UNIVERSITÄT BREMEN

DISSERTATION

Optimization under Explorable Uncertainty: Beyond the Worst-Case

vorgelegt von
Jens Schlöter, M.Sc.
geboren in Georgsmarienhütte

Vom Fachbereich 3 – Mathematik und Informatik
der Universität Bremen
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Gutachter*innen:	Prof. Dr. Nicole Megow
	Prof. Dr. Ola Svensson

Tag der wissenschaftlichen Aussprache: 10. November 2023

Bremen, 2024

Acknowledgments

First and foremost, I want to thank my advisor Nicole Megow for the chance to pursue my PhD in her group, for her support and for introducing me to the field of combinatorial optimization and optimization under uncertainty. I am thankful for all the fruitful research discussions and for the great atmosphere in her group. Furthermore, I want to thank all my collaborators. In particular, I want to thank Thomas Erlebach, Murilo Santos de Lima, Christoph Dürr and Evripidis Bampis, who worked with me on the topic of this thesis. We started working with Thomas and Murilo shortly after I started my PhD and I very much appreciate the opportunity to learn from them about research early on. I also want to thank Ola Svensson for taking on the second assessment of this thesis.

Moreover, I am grateful to all of my colleagues for all the discussions and the great environment in the office. In particular, I want to thank Alex Lindermayr and Lukas Nölke for proofreading different parts of this thesis. A special thanks goes to Alex for all the fun "side" projects that distracted me when I was frustrated with the progress on my thesis and turned into very nice results.

Finally, I want to thank my family and friends for the support throughout my PhD.

Bremen, January 2024 Jens Schlöter

Table of Contents

Chapter 1

Introduction

When solving optimization problems that arise in real-world applications, uncertainty in the input data and incomplete information are major challenges. Consider for example varying transportation times depending on traffic conditions or weather, unknown execution times of tasks to be scheduled, variable parameters such as bandwidth and demands, dynamically changing locations of moving agents, or decentralized data that is updated infrequently.

Since uncertainty is ubiquitous in a wide range of real-world applications, there has been plenty of research devoted to providing mathematical frameworks for modeling uncertainty in optimization problems. A first major framework is *online optimization*, where parts of the input are initially unknown and the missing data is revealed incrementally over time. Whenever new data arrives, an optimization algorithm has to make irrevocable decisions on how to handle the new information (cf., e.g., [BE98]). *Stochastic optimization* refers to a setting where the uncertain input data is modeled via (known or unknown) probability distributions. The goal is to find solutions that perform good in expectation or with high probability. The actual realizations of the uncertain parts of the input are usually revealed sequentially in a number of stages (cf., e.g., [BL11]). In the third major framework, *robust optimization*, the uncertainty is typically modeled by an, explicitly or implicitly, given set of scenarios that could potentially occur. The most common goal is to find a single solution that performs reasonably well for every possible scenario (cf., e.g., [BGN09]).

All these models have in common that the uncertain information is either revealed passively over time or not at all. Optimization algorithms for problems in these models have to cope with this type of passively revealed uncertainty and have to make decisions based on incomplete information. In particular, they do not have the option, not even at a cost, to actively obtain new information that helps to handle the optimization task at hand. In a number of real-world applications however, the possibility to *query* uncertain parts of the input at a certain cost is a reasonable assumption. Uncertain execution times could for example be determined by executing further analysis (cf. [Sha16] for an example in maintenance work using fault analysis), variable parameters such as the bandwidth of a network connection can be measured, dynamically changing locations of moving agents can be determined via communication [Kah91] and decentralized data can be updated by queries to a master database [OW00].

The area of *explorable uncertainty* was introduced in a seminal paper by Kahan [Kah91] and considers exactly such scenarios where uncertain parts in the input of an optimization problem can be queried to reveal more information. As queries are costly, the goal is to minimize the query cost necessary to find an optimal (or approximative) solution for the underlying optimization problem. In particular, the model considers uncertainty in numerical input parameters. Instead of having access to the precise values of these parameters, we are given *uncertainty intervals* that are guaranteed to contain the corresponding precise values, but we do not have any information on where inside the interval the value actually is. Since the optimal solution of the underlying optimization problem can depend on the uncertain parameters, finding an optimal (or approximative) solution with respect to the uncertain values might be impossible without obtaining further information. Thus, the goal in explorable uncertainty is to design algorithms that query uncertain parameters until the revealed information is sufficient to determine an optimal solution to the underlying optimization problem, with the objective to minimize the query cost. This thesis, and the majority of existing research in the field, considers *adaptive algorithms* for problems under explorable uncertainty that are allowed to take the results of previous queries into account when deciding upon the next query.

As an example, consider the classic minimum spanning tree problem, where we are given an undirected graph $G = (V, E)$ with edge weights w_e for all $e \in E$. The goal is to find a *minimum spanning tree (MST)*, i.e., a subset $T \subseteq E$ of minimum weight $w(T) = \sum_{e \in T} w_e$ such that the subgraph $G' = (V, T)$ is connected and contains no cycles. Under explorable uncertainty, the precise edge weights w_e are initially unknown. Instead, we only have access to *uncertainty intervals* $I_e = (L_e, U_e)$, for all $e \in E$, that are guaranteed to contain the precise edge weights, i.e., $w_e \in I_e$. The goal remains to find an MST for the unknown precise edge weights. To find such an MST despite the lack of information on the weights, we are allowed to query edges and a query to an edge e reveals the precise weight w_e at query cost c_e. The task is to design algorithms that adaptively query edges until the revealed information is sufficient to determine an MST w.r.t. the precise weights, while minimizing the total query cost.

So far, explorable uncertainty has mostly been studied in the *adversarial (or worst-case) setting*, where we assume that query results are returned in a worst-case manner. Since there usually exist problem instances that cannot be solved without querying all uncertain parameters, algorithms in this setting are typically analyzed in an instance-dependent way via *competitive analysis*. We give a formal definition later, but we say that an algorithm is *ρ-competitive* for a problem under explorable uncertainty if, for every problem instance, the query cost of the algorithm is larger by a factor of at most ρ than the optimal query cost for that particular instance. The minimum ρ for which an algorithm is ρ-competitive is called the *competitive ratio* of the algorithm.

The most studied problems in adversarial explorable uncertainty are selection-type problems, e.g., selecting the minimum [Kah91], sorting [HL21], selecting the k'th smallest element [Kah91; Fed+03], selecting a minimum spanning tree [Hof+08; EH14; MMS17] and geometric problems [Bru+05]. These problems are well-understood and admit constant-competitive algorithms with matching lower bounds. All these problems have in common that they essentially (but non-trivially) can be reduced to comparing single uncertainty intervals and that the lower bounds are caused already by very simple problem instances. Once we have to compare two sets (sums) of uncertainty intervals, no deterministic algorithm can have a better competitive ratio than $\Omega(n)$ [EHK16], where n is the number of given uncertainty intervals. This lower bound of $\Omega(n)$ translates to combinatorial optimization problems under explorable uncertainty such as computing the shortest path in a graph [Fed+07], knapsack [Mei18] and maximum matchings [Mei18], and prevents us from obtaining non-trivial results for these problems.

Motivated by these simple and strong lower bounds, this thesis asks the question whether worst-case query results and having no additional information on the uncertain input parameters (apart from the uncertainty intervals) is too pessimistic. Going back to the example applications, the quality of links measured using metrics such as throughput and reliability in a wireless network often fluctuates over time within a certain interval. The actual quality of a link can be obtained via a new measurement. If we wish to build a minimum spanning tree using links that currently have the highest link quality and want to minimize the additional measurements needed, we arrive at the MST problem under explorable uncertainty. Assuming that we do not have any additional information on the results of the measurements might however be too pessimistic, as one can for example use machine learning methods to predict the precise link quality based on time-series data of previous link quality measurements [Abd+20].

Since these predictions are not completely accurate, we might still want to execute queries to guarantee that we find an MST with respect to the current link quality, but the additional information might be helpful to select the query strategy. As another example, the known past location and maximum movement speed of a mobile node yields an uncertainty interval that is guaranteed to contain the current location. However, using statistical tools one might be able to predict the most likely true location of the node based on past movement data.

In this thesis, we consider different problems under explorable uncertainty and analyze them in settings that go *beyond the worst-case*. Several frameworks for analyzing algorithms beyond the worst-case have for example been discussed in [Rou20]. Here, we study a *learning-augmented* and a *stochastic* setting for problems under explorable uncertainty.

In the *learning-augmented setting*, we assume access to predictions on the uncertain values. Given the rise of artificial intelligence and machine learning (ML) methods in recent decades, it seems reasonable to expect predictions of good accuracy. However, there is no guarantee and the predictions might be arbitrarily wrong for some instances. Thus, we aim at designing algorithms that achieve an improved competitive ratio if the predictions are of good accuracy, and at the same time match the performance guarantees of algorithms without access to predictions even if the predictions are arbitrarily wrong. This corresponds to a recent research trend that considers the usage of untrusted predictions for online algorithms; we later give an overview of related work in this field. Adopting the notions introduced in [LV21; PSK18], we say that 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 even if the predictions are arbitrarily wrong. Ideally, we want to guarantee a smooth transition between consistency and robustness depending on a *prediction error*. In Chapters 4 and 5, we design learning-augmented algorithms for several problems under explorable uncertainty and analyze them using these notions.

Another model for analyzing problems under explorable uncertainty beyond the worst-case is the *stochastic setting*. Instead of assuming that the query results are returned in a worst-case manner, we assume that they are drawn from (known or unknown) probability distributions over the corresponding intervals. In contrast to the learning-augmented setting, the stochastic information is reliable and algorithms are analyzed with respect to the ratio between the expected query cost of an algorithm and the expected optimal solution cost. In Chapters 3 and 6, we consider problems under explorable uncertainty in different stochastic settings and design algorithms that, in expectation, improve upon adversarial lower bounds.

