algorithm. Since $|W \cap \mathrm{OPT}| \geq \lfloor |W|/2 \rfloor$ (as the subgraph induced by W contains a path of the vertices in W), clearly the algorithm queries at most $2 \cdot |W \cap \mathrm{OPT}|$ vertices in W.

Now assume that $|P| = p$ is even. Then either x_1 or x_p (but not both) is the distinct partner of a critical isolated member of W, otherwise $|W|$ would be even. If I_{x_1} intersects I_v of the critical isolated vertex v, then the algorithm queries $\{x_1, x_3, \ldots, x_{p-1}\}$ in Line 12. If w_{x_1} forces a query to x_2 in Line 14, then $|\{x_1, x_2, v\}| \leq 2 \cdot |\{x_1, x_2, v\} \cap \mathrm{OPT}|$ and the remaining vertices $W' = W \setminus \{x_1, x_2, v\}$ form an even path, which implies $|W'| \leq 2 \cdot |W' \cap \mathrm{OPT}|$ and, therefore $|W| \leq 2 \cdot |W \cap \mathrm{OPT}|$. If w_{x_1} forces no query to x_2 in Line 14, then $|\{x_1, v\}| \leq 2 \cdot |\mathrm{OPT} \cap \{x_1, v\}|$ and we analyze $W' = W \setminus \{x_1, v\}$ as in the subcase for odd $|P|$. Hence, the algorithm queries at most $2 \cdot |W \cap \mathrm{OPT}|$ intervals of W.

If I_{x_p} intersects the interval I_v of critical isolated member v, then we can analyze W analogously. □

Lemma 4.6. *Algorithm 2 spends at most* $|\mathrm{OPT}| + k_M \leq |\mathrm{OPT}| + k_h$ *queries.*

Proof. We show that the algorithm spends at most $|\mathrm{OPT}| + k_M$ queries. Theorem 2.7 implies $|\mathrm{OPT}| + k_M \leq |\mathrm{OPT}| + k_h$. Fix an optimum solution OPT. Every vertex queried in Lines 2 and 5 is in OPT by Lemma 2.3. Every vertex queried in Line 4 that is not in OPT is clearly in $\mathcal{I}_P \setminus \mathcal{I}_R$ and contributes one to k_M.

For each path P considered in Line 9, let P' be the vertices queried in Lines 10–13. It clearly holds that $|P'| \leq |P \cap \mathrm{OPT}|$. Finally, every vertex queried in Line 14 is in $\mathcal{I}_R \setminus \mathcal{I}_P$, and therefore contributes to k_M, because we query all prediction mandatory vertices at the latest in Line 4. □

The previous two lemmas imply Theorem 4.2 for $k \in \{k_h, k_M\}$. In the full version, we show the same for $k = k_\#$.

5 Final Remarks

We showed how untrusted predictions enable us to circumvent known lower bounds for hypergraph orientation and sorting under explorable uncertainty and sparked the discussion on error measures by presenting a new error metric and showing relations between the different errors. As a next research step, we suggest investigating the application of error measure k_M for different problems under explorable uncertainty.

Acknowledgements

This research was supported by the German Science Foundation (DFG) grant 517912373 and the EPSRC grants EP/S033483/1, EP/S033483/2, and EP/T01461X/1.

References

[Albers and Eckl, 2020] Susanne Albers and Alexander Eckl. Explorable uncertainty in scheduling with non-uniform testing times. In *WAOA*, volume 12806 of *Lecture Notes in Computer Science*, pages 127–142. Springer, 2020.

[Albers and Eckl, 2021] Susanne Albers and Alexander Eckl. Scheduling with testing on multiple identical parallel machines. In *WADS*, volume 12808 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2021.

[Angelopoulos *et al.*, 2020] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc P. Renault. Online computation with untrusted advice. In *ITCS*, volume 151 of *LIPIcs*, pages 52:1–52:15, 2020.

[Arantes *et al.*, 2018] Luciana Arantes, Evripidis Bampis, Alexander V. Kononov, Manthos Letsios, Giorgio Lucarelli, and Pierre Sens. Scheduling under uncertainty: A query-based approach. In *IJCAI 2018*, pages 4646–4652, 2018.

[Azar *et al.*, 2021] Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *STOC*, pages 1070–1080. ACM, 2021.

[Azar *et al.*, 2022a] Yossi Azar, Stefano Leonardi, and Noam Touitou. Distortion-oblivious algorithms for minimizing flow time. In *SODA*, pages 252–274. SIAM, 2022.

[Azar *et al.*, 2022b] Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *SODA*, pages 35–66. SIAM, 2022.

[Bampis *et al.*, 2021] Evripidis Bampis, Christoph Dürr, Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Orienting (hyper)graphs under explorable stochastic uncertainty. In *ESA*, volume 204 of *LIPIcs*, pages 10:1–10:18, 2021.

[Bampis *et al.*, 2022] Evripidis Bampis, Konstantinos Dogeas, Alexander V. Kononov, Giorgio Lucarelli, and Fanny Pascual. Scheduling with untrusted predictions. In *IJCAI*, pages 4581–4587. ijcai.org, 2022.