Overall, the results of this thesis will illustrate that we can improve upon adversarial lower bounds for problems under explorable uncertainty when analyzing them beyond the worst-case. To achieve these improved results, we design several new algorithms using algorithmic techniques and analyses that have not yet been used for problems in this field. We hope that our results and technical contributions lay the foundation for further research on problems under explorable uncertainty beyond the worst-case.

1.1 Outline

This thesis considers several problems under explorable uncertainty in the learning-augmented and stochastic setting. In the following, we give a chapter-wise outline of its concrete contents. While the different chapters are not independent of each other and, in particular, reference each other, they can all also be read standalone.

Chapter 2: Preliminaries and Structural Results

In this chapter, we give general definitions and notation for optimization problems under explorable uncertainty and formally define the concrete problems that we consider throughout this thesis. We define *worst-case (or adversarial) competitive analysis*, which is usually employed to analyze the performance of algorithms for problems under explorable uncertainty. In the process, we survey existing lower bounds on the best possible worst-case performance guarantees that can be achieved for our problems. We briefly discuss the stochastic setting and learning-augmented algorithm design for problems under explorable uncertainty, two types of analysis that go beyond the worst-case and will be used in the consecutive chapters.

In the second part of the chapter, we discuss the *witness set algorithm* [Bru+05], a powerful tool for designing algorithms that achieve good performance guarantees in the worst-case. To this end, we summarize existing results that show how to employ this algorithm for the problems considered in this thesis and generalize them to not only minimize the number of queries but also more general query costs. The summary includes several techniques that allow to compare the query cost of an algorithm with the query cost of an optimal solution. While nearly all existing results on optimization under explorable uncertainty use worst-case analysis and the witness set algorithm, we finish the second part of this chapter with a brief discussion on the limits of this algorithm in settings beyond the worst-case.

Finally, we give an overview of previous related work in the field of explorable uncertainty.

Bibliographic remark: This chapter includes some structural results that are based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+23; Erl+20] and joint work with the same group of authors with the addition of E. Bampis and C. Dürr [Bam+21]. Further, the chapter contains some observations from [MS23]. Therefore, some parts correspond to or are identical with [Erl+23; Erl+20; Bam+21; MS23].

Chapter 3: Orienting (Hyper)graphs under Explorable Stochastic Uncertainty

This chapter considers the *hypergraph orientation problem under explorable uncertainty* in the stochastic setting. Given a hypergraph with uncertain vertex weights that follow known probability distributions, we study the problem of querying vertices of minimum total cost until the identity of a vertex with minimum weight can be determined for each hyperedge. Querying a vertex incurs a cost and reveals the precise weight of the vertex, drawn from the given probability distribution. Using stochastic competitive analysis, we compare the expected query cost of an algorithm with the expected cost of an offline optimal query set for the given instance.

For the general problem, we give a polynomial-time $f(\alpha)$-competitive algorithm, where $f(\alpha) \in [1.618 + \epsilon, 2]$ depends on the approximation ratio α for an underlying vertex cover problem. We also show that no algorithm using a similar approach can be better than 1.5-competitive.

Furthermore, we give polynomial-time $4/3$-competitive algorithms for orienting bipartite graphs with arbitrary query costs and for orienting hypergraphs with a single hyperedge and uniform query costs. We complement both of these results with matching lower bounds.

Bibliographic remark: This chapter is mainly based on joint work with E. Bampis, C. Dürr, T. Erlebach, M. de Lima and N. Megow [Bam+21]. Some minor structural results are based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+23; Erl+20]. Therefore, some parts correspond to or are identical with [Erl+23; Bam+21; Erl+20].

Chapter 4: Sorting and Hypergraph Orientation under Uncertainty with Predictions

We consider learning-augmented algorithms for hypergraph orientation under explorable uncertainty and the special case of sorting a set of uncertainty intervals. In the learning-augmented setting, we assume access to untrusted predictions for the uncertain vertex weights. Our algorithms provide improved performance guarantees for accurate predictions while maintaining worst-case guarantees that match the best possible guarantees without access to predictions. For hypergraph orientation, for any integral $\gamma \geq 2$, we give an algorithm that achieves a competitive ratio of $1 + 1/\gamma$ for correct predictions and γ for arbitrarily wrong predictions. For sorting, we achieve an optimal solution for accurate predictions while still being 2-competitive for arbitrarily wrong predictions. These tradeoffs are 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.

Bibliographic remark: This chapter is mainly based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+23] that will also appear in the proceedings of IJCAI 2023. Some results are based on a different joint work with the same group of authors [Erl+22; Erl+20]. Therefore, some parts correspond to or are identical with [Erl+23; Erl+22; Erl+20].

Chapter 5: Learning-Augmented Algorithms for Minimum Spanning Tree with Uncertainty

This chapter studies learning-augmented algorithms for the minimum spanning tree problem under explorable uncertainty, a fundamental combinatorial optimization problem that has been central also to the research area of explorable uncertainty. We are given a (multi)graph with uncertain edge weights that can be revealed via queries. Our aim is to minimize the number of queries necessary to obtain sufficient information for identifying a minimum spanning tree. For all integral $\gamma \geq 2$, we present algorithms that are γ-robust and $(1 + \frac{1}{\gamma})$-consistent and show that this tradeoff is best possible. Furthermore, we argue that the *hop distance*, an error metric introduced in the previous chapter, is a useful measure for the amount of prediction error and design algorithms with performance guarantees that degrade smoothly with the hop distance. Our results demonstrate that access to untrusted predictions can help to circumvent the known lower bound of two, without any degradation of the worst-case ratio. In the process, we provide new structural insights for the minimum spanning tree problem under explorable uncertainty that might be useful in the context of query-based algorithms regardless of predictions.

Bibliographic remark: This chapter is mainly based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+22]. Some results are based on a different joint work with the same group of authors [Erl+23]. Therefore, some parts correspond to or are identical with [Erl+23; Erl+22].

Chapter 6: Set Selection under Explorable Stochastic Uncertainty via Covering Techniques

Finally, we consider the set selection problem under explorable stochastic uncertainty. Given subsets of uncertain weights, we study the problem of identifying the subset of minimum total weight (sum of the uncertain weights contained in the set) by querying as few weights as possible. This *set selection problem* is of intrinsic importance within the field of explorable uncertainty as it implies strong adversarial lower bounds for a wide range of interesting combinatorial problems such as knapsack and matchings [Mei18]. We consider a stochastic problem variant and give algorithms that, in expectation, improve upon these adversarial lower bounds. The key to our results is to prove a strong structural connection to a seemingly unrelated covering problem with uncertainty in the constraints via a linear programming formulation. We exploit this connection to derive an algorithmic framework that can be used to solve both problems under uncertainty, obtaining nearly tight bounds on the competitive ratio. This is the first non-trivial stochastic result concerning the sum of unknown weights without further structure known for the set.

Bibliographic remark: This chapter is mainly based on joint work with N. Megow [MS23]. Therefore, some parts correspond to or are identical with [MS23].

Chapter 2

Preliminaries and Structural Results

In this chapter, we give general definitions and notation for optimization problems under explorable uncertainty and formally define the concrete problems that we consider throughout this thesis. We define *worst-case (or adversarial) competitive analysis*, which is usually employed to analyze the performance of algorithms for problems under explorable uncertainty. In the process, we survey existing lower bounds on the best possible worst-case performance guarantees that can be achieved for our problems. We briefly discuss the *stochastic setting* and *learning-augmented algorithm design* for problems under explorable uncertainty, two types of analysis that go beyond the worst-case and will be used in the consecutive chapters.

In the second part of this chapter, we discuss the *witness set algorithm* [Bru+05], a powerful tool for designing algorithms that achieve good performance guarantees in the worst-case. To that end, we summarize existing results that show how to employ this algorithm for the problems considered in this thesis and generalize them to not only minimize the number of queries but also more general query costs. The summary includes several techniques that allow to compare the query cost of an algorithm with the query cost of an optimal solution. While nearly all existing results on optimization under explorable uncertainty use worst-case analysis and the witness set algorithm, we finish the second part of this chapter with a brief discussion on the limits of this algorithm in settings beyond the worst-case.

Finally, we give an overview of previous related work in the field of explorable uncertainty.

Bibliographic remark: The first and second section of this chapter include some structural results that are based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+23; Erl+20] and joint work with the same group of authors with the addition of E. Bampis and C. Dürr [Bam+21]. Therefore, some parts correspond to or are identical with [Erl+23; Erl+20; Bam+21].

Contents

2.1 Explorable Uncertainty: Formal Problem Definitions

We start by giving general definitions for problems under explorable uncertainty and introduce the problems considered in this thesis. In explorable uncertainty, we generally consider combinatorial optimization problems with uncertainty in the numeric input parameters. Instead of having access to the precise values of these parameters, we initially only know *uncertainty intervals* that are guaranteed to contain the precise values. The uncertain input parameters can be queried to reveal their precise values, and our goal is to adaptively query parameters until we have sufficient information to optimally solve the underlying optimization problem. In this context, *adaptivity* means that the selection of the next query is allowed to depend on the precise values revealed by previous queries. Each query comes at a cost and our goal is to minimize the total query cost.

2.1.1 Basic Definitions for Problems under Explorable Uncertainty

During the course of this thesis, we use Q to refer to a set of queries or *query set*. Depending on the concrete problem, a query set contains different types of elements. In one problem we will for example query edges of a graph, while in another problem we query vertices.

Let P denote the problem-dependent set of such uncertain and queryable elements. Then, each $e \in P$ has a *precise value or weight* w_e that is initially unknown. Instead, we only know the uncertainty interval I_e that is guaranteed to contain w_e. A query to e reveals the precise weight w_e and, therefore, reduces the uncertainty interval to $I_e = \{w_e\}$. We call an uncertainty interval *trivial* if it only contains the corresponding precise weight. Consequently, uncertainty intervals that contain more than one value are called *non-trivial*.

For all problems considered in this thesis, the uncertainty intervals are either *open*, i.e., $I_e = (L_e, U_e)$, or trivial. This is a standard technical assumption in explorable uncertainty, which we justify in Section 2.2. We call L_e and U_e the *upper and lower limit* of I_e. If $I_e = \{w_e\}$, then $U_e = L_e = w_e$.

A query set $Q \subseteq P$ is *feasible* if querying Q reveals sufficient information to optimally solve the underlying combinatorial optimization problem. The feasibility of a query set is problem dependent and we characterize it for the concrete problems down below. We use $\mathcal{Q}$ to refer to the set of all feasible query sets. Each uncertain element $e \in P$ has a *query cost* $c_e \geq 0$. We denote the cost of a query set Q by $c(Q) = \sum_{e \in Q} c_e$. Our goal is to (adaptively) find a feasible query set Q minimizing $c(Q)$.

If $c_e = c_{e'}$ for all queryiable elements $e, e' \in P$, then we say that the query costs are *uniform*. In that case, minimizing the query cost $c(Q)$ is equivalent to minimizing the number of queries $|Q|$. Otherwise, we speak of arbitrary query costs.

2.1.2 Hypergraph Orientation and Sorting under Explorable Uncertainty

We consider the *hypergraph orientation problem under uncertainty* [Bam+21], which captures several selection and sorting problems.

In this problem, we are given a hypergraph $H = (V, E)$ with uncertain vertex weights and our task is to orient each hyperedge $S \in E$ towards the vertex of minimum precise weight in S. In line with the definitions given above, each vertex $v \in V$ is associated with an *uncertainty interval* $I_v = (L_v, U_v)$ and an, initially unknown, *precise weight* $w_v \in I_v$. See Figure 2.1 for an example instance of the hypergraph orientation problem.

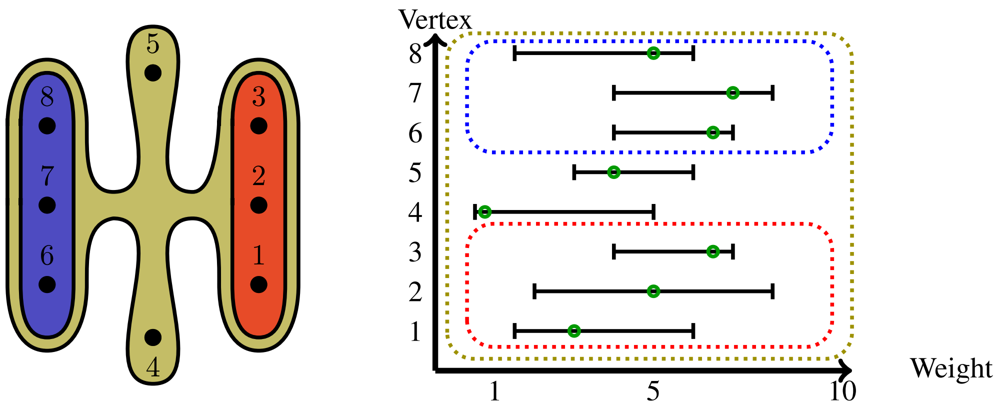


FIGURE 2.1: Example instance for the hypergraph orientation problem. Hypergraph $H = (V, E)$ with the vertices $V = \{1, \dots, 8\}$ and hyperedges $E = \{\{1,2,3,4,5,6,7,8\}, \{6,7,8\}, \{1,2,3\}\}$ (left) and uncertainty intervals for the vertices with (initially uncertain) precise weights indicated by green circles (right).

A query of a vertex v has cost c_v, reveals its precise weight w_v and, thus, reduces its uncertainty interval to $I_v = \{w_v\}$. To orient the hypergraph, we have to adaptively query vertices to learn their precise weights until we have sufficient information to find the orientation of the hyperedges. Thus, a query set $Q \subseteq V$ is called *feasible* if querying Q reveals sufficient information to find the orientation. After querying Q, we must be able to orient each hyperedge S towards a vertex v such that v is of minimum precise weight for the weights w_u of vertices $u \in Q$ and *all possible realizations* $w_u \in I_u$ of the precise weights of unqueried vertices $u \in V \setminus Q$. Note that it suffices to identify the vertex of minimum weight in each hyperedge S and it is *not* required to determine the precise weight of that vertex. Our goal is to (adaptively) find a feasible query set, while minimizing the query costs.

The following lemma fully characterizes feasible query sets. Note that the characterization depends on uncertain precise vertex weights. Thus, an algorithm cannot necessarily use it to decide whether a query set Q is feasible without actually querying Q.

Lemma 2.1.1. *Consider an instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. A query set $Q \subseteq V$ is feasible if and only if it, for each hyperedge S, satisfies at least one of the following conditions:*

1. *Q contains all vertices v in S with non-trivial uncertainty intervals I_v that contain the minimum precise weight $w^* = \min_{u \in S} w_u$ of the vertices in S.*

2. *Let $v \in S$ be a vertex of minimum weight in S, i.e., $w_v = w^* = \min_{u \in S} w_u$. Q contains all vertices $u \in S \setminus \{v\}$ with intervals that intersect I_v and $w_u \geq U_v$ holds for all $u \in S \setminus \{v\}$.*

Proof. Consider an instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. Let $Q \subseteq V$ be a query set.

We prove the lemma by showing for each hyperedge S that we have sufficient information to orient S after querying Q if and only if Q satisfies at least one condition of the lemma. To that end, let $S \in E$ be an arbitrary hyperedge.

First, assume that Q satisfies the first condition of the lemma for hyperedge S and contains all vertices in S with non-trivial uncertainty intervals that contain $w^* = \min_{u \in S} w_u$. Since querying Q reduces the uncertainty intervals of all vertices $v \in Q$ to $I_v = \{w_v\}$ and all vertices in S satisfy $w_v \geq w^*$, the vertices in Q must have a lower limit of at least w^* *after* querying Q. By assumption, each vertex v in $S \setminus Q$ has a lower limit $L_v \geq w^*$ even before querying Q. Thus, after querying Q, all vertices $u \in S$ satisfy $w^* \leq L_u$ and at least one vertex $v \in S$ must satisfy $U_v = L_v = w_v = w^*$. This implies that no vertex in S can have a smaller precise weight than v and, therefore, we have sufficient information to orient S towards v.

Next, assume that Q satisfies the second condition of the lemma for hyperedge S. Then, there is a vertex v of minimum precise weight in S, $w_u \geq U_v$ holds for all $u \in S \setminus \{v\}$, and Q contains all vertices $u \in S \setminus \{v\}$ with $I_u \cap I_v \neq \emptyset$. By assumption, $L_u \geq U_v$ holds for all $u \in (S \setminus \{v\}) \setminus Q$ even before querying Q. Furthermore, since querying Q reduces the intervals I_u of all $u \in Q$ to $I_u = \{w_u\}$ and all $u \in Q$ satisfy $w_u \geq U_v$, we have $L_u = U_u = w_u \geq U_v$ for all $u \in Q$ after querying Q. This implies that $L_u \geq U_v$ holds for all $u \in S \setminus \{v\}$ after querying Q. Thus, no vertex in S can have a smaller weight than v and we have sufficient information to orient S towards v.

Finally, assume that Q does not satisfy any of the conditions of the lemma for some hyperedge S. Since Q does not satisfy the first condition, there must be a vertex $v \in S \setminus Q$ with a non-trivial interval I_v that contains w^*. We distinguish between the cases $w_v = w^*$ and $w_v \neq w^*$.

If $w_v \neq w^*$, then there exists a vertex $u \in S \setminus \{v\}$ with $w^* = w_u \in I_v$. This means that querying Q does not give us sufficient information to orient S as we cannot verify $w^* = w_u \leq w_v$ without querying v.

Next, assume $w_v = w^*$. As Q also does not satisfy the second condition, there either must be a vertex $u \in (S \setminus \{v\}) \setminus Q$ with $I_v \cap I_u \neq \emptyset$ or $w_u \in I_v$ must hold for a vertex $u \in S \setminus \{v\}$. Let u be such a vertex of minimum L_u. In the former case, even after querying Q, we cannot distinguish between realizations of precise weights where $w_v < w_u$ and v is of minimum weight in S and realizations of precise weights where $w_u < w_v$ and u is of minimum weight in S. In the latter case, we cannot verify $w_v \leq w_u$ without querying v. Thus, in both cases, we do not have sufficient information to orient S even after querying Q. This implies that Q is not feasible. □

For the example instance of Figure 2.1, the query set $Q = \{1, 2, 4, 6, 7\}$ satisfies the conditions of the lemma and, thus, is feasible. Since Q contains the vertices 1 and 2, it satisfies the first condition for the red hyperedge, since it contains 4, it satisfies the first condition for the olive hyperedge, and since it contains 6 and 7, it satisfies the second condition for the blue hyperedge.

We also consider the special case where we are given a graph $G = (V, E)$ instead of a hypergraph. We refer to this special case as *graph orientation*. Another important special case of hypergraph orientation is when the input graph is a simple graph that is exactly the interval graph induced by the uncertainty intervals $\mathcal{I} = \{I_v \mid v \in V\}$. 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*.