[Bruce *et al.*, 2005] Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. Efficient update strategies for geometric computing with uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005.

[Chaplick *et al.*, 2021] Steven Chaplick, Magnús M. Halldórsson, Murilo S. de Lima, and Tigran Tonoyan. Query minimization under stochastic uncertainty. *Theor. Comput. Sci.*, 895:75–95, 2021.

[Dürr *et al.*, 2020] Christoph Dürr, Thomas Erlebach, Nicole Megow, and Julie Meißner. An adversarial model for scheduling with testing. *Algorithmica*, 82(12):3630–3675, 2020.

[Eberle *et al.*, 2022] Franziska Eberle, Alexander Lindermayr, Nicole Megow, Lukas Nölke, and Jens Schlöter. Robustification of online graph exploration methods. In *AAAI*, 2022.

[Erlebach and Hoffmann, 2014] Thomas Erlebach and Michael Hoffmann. Minimum spanning tree verification under uncertainty. In *WG*, volume 8747 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2014.

[Erlebach and Hoffmann, 2015] Thomas Erlebach and Michael Hoffmann. Query-competitive algorithms for computing with uncertainty. *Bulletin of the EATCS*, 116:22–39, 2015.

[Erlebach *et al.*, 2008] Thomas Erlebach, Michael Hoffmann, Danny Krizanc, Matús Mihalák, and Rajeev Raman. Computing minimum spanning trees with uncertainty. In *STACS*, pages 277–288, 2008.

[Erlebach *et al.*, 2022] Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Learning-augmented query policies for minimum spanning tree with uncertainty. In *ESA*, volume 244 of *LIPIcs*, pages 49:1–49:18, 2022.

[Erlebach *et al.*, 2023] Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Sorting and hypergraph orientation under uncertainty with predictions. *CoRR*, abs/2305.09245, 2023.

[Feder *et al.*, 2007] Tomás Feder, Rajeev Motwani, Liadan O'Callaghan, Chris Olston, and Rina Panigrahy. Computing shortest paths with uncertainty. *Journal of Algorithms*, 62(1):1–18, 2007.

[Focke *et al.*, 2020] Jacob Focke, Nicole Megow, and Julie Meißner. Minimum spanning tree under explorable uncertainty in theory and experiments. *ACM J. Exp. Algorithmics*, 25:1–20, 2020.

[Goerigk *et al.*, 2015] Marc Goerigk, Manoj Gupta, Jonas Ide, Anita Schöbel, and Sandeep Sen. The robust knapsack problem with queries. *Computers & Operations Research*, 55:12–22, 2015.

[Gong *et al.*, 2022] Mingyang Gong, Randy Goebel, Guohui Lin, and Eiji Miyano. Improved approximation algorithms for non-preemptive multiprocessor scheduling with testing. *J. Comb. Optim.*, 44(1):877–893, 2022.

[Halldórsson and de Lima, 2021] Magnús M. Halldórsson and Murilo Santos de Lima. Query-competitive sorting with uncertainty. *Theor. Comput. Sci.*, 867:50–67, 2021.

[Kahan, 1991] Simon Kahan. A model for data in motion. In *STOC'91: 23rd Annual ACM Symposium on Theory of Computing*, pages 265–277, 1991.

[Kumar *et al.*, 2019] Ravi Kumar, Manish Purohit, Aaron Schild, Zoya Svitkina, and Erik Vee. Semi-online bipartite matching. In *ITCS*, volume 124 of *LIPIcs*, pages 50:1–50:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[Lattanzi *et al.*, 2020] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online

scheduling via learned weights. In *SODA*, pages 1859–1877. SIAM, 2020.

[Li and Xian, 2021] Shi Li and Jiayi Xian. Online unrelated machine load balancing with predictions revisited. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 6523–6532. PMLR, 2021.

[Lindermayr and Megow, 2022] Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. In *SPAA*, pages 357–368. ACM, 2022.

[Lindermayr *et al.*, 2022] Alexander Lindermayr, Nicole Megow, and Bertrand Simon. Double Coverage with Machine-Learned Advice. In *ITCS*, volume 215 of *LIPIcs*, pages 99:1–99:18, 2022.

[Lykouris and Vassilvtiskii, 2018] Thodoris Lykouris and Sergei Vassilvtiskii. Competitive caching with machine learned advice. In *Proceedings of ICML*, pages 3302–3311, 2018.

[Megow *et al.*, 2017] Nicole Megow, Julie Meißner, and Martin Skutella. Randomization helps computing a minimum spanning tree under uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, 2017.

[Merino and Soto, 2019] Arturo I. Merino and José A. Soto. The minimum cost query problem on matroids with uncertainty areas. In *ICALP*, volume 132 of *LIPIcs*, pages 83:1–83:14, 2019.

[Mitzenmacher, 2020] Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In *Proceedings of ITCS*, volume 151 of *LIPIcs*, pages 14:1–14:18, 2020.

[Purohit *et al.*, 2018] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Proceedings of NIPS*, pages 9661–9670, 2018.

[Valiant, 1984] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.