2.1.3 The Minimum Spanning Tree Problem under Explorable Uncertainty

The next problem we consider is the *minimum spanning tree (MST) problem under explorable uncertainty*. We are given a (multi)graph $G = (V, E)$ with initially uncertain precise edge weights $w_e \in \mathbb{R}_+$ for all $e \in E$. For each edge e, we are given an uncertainty interval I_e that contains w_e and is either open or trivial, i.e., $I_e = (L_e, U_e)$ or $I_e = \{w_e\}$. A *query* of edge e has cost c_e, reveals the precise weight w_e and, thus, reduces the corresponding uncertainty interval to $I_e = \{w_e\}$.

The task is to determine a minimum spanning tree with respect to the initially uncertain precise weights w_e. A *spanning tree* of G is a subgraph of G that contains no cycles and connects all the vertices of G, and a *minimum spanning tree* is a spanning tree of G with minimum total edge weight. During the course of this thesis, we characterize a spanning

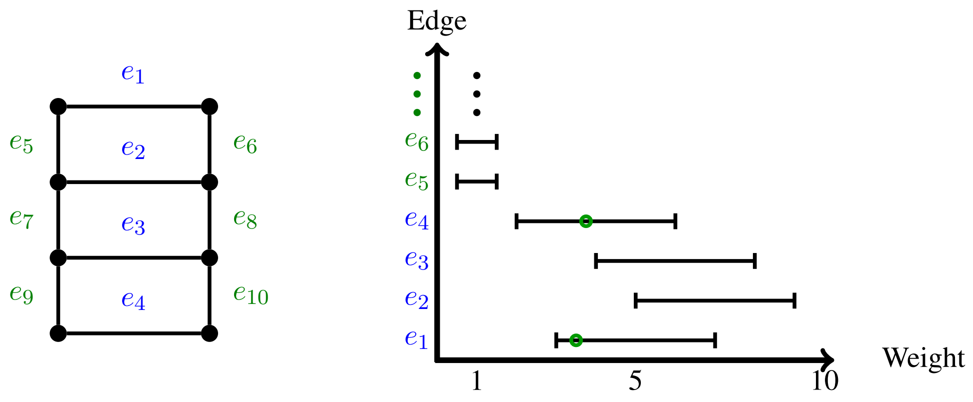


FIGURE 2.2: Example instance for the MST problem under explorable uncertainty. Graph (left) and uncertainty intervals with precise weights for the edges e_1 and e_4 illustrated by green circles (right).

tree by its edge set and say that $T \subseteq E$ is a spanning tree if the subgraph $G' = (V, T)$ is a spanning tree. For the minimum spanning tree problem, a query set is called *feasible* if it reveals sufficient information to identify an MST. Our goal is to adaptively find a feasible query set of minimum cost.

To more formally define feasible and optimal query sets, we say that a query set $Q \subseteq E$ is *feasible* if there exists a set of edges $T \subseteq E$ such that T is an MST for the precise weights w_e of all $e \in Q$ and *every possible* combination of edge weights in I_e for the unqueried edges $e \in E \setminus Q$. That is, querying a feasible query set Q must give us sufficient information to identify a spanning tree T that is an MST for the precise weights no matter what the precise weights of the unqueried edges $E \setminus Q$ actually are. Note that it is sufficient to identify an MST and *not* required to compute the precise weight of that MST. We call a feasible query set Q *optimal* if it has minimum cost $c(Q)$ among all feasible query sets.

Figure 2.2 shows an example instance for the MST problem under explorable uncertainty. We can observe that the query set $Q = \{e_1, e_4\}$ is feasible for this instance and that the edges $\{e_1, e_5, \ldots, e_{10}\}$ induce an MST for the instance. No matter where the precise weight of an edge e_i with $i \in \{5, \ldots, 10\}$ lies within its respective uncertainty interval, it is uniquely minimal in a cut and, therefore, has to be part of *every* MST. Thus, we can conclude that the edges $\{e_5, \ldots, e_{10}\}$ are part of the MST without executing any queries. Besides those edges, an edge in $\{e_1, \ldots, e_4\}$ of minimum precise weight among those edges has to be part of the MST to ensure connectivity. Since the uncertainty intervals of those edges intersect, we have to execute queries to find such an edge. For this example, querying $Q = \{e_1, e_4\}$ suffices to prove that e_1 has minimum weight in $\{e_1, \ldots, e_4\}$ independent of the precise weights of the unqueried edges e_2 and e_3.

2.1.4 Set Selection under Explorable Uncertainty

The final problem we consider is the *set selection problem* under explorable uncertainty. We are given a set of n uncertain weights represented by uncertainty intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ and a family of m sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ with $S \subseteq \mathcal{I}$ for all $S \in \mathcal{S}$. A weight $w_i \in \mathbb{R}_+$ lies in its uncertainty interval I_i, is initially unknown, and can be revealed via a query at cost c_i. The weight of a subset $S \in \mathcal{S}$ is $w(S) = \sum_{I_i \in S} w_i$ and our goal is to determine a subset of minimum weight *and* the corresponding weight while minimizing the total query cost. Note that the latter requirement is in contrast to the previous problems

We again assume that each interval $I_i \in \mathcal{I}$ is either open (*non-trivial*) or *trivial*, i.e., $I_i = (L_i, U_i)$ or $I_i = \{w_i\}$. Based on the uncertainty intervals, we can define intervals

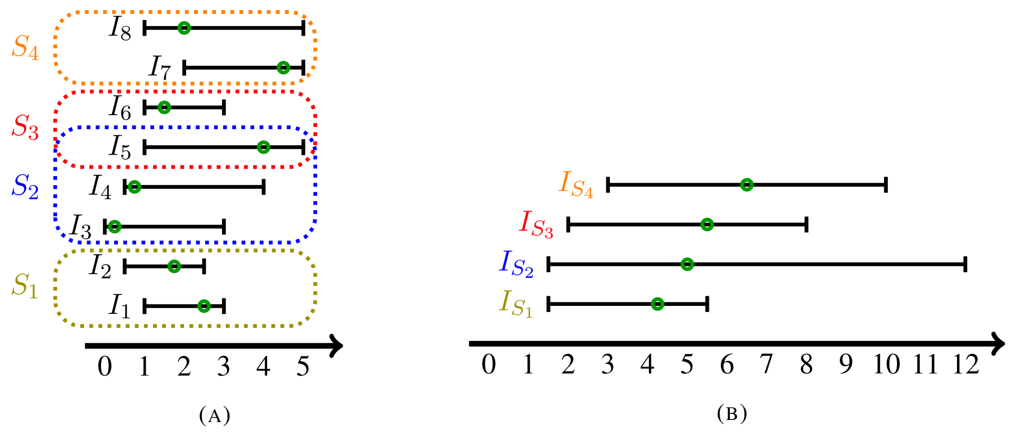


FIGURE 2.3: Example instance for set selection under explorable uncertainty with intervals $\mathcal{I} = \{I_1, I_2, \dots, I_8\}$ and sets $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$ with $S_1 = \{I_1, I_2\}$, $S_2 = \{I_3, I_4, I_5\}$, $S_3 = \{I_4, I_5, I_6\}$ and $S_4 = \{I_7, I_8\}$ (A) and the corresponding uncertainty intervals I_{S_j} for the sets S_j with $j \in \{1, \dots, 4\}$ (B).

I_S for the sets $S \in \mathcal{S}$. To that end, we define the *lower limit* $L_S = \sum_{I_i \in S} L_i$ and the *upper limit* $U_S = \sum_{I_i \in S} U_i$. If S contains only trivial uncertainty intervals, then we define $I_S = [L_S, U_S] = \{w(S)\}$ and call I_S *trivial*. Otherwise, we define $I_S = (L_S, U_S)$. Clearly, the weight $w(S)$ of a set $S \in \mathcal{S}$ is contained in the interval I_S, i.e., $w(S) \in I_S$. We call I_S the *uncertainty interval of set* S. See Figure 2.3 for an example instance including the uncertainty intervals of the sets.

Since the intervals I_S of the sets $S \in \mathcal{S}$ can intersect, we might have to execute queries to determine the set of minimum weight. A query to an interval I_i reveals the precise weight w_i and, thus, replaces both, L_i and U_i, with w_i. This also gives us new information about the intervals I_S of sets S with $I_i \in S$. In a sense, a query to an $I_i \in S$ reduces the range (L_S, U_S) in which $w(S)$ might be by increasing L_S by $w_i - L_i$ and decreasing U_S by $U_i - w_i$; see Figure 2.4 for an illustration. We use L_S and U_S to refer to the initial limits and $L_S(Q)$ and $U_S(Q)$ to denote the lower and upper limits of a set $S \in \mathcal{S}$ *after* querying a set of intervals $Q \subseteq \mathcal{I}$. Let $w^* = \min_{S \in \mathcal{S}} w(S)$ be the initially uncertain minimum set weight. To solve the problem, we have to adaptively query a set of intervals Q until $U_{S^*}(Q) = L_{S^*}(Q) = w^*$ holds for some $S^* \in \mathcal{S}$ and $L_S(Q) \geq w^*$ holds for all $S \in \mathcal{S}$. Only then, we know for sure that w^* is indeed the minimum set value and that S^* achieves this value. We say that a set Q is *feasible* if it satisfies these conditions (cf. Chapter 6 for a full characterization of feasible query sets). Our goal is again to adaptively find a feasible query set Q of minimum cost $c(Q)$.

For the example of Figure 2.3, the query set $Q = \{I_1, I_2, I_5, I_7\}$ is feasible; cf. Figure 2.5 for an illustration of the instance *after* querying Q. After querying Q, we have $U_{S_1}(Q) = L_{S_1}(Q) = w(S_1)$ and $L_{S_i}(Q) > w(S_1)$ for all $i \in \{2, 3, 4\}$. Thus, querying Q proves that set S_1 has a weight of $w(S_1)$ and that no other set has a smaller weight, which implies that Q is feasible.

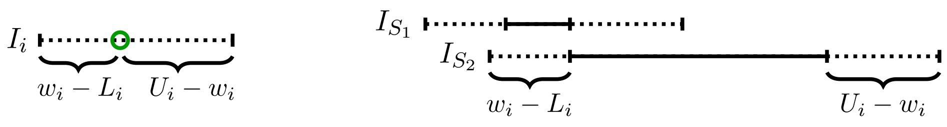


FIGURE 2.4: Example of how a query to an interval I_i changes the intervals of two sets S_1, S_2 with $I_i \in S_1 \cap S_2$ in the set selection problem under explorable uncertainty.

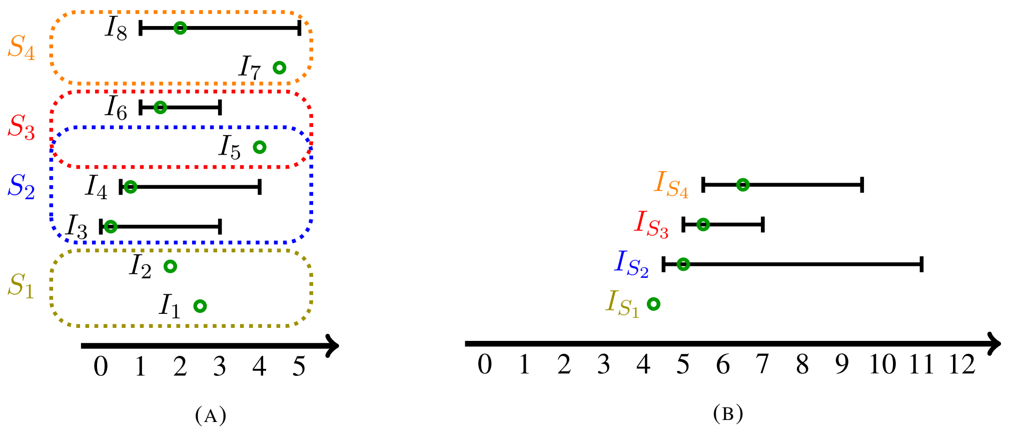


FIGURE 2.5: Instance of Figure 2.3 after querying $Q = \{I_1, I_2, I_5, I_7\}$: (A) Updated uncertainty intervals $\mathcal{I}$ and (B) updated set uncertainty intervals.

2.2 Competitive Analysis

During the course of this thesis, we design algorithms (or *query policies*) that adaptively query feasible query sets for the problems introduced above. In this section, we discuss how to analyze the performance of such algorithms.

For all problems considered in this thesis, there exist input instances that require queries to *all* queryable elements in order to solve the underlying problems. Thus, instead of giving absolute performance bounds, we analyze our algorithms in an instance-dependent manner by employing *competitive analysis*.

In competitive analysis, we compare the query cost of our algorithm against the query cost of an *offline* optimal solution. Here, an *offline* optimal solution refers to the optimal feasible query set that can be computed by an algorithm that knows all query results up-front before actually executing any queries. In a sense, the offline optimal solution is the cheapest certificate that someone *without* up-front access to the query results can query to obtain enough information to verify an optimal solution for the given instance of the underlying optimization problem. Since an *online* algorithm that does not know the query results up-front operates with less information on the input than the offline optimal solution, it cannot necessarily match the cost of the offline solution, even if it has unlimited running time and space.

In the following, we define different types of competitive analysis. We start with *adversarial* or *worst-case* competitive analysis and survey existing lower bounds on the best worst-case performance guarantees achievable for our problems. While most existing results on problems under explorable uncertainty employ this form of analysis, the main contribution of this thesis is to go beyond the worst-case. To this end, we also define *learning-augmented* and *stochastic* competitive analysis.

2.2.1 Worst-Case Analysis and Lower Bounds

In the *adversarial* or *worst-case* setting, we assume that query results are revealed in a worst-case manner, i.e., we assume that an *adversary* chooses the query results in such a way that leads to the worst performance of our algorithm. The following definition formalizes this setting. Note that a problem instance J refers to an input instance *including* fixed but initially unknown precise weights.

Definition 2.2.1 (Adversarial competitive ratio)**.** *Consider a problem under explorable uncertainty and let* ALG *be a fixed deterministic algorithm for this problem. For an instance J of the problem let* $\mathrm{ALG}(J)$ *denote the query cost needed by* ALG *to solve J and let* $\mathrm{OPT}(J)$

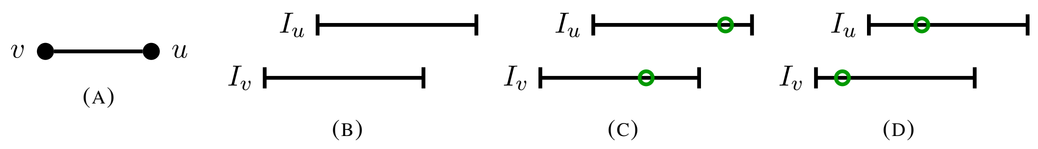


FIGURE 2.6: Lower bound example for the (hyper)graph orientation problem under explorable uncertainty. Shows the input graph (A), the uncertainty intervals (B), and two possible realization of precise vertex weights (C,D).

denote the offline optimal query cost necessary to solve J. The competitive ratio *of* ALG *is*

$$\max_{J \in \mathcal{J}} \frac{\mathrm{ALG}(J)}{\mathrm{OPT}(J)},$$

where $\mathcal{J}$ refers to the set of all instances of the considered problem. We say that ALG *is ρ-competitive if it has a competitive ratio of at most ρ.*

2.2.2 Lower Bounds on the Adversarial Competitive Ratio

To further illustrate adversarial competitive analysis, we review existing lower bounds on the adversarial competitive ratio for the problems considered in this thesis, starting with the hypergraph orientation problem. We will see later in this chapter that all these lower bounds are tight, meaning that there exist algorithms with matching competitive ratios.

Theorem 2.2.2 (Kahan [Kah91])**.** *No deterministic algorithm has a competitive ratio better than* 2 *for the hypergraph orientation problem under explorable uncertainty in the adversarial setting, even for uniform query costs. This lower bound also holds for sorting and graph orientation.*

Proof. Consider the (non-hyper) graph and uncertainty intervals given in Figure 2.6a and Figure 2.6b with uniform query costs, i.e., $c_v = c_u = 1$. Since we are given only a single (non-hyper) edge and the uncertainty intervals of the vertices intersect, the input is a valid instance for hypergraph orientation, graph orientation and sorting under explorable uncertainty.

Every algorithm, including the offline optimum, has to execute queries until it has sufficient information to determine the orientation of the only edge $\{u, v\}$. As the uncertainty intervals of u and v intersect, each such algorithm has to execute at least one query.

Fix a deterministic algorithm ALG. If ALG starts by querying u, then the precise weight of u might be revealed to be contained in the intersection of both uncertainty intervals, i.e., $w_u \in I_v \cap I_u$. This means that ALG still does not have sufficient information to determine the orientation as the still unknown w_v could still be both, larger or smaller, than w_u. This forces ALG to also query v. But then, the precise weight of v might be revealed to be contained in $I_v \setminus I_u$ (cf. Figure 2.6c), which means that the offline optimal solution for the instance queries *only* v. Thus, the algorithm has a query cost of 2 for the instance specified by the Figures 2.6a and 2.6c while the optimal solution has a query cost of only 1.

Symmetrically, if ALG queries v first, then it has a query cost of 2 for the instance specified by the Figures 2.6a and 2.6d while the optimal solution has a query cost of only 1.

Since each deterministic algorithm has to start by querying either v or u, we can conclude that for every deterministic algorithm there exists an instance on which the algorithm executes twice as many queries as the offline optimal solution. This implies an adversarial competitive ratio of at least two. □

Note that the lower bound example of Theorem 2.2.2 essentially boils down to comparing two uncertain weights with intersecting uncertainty intervals. For the minimum spanning tree

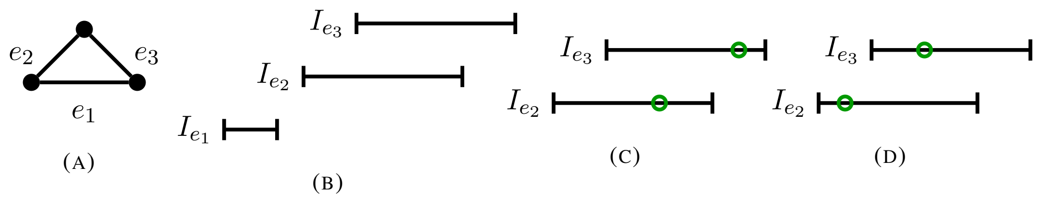


FIGURE 2.7: Lower bound example for the minimum spanning tree problem under explorable uncertainty. Shows the input graph (A), the uncertainty intervals (B), and two possible realization of precise weights of the edges e_2 and e_3 (C,D).

problem, one can prove a lower bound of 2 by constructing a graph with uncertainty intervals that lead to the same situation.

Theorem 2.2.3 (Erlebach et al. [Hof+08]). *No deterministic algorithm has a competitive ratio better than* 2 *for the minimum spanning tree problem under explorable uncertainty in the adversarial setting, even for uniform query costs.*

Proof. Consider the graph and uncertainty intervals given in Figure 2.6a and Figure 2.6b and assume uniform query costs, i.e., $c_{e_1} = c_{e_2} = c_{e_3} = 1$. Independent of where the precise edge weights are in their respective uncertainty intervals, edge e_1 is contained in every MST. Thus, to solve the problem, every deterministic algorithm only has to query until it has sufficient information to decide whether e_2 or e_3 is of larger precise weight.

This leads to exactly the same situation as in the proof of Theorem 2.2.2. If a deterministic algorithm starts by querying e_2, then it has a query cost of 2 for the instance specified by the Figures 2.7a and 2.7c while the offline optimal solution has a query costs of only 1. Symmetrically, if a deterministic algorithm starts by querying e_3, then it has a query cost of 2 for the instance specified by the Figures 2.7a and 2.7d while the offline optimal solution has a query cost of only 1. This implies the theorem. □

While the lower bounds for hypergraph orientation and MST under explorable uncertainty are based on the comparison of two uncertain weights, we can construct a stronger lower bound for set selection under explorable uncertainty by exploiting that we have to compare *sums* of uncertain weights. Erlebach et al. proved a lower bound of $2d$ on the adversarial competitive ratio for the problem variant that does *not* require us to compute the the minimum set weight, where d is the cardinality of the largest given set [EHK16]. A slight modification of their lower bound example allows us to obtain a lower bound of d on the adversarial competitive ratio for our problem variant where we have to also determine the minimum set weight.

Theorem 2.2.4 (Erlebach et al. [EHK16]). *No deterministic algorithm has a competitive ratio better than* d *for the set selection problem under explorable uncertainty in the adversarial setting, even for uniform query costs, where* d *is the cardinality of the largest set in the input.*

Proof. Consider the instance given in Figure 2.8, which consists of the uncertainty intervals $\mathcal{I} = \{I_0, \ldots, I_d\}$ and sets $\mathcal{S} = \{S_1, S_2\}$ with $S_1 = \{I_1, \ldots, I_d\}$ and $S_2 = \{I_0\}$. For $i \in \{1, \ldots, d\}$, the uncertainty intervals are $I_i = (0, 1)$. Interval I_0 is trivial with $I_0 = \{0.65\}$. See Figure 2.8 for an illustration of the instance.

Since S_2 only contains the trivial interval I_0, we already know that $w(S_2) = w_0 = 0.65$ and it only remains to determine whether $w(S_1)$ is smaller than 0.65. Based on the given intervals, we only know that $w(S_1)$ is contained in the interval $I_{S_1} = (L_{S_1}, U_{S_1}) = (\sum_{I_i \in S_1} L_i, \sum_{I_i \in S_1} U_i) = (0, d)$. Recall that a query to an interval $I_i \in S_1$ reveals w_i and, thus, reduces the interval that could potentially contain $w(S_1)$ to $(L_{S_1} + (w_i - L_i), U_{S_1} - (U_i - w_i))$.

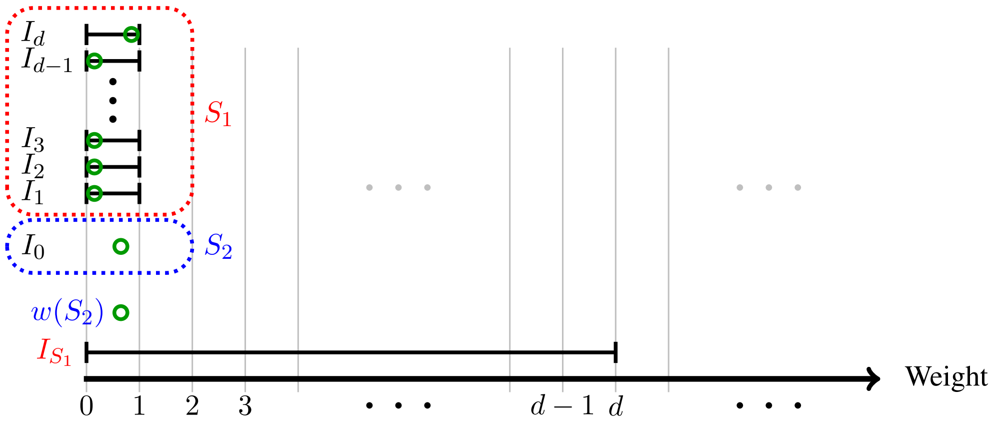


FIGURE 2.8: Lower bound example for the set selection problem under explorable uncertainty. The instance consists of the intervals $\mathcal{I} = \{I_0, \ldots, I_d\}$ and the sets $\mathcal{S} = \{S_1, S_2\}$ with $S_1 = \{I_1, \ldots, I_d\}$, $S_2 = \{I_0\}$, $I_0 = \{0.65\}$ and $I_i = (0, 1)$ for $i \in \{1, \ldots, d\}$. Also shows the interval I_{S_1} for the set S_1.

Fix an arbitrary deterministic algorithm ALG. Since the precise weights w_i are unknown to ALG, it cannot distinguish the intervals $I_1, \ldots, I_d$ and queries these intervals in an arbitrary order until it has sufficient information to decide whether $w(S_1) \geq 0.65$.

Assume without loss of generality that ALG queries the intervals in increasing order of their indices. Then, the precise weights might be revealed as $w_i = \varepsilon$ for all $i \in \{1, \ldots, d-1\}$ for an infinitesimally small $\varepsilon > 0$. But then, even after querying $I_1, \ldots, I_{d-1}$, the value $w(S_1)$ can still be anywhere in the interval $(\varepsilon \cdot (d-1), 1 + \varepsilon \cdot (d-1))$. Thus, ALG can still not decide whether $w(S_1) \geq 0.65$ and has to also query I_d. This leads to a total of d queries by ALG. If this last query reveals $w_d = 1 - \varepsilon$, then querying *only* I_d reduces the interval I_{S_1} that can potentially contain $w(S_1)$ to $(1 - \varepsilon, d - \varepsilon)$ and, therefore, suffices to prove that $w(S_1) > 0.65$. Thus, we get ALG $= d$ and OPT $= 1$, which implies the theorem. See Figure 2.8 for an illustration of the precise weights. □

We can also use adversarial competitive analysis to justify the assumption that the given uncertainty intervals are either open or trivial by giving stronger adversarial lower bounds for the problem variants where the intervals can be closed. Such lower bounds exist for all problems considered in this thesis, except for the sorting problem for which Halldórsson and de Lima [HL21] showed that 2 remains a tight bound on the competitive ratio even for closed uncertainty intervals. The following theorem states the stronger lower bounds for the other problems.

Theorem 2.2.5 (Erlebach et al. [Hof+08; EHK16], Kahan [Kah91])**.** *If the given uncertainty intervals can be closed, then no deterministic algorithm can have an adversarial competitive ratio better than n for the hypergraph orientation, minimum spanning tree or set selection problem under explorable uncertainty. Here, n is the number of given uncertainty intervals. For set selection, this holds even for instances where all sets have cardinality one.*

Proof. First, consider a hypergraph orientation instance with hypergraph $H = (V, E)$, vertices $V = \{v_1, \ldots, v_n\}$, a single hyperedge $S = V$, closed uncertainty intervals $I_{v_i} = [0, 10]$ for all $i \in \{1, \ldots, n\}$ and uniform query costs. The goal is to adaptively query vertices until we have sufficient information to orient the only hyperedge towards a vertex of minimum precise weight. Since the uncertainty intervals are closed, querying a vertex v_i with $w_{v_i} = 0$ immediately solves the instance as no vertex v_j with $i \neq j$ can have a smaller weight than 0.

A deterministic algorithm ALG cannot distinguish between the intervals of the different vertices and, therefore, queries the vertices in an arbitrary order until the instance is solved.

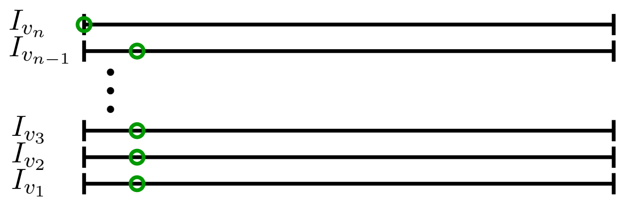


FIGURE 2.9: Uncertainty intervals and precise weights for the lower bound example of Theorem 2.2.5 for the hypergraph orientation problem.

Assume without loss of generality that ALG queries the vertices in increasing order of their indices. Then, an adversary can reveal the precise weights of all v_i with $i < n$ as $w_{v_i} = 1$. This means that the instance is not solved even after the algorithm queried the vertices $v_1, \dots, v_{n-1}$ as vertex v_n could still have a weight $w_{v_n} < 1$ or $w_{v_n} > 1$. This forces the algorithm to also query v_n leading to a query cost of $\text{ALG} = n$. If the adversary then reveals $w_{v_n} = 0$, then querying only v_n would have been sufficient to solve the instance, which implies $\text{OPT} = 1$. Thus, $\text{ALG} = n \cdot \text{OPT}$. See Figure 2.9 for an illustration.

For set selection, consider an instance with uncertainty intervals $\mathcal{I} = \{I_1, \dots, I_n\}$ and $I_i = [0, 10]$ for all $I_i \in \mathcal{I}$, and sets $\mathcal{S} = \{S_1, \dots, S_n\}$ with $S_i = \{I_i\}$ for all $i \in \{1, \dots, n\}$. Since all sets have size one, this problem is equivalent to the hypergraph orientation problem with a single hyperedge with the only difference that we require to determine the precise minimum weight of the intervals. For the lower bound instance above however, this does not make a difference and we can prove a lower bound for set selection in exactly the same way.

For the minimum spanning tree problem, consider an instance consisting of a single cycle with edges $\{e_1, \dots, e_n\}$ and closed uncertainty intervals $I_{e_i} = [0, 10]$ for all $i \in \{1, \dots, n\}$. The problem comes down to determining the edge with maximum weight on the cycle. We can prove a lower bound of n in the same way as for hypergraph orientation, but now the adversary reveals the last weight as $w_{e_n} = 10$. See Figure 2.10 for an illustration. □

2.2.3 Competitive Analysis Beyond the Worst-Case

In the previous section, we reviewed lower bound instances for the adversarial competitive ratio of all problems considered in this thesis. All these lower bounds hold for quite simple instances but require a very specific worst-case realization of precise weights. The assumption that the precise weights are realized in such a worst-case manner and that the algorithm has no additional knowledge (apart from the uncertainty intervals) about them, might in many cases be too pessimistic.

For this reason, we consider different models for optimization under explorable uncertainty where we assume access to additional information on the precise weights and analyze our algorithms beyond the worst-case. To that end, we define the stochastic setting and

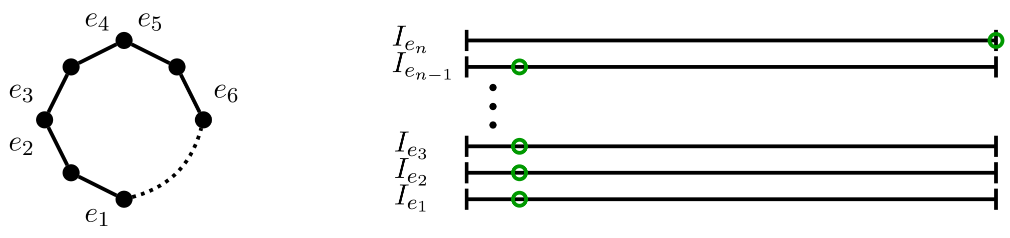


FIGURE 2.10: Graph, uncertainty intervals, and precise weights for the lower bound example of Theorem 2.2.5 for the minimum spanning tree problem.

learning-augmented algorithm design, two less pessimistic models to analyze algorithms for optimization under explorable uncertainty.

The Stochastic Setting Consider any problem under explorable uncertainty with the uncertainty intervals $\mathcal{I} = \{I_1, \dots, I_n\}$. In the stochastic setting, we assume that each interval $I_i \in \mathcal{I}$ is associated with a probability distribution d_i over the uncertainty interval $I_i = (L_i, U_i)$. In contrast to the adversarial setting, we assume that the precise weight w_i of I_i is drawn from I_i according to the distribution d_i. During the course of this thesis, we will assume that the distributions of different intervals are independent of each other and consider scenarios where the distributions are known or unknown to the algorithm.

To analyze algorithms for optimization under explorable uncertainty in the stochastic setting, we use the *stochastic competitive ratio*. Note that in the stochastic setting, a problem instance J is associated with distributions instead of initially unknown precise weights. The offline optimal query cost $\mathrm{OPT}(J)$ necessary to solve J now depends on the distributions d_i instead of the fixed but hidden precise weights w_i. Thus, $\mathrm{OPT}(J)$ is a random variable. Since an adaptive algorithm ALG can decide its next query depending on results of previously executed queries, the execution and cost of ALG on instance J also depends on the distributions and, thus, $\mathrm{ALG}(J)$ is also a random variable. The stochastic competitive ratio considers the expected values of $\mathrm{OPT}(J)$ and $\mathrm{ALG}(J)$.

Definition 2.2.6 (Stochastic competitive ratio)**.** *Consider a problem under stochastic explorable uncertainty and let* ALG *be a fixed deterministic algorithm for this problem. For an instance J of the problem let* $\mathrm{ALG}(J)$ *denote the query cost needed by* ALG *to solve J and let* $\mathrm{OPT}(J)$ *denote the offline optimal query cost necessary to solve J. The* competitive ratio *of* ALG *is*

$$\max_{J \in \mathcal{J}} \frac{\mathbb{E}[\mathrm{ALG}(J)]}{\mathbb{E}[\mathrm{OPT}(J)]},$$

where $\mathcal{J}$ refers to the set of all instances of the considered problem. We say that ALG *is ρ-competitive in the stochastic setting if it has a stochastic competitive ratio of at most ρ.*

In Chapters 3 and 6 we consider the hypergraph orientation and set selection problems under stochastic explorable uncertainty and design algorithms for these problems that improve upon the adversarial lower bounds in terms of their stochastic competitive ratio. The only previous work in the stochastic setting we are aware of is by Chaplick et al. [Cha+21] and considers sorting and other special cases of hypergraph orientation; we will discuss their results in Chapter 3.

Learning-Augmented Algorithm Design In learning-augmented algorithm design, we assume access to *untrusted predictions* on the given problem instance. For a problem under explorable uncertainty with the uncertainty intervals $\mathcal{I} = \{I_1, \dots, I_n\}$, we could for example have access to predictions $\overline{w}_i$ on the precise weights w_i of all intervals $I_i \in \mathcal{I}$. Since these predictions are untrusted, we might have $\overline{w}_i \neq w_i$ for some intervals $I_i \in \mathcal{I}$ and, therefore, an algorithm still has to execute queries in order to solve the underlying problem w.r.t. the precise weights. However, adaptive algorithms can use these predictions to decide their next query.

If the predictions are accurate, they can help algorithms to improve upon adversarial lower bounds. In the lower bound instance of Figure 2.6 for example, an accurate prediction on whether the weights w_v or w_u are contained in the intersection of both intervals would help to decide what vertex to query first. Relying too much on the predictions however, could potentially lead to worse competitive ratios than the adversarial lower bounds if the predictions are faulty.

The goal in learning-augmented algorithm design is to give algorithms that perform better than adversarial lower bounds if the predictions are correct, but still achieve good performance guarantees even if the predictions are arbitrarily bad. To formalize these properties, we employ the notions of *α-consistency* and *β-robustness* as introduced in [LV21; PSK18].

Definition 2.2.7 (Consistency and robustness)**.** *Consider a problem under explorable uncertainty with uncertainty intervals $\mathcal{I} = \{I_1, \dots, I_n\}$ and let* ALG *be a fixed deterministic algorithm for this problem with access to untrusted predictions $\overline{w}_i$ on the precise weights w_i for all intervals $I_i \in \mathcal{I}$. We say that* ALG *is α-consistent if it has an adversarial competitive ratio of at most α for completely accurate predictions, i.e., if $w_i = \overline{w}_i$ for all $I_i \in \mathcal{I}$. The algorithm is β-robust if it achieves an adversarial competitive ratio of at most β even for arbitrarily wrong predictions.*

Note that consistency and robustness only formulate the two extreme cases in terms of prediction quality, perfect predictions and arbitrarily wrong predictions. In learning-augmented algorithm design we aim at giving more fine-grained performance guarantees that smoothly transition between consistency and robustness depending on suitably defined prediction errors.

In Chapters 4 and 5, we consider hypergraph orientation and the MST problem under explorable uncertainty with untrusted predictions. We will discuss suitable error measures for these problems, give consistent and robust algorithms with guarantees depending on these errors, and prove bounds on the optimal tradeoff between consistency and robustness. Furthermore, we discuss the learnability of the predictions with respect to different error measures. To our knowledge, there exists no previous work on explorable uncertainty in learning-augmented algorithm design.

2.3 Local Bounds on OPT and the Witness Set Algorithm

After we have seen adversarial lower bounds for our problems in the previous section, we now consider algorithms that match these lower bounds.

To that end, we discuss techniques that allow comparisons between the query costs of an algorithm and an optimal solution. Based on these techniques, we consider the *witness set algorithm* [Bru+05], a powerful framework that is used for the majority of existing results on optimization under explorable uncertainty. While the witness set algorithm in the literature is only stated for uniform query costs, we generalize it to arbitrary query costs by using a local ratio technique. To our knowledge, this generalization is not yet published in the literature. Afterwards, we show how to apply the generalized witness set algorithm to hypergraph orientation and set selection under explorable uncertainty and match the adversarial lower bounds of the previous section. We remark that the application of the witness set algorithm for set selection is largely based on existing work by Erlebach et at. [EHK16] and only generalized to arbitrary weights. The application of the witness set algorithm to hypergraph orientation exploits the observations by Kahan [Kah91] for the problem of orienting a single hyperedge. The witness set framework can also be applied to match the adversarial lower bound of the minimum spanning tree problem [Hof+08]; we discuss this application in Chapter 5.

While nearly all existing results on optimization under explorable uncertainty use worst-case analysis and the witness set algorithm, we finish this section by briefly discussing the limits of the witness set algorithm in settings beyond the worst-case.

2.3.1 Witness Sets and the Witness Set Algorithm

We continue by surveying existing algorithms for our problems under explorable uncertainty. The absolute majority of those algorithms are analyzed in the adversarial setting and we focus

Algorithm 1: Abstract formulation of the witness set algorithm for uniform query costs by Bruce et al. [Bru+05].

```
Input: Instance of a problem under explorable uncertainty with uniform query costs
       and the set of queryable elements $P$.
Output: A feasible query set $Q$ for the given problem instance.
1 $Q \leftarrow \emptyset$;
2 while The problem is not solved yet do
3     Query all elements of a witness set $W \subseteq P \setminus Q$;
4     $Q \leftarrow Q \cup W$;
5 return $Q$;
```

on these algorithms. In particular, we consider the algorithms for our problems that match the adversarial lower bounds of the previous section. All these algorithms are implementations of the *witness set algorithm* as introduced by Bruce et al. [Bru+05], which is based on the concept of *witness sets*.

Definition 2.3.1 (Bruce et al. [Bru+05])**.** *Consider an instance of a problem under explorable uncertainty with the set of queryable elements P. A subset $W \subseteq P$ is a* witness set *if every feasible solution for the given problem instance contains at least one member of W. That is, $|W \cap Q| \geq 1$ for all $Q \in \mathcal{Q}$, where $\mathcal{Q}$ is the set of feasible query sets for the given instance.*

In most existing work on explorable uncertainty with uniform query costs, witness sets are the key concept to compare the queries selected by an algorithm with the queries of an optimal solution as even the optimal solution must contain at least one member of each witness set. The core idea of the witness set algorithm [Bru+05] for uniform query costs is to repeatedly query disjoint witness sets until the given problem is solved. For pseudocode see Algorithm 1. Bruce et al. [Bru+05] showed that querying only small witness sets is sufficient to achieve good competitive ratios, as formalized by the following theorem.

Theorem 2.3.2 (Bruce et al. [Bru+05])**.** *Consider a problem under explorable uncertainty with uniform query costs. If the witness set algorithm can solve every instance of this problem while only querying witness sets of size at most ρ, then the witness set algorithm is ρ-competitive for the problem.*

Proof. Consider an arbitrary problem under explorable uncertainty and an arbitrary instance of this problem with the set of queryable elements P. By assumption of the theorem, the witness set algorithm can solve this instance by querying only witness sets of size at most ρ.

Let k denote the number of iterations of the while-loop that the witness set algorithm executes to solve the problem instance while only querying witness sets of size at most ρ in Line 3. Furthermore, let Q_i with $i \in \{1, \dots, k\}$ denote the set of queries executed by the algorithm in the i'th execution of the while-loop. By definition of Line 3, we have $Q_i \cap Q_{i'} = \emptyset$ for all $i, i' \in \{1, \dots, k\}$ with $i \neq i'$. The assumption of the theorem implies $|Q_i| \leq \rho$ for all $i \in \{1, \dots, k\}$. Thus, the number of queries executed by the algorithm is $\mathrm{ALG} = \sum_{i \in \{1,\dots,k\}} |Q_i| \leq \rho \cdot k$.

Next, consider the set OPT of queries executed by an optimal solution for the problem instance. As all sets Q_i with $i \in \{1, \dots, k\}$ are witness sets, we have $|Q_i \cap \mathrm{OPT}| \geq 1$. Since the sets Q_i are pairwise disjoint, this directly implies $|\mathrm{OPT}| \geq k$.

Thus, we can conclude $\mathrm{ALG} \leq \rho \cdot k \leq \rho \cdot |\mathrm{OPT}|$, which implies the theorem. □

While Bruce et al. [Bru+05] only considered uniform query costs, it is not hart to generalize the witness set algorithm to arbitrary (non-negative) query costs by employing a *local ratio*

Algorithm 2: Abstract formulation of the witness set algorithm for arbitrary query costs.

Input: Instance of a problem under explorable uncertainty with the set of queryable elements P and arbitrary query costs $c_e \geq 0$ for all $e \in P$.

Output: A feasible query set Q for the given problem instance.

```
1 $Q \leftarrow \emptyset$;
2 $c'_e \leftarrow c_e$ for all $e \in P$;
3 while The problem is not solved yet do
4     $W \leftarrow$ witness set with $W \subseteq P \setminus Q$;
5     $\delta \leftarrow \min_{e \in W} c'_e$;
6     $c'_e \leftarrow c'_e - \delta$ for all $e \in W$;
7     Query $W' = \{e \in W \mid c'_e = 0\}$;
8     $Q \leftarrow Q \cup W'$;
9 return $Q$;
```

technique; see [Bar+04] for a survey on the local ratio technique. In contrast to the case of uniform weights, the witness set algorithm for arbitrary query costs cannot necessarily afford to compute and query a complete witness set W in each iteration. This is, because W might contain very expensive queries that are not part of the optimal solution. Executing these expensive queries might already by enough for the algorithm to not be ρ-competitive anymore. Instead, the witness set algorithm for arbitrary query costs (cf. Algorithm 2) computes a witness set W and the minimum query cost δ over the elements of W. Since W is a witness set, even the optimal solution has to invest query costs of at least δ into querying elements of W. The algorithm then reduces the query costs of all members of W by δ and queries only the elements of W whose query costs were reduced to zero (cf. Lines 4 to 8 in Algorithm 2). We show that this adjustment of the witness set algorithm is sufficient to obtain the following generalization of Theorem 2.3.2.

Theorem 2.3.3. *Consider a problem under explorable uncertainty with arbitrary query costs. If the witness set algorithm for arbitrary query costs can solve every instance of this problem while only considering witness sets of size at most ρ in Line 4, then the witness set algorithm is ρ-competitive for the problem.*

Proof. Consider an arbitrary problem under explorable uncertainty and an arbitrary instance of this problem with the set of queryable elements P. By assumption of the theorem, the witness set algorithm for arbitrary query costs only considers witness sets of size at most ρ in Line 4.

Let k denote the number of iterations of the while-loop executed by the algorithm. For each iteration $i \in \{1, \dots, k\}$, let δ_i denote the value δ as computed by the algorithm in Line 5 during iteration i. Furthermore, let $c_i \colon P \to \mathbb{R}_+$ denote the current function c' as used by the algorithm at the *end* of iteration i and let c_0 denote the initially given query costs. Note that the choice of δ in Line 5 ensures that $c_i(e) \geq 0$ for all $e \in P$ and all $i \in \{1, \dots, k\}$. Finally, let OPT_i denote the cost of an optimal solution for the current problem instance at the end of iteration i and query costs c_i. Let $\mathrm{OPT}_0 = \mathrm{OPT}$ denote the cost of an optimal solution for the initially given query costs and the initial instance.

We first lower bound the optimal query cost for the given instance. As the query costs c_i remain non-negative over the execution of the algorithm, we have $\mathrm{OPT}_i \geq 0$. We argue that $\mathrm{OPT}_i \leq \mathrm{OPT}_{i-1} - \delta_i$ holds for all $i \in \{1, \dots, k\}$. This implies $\mathrm{OPT} = \mathrm{OPT}_0 \geq \sum_{i=1}^{k} \delta_i$. To prove that $\mathrm{OPT}_i \leq \mathrm{OPT}_{i-1} - \delta_i$ holds for all $i \in \{1, \dots, k\}$, consider an arbitrary iteration i and let W_i denote the set W as computed by the algorithm in the execution of Line 4 during iteration i. Since W_i is a witness set, every feasible query set, including the optimal solution for query costs c_{i-1}, has to query at least one member of W_i. Thus, by reducing the query cost of each member of W_i by δ_i, we decrease the cost of each feasible solution by at least δ_i. This implies that the optimal solution for the instance at the end of iteration i with query costs c_i is cheaper than the optimal solution for the instance at the end of iteration $i-1$ with query costs c_{i-1} by at least δ_i. Thus, $\mathrm{OPT}_i \leq \mathrm{OPT}_{i-1} - \delta_i$, and we can conclude $\mathrm{OPT} \geq \sum_{i=1}^{k} \delta_i$.

To finish the proof, we show $\mathrm{ALG} \leq \rho \cdot \sum_{i \in \{1,\dots,k\}} \delta_i$ for the query cost ALG of the witness set algorithm. Since the algorithm only queries elements e with $c'_e = 0$, the total query cost of the algorithm is at most $\sum_{e \in P} c_0(e) - c_k(e)$, which is the total reduction of the query costs over the execution of the algorithm. By definition of Lines 5 and 6, this value is exactly $\sum_{i \in \{1,\dots,k\}} |W_i| \cdot \delta_i$. As the sets W_i are witness sets of size at most ρ, we get

$$\mathrm{ALG} \leq \sum_{e \in P} c_0(e) - c_k(e) = \sum_{i \in \{1,\dots,k\}} |W_i| \cdot \delta_i \leq \rho \cdot \sum_{i \in \{1,\dots,k\}} \delta_i \leq \rho \cdot \mathrm{OPT},$$

which implies the theorem. □

2.3.2 Mandatory Elements and Preprocessing Algorithms

An important special case of witness sets are those of cardinality one. We say that a queryable element that comprises a witness set of size one is *mandatory* because every feasible query set *must* contain such elements.

Definition 2.3.4 (Mandatory Elements)**.** *Consider an instance of a problem under explorable uncertainty with the set of queryable elements P. An element $e \in P$ is* mandatory *for the instance if $e \in Q$ holds for every $Q \in \mathcal{Q}$, where $\mathcal{Q}$ is the set of all feasible query sets for the instance.*

Identifying mandatory elements is an important part of many algorithms under explorable uncertainty because they can be queried without ever worsening the competitive ratio of the algorithm. This often happens in a preprocessing step that also transfers the input instance into a certain structure; we will see examples for such preprocessing steps in Chapters 3 to 5.

2.3.3 A Witness Set Algorithm for Hypergraph Orientation

We now consider an implementation of the witness set algorithm for hypergraph orientation under explorable uncertainty. The implementation exploits results and observations by Kahan [Kah91] for the problem of orienting a single hyperedge.

Recall that we are given a hypergraph $H = (V, E)$ and uncertainty intervals I_v for each $v \in V$. Each vertex has an initially unknown precise weight $w_v \in I_v$ that can be revealed via a query at cost c_v and each uncertainty interval I_v is either open or trivial, i.e., $I_v = (L_v, U_v)$ or $I_v = \{w_v\}$. Our goal is to adaptively query vertices until we have sufficient information to orient each hyperedge $S \in E$ towards a vertex of minimum precise weight in S, while minimizing the total query costs.

In order to define the witness set algorithm, we first give characterizations of mandatory elements and witness sets for hypergraph orientation under explorable uncertainty.

Mandatory Vertices We first give a full characterization of mandatory vertices based on the precise weights of the vertices. Since the precise weights are initially unknown, we cannot directly use this characterization to identify mandatory vertices within an algorithm. Therefore, we also give criteria to identify mandatory vertices based only on the structure of the vertex intervals and precise weights revealed by already executed queries.

Lemma 2.3.5. *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. A vertex $v \in V$ with a non-trivial uncertainty interval I_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 a minimum-weight vertex u of S.*

A common proof technique to show that a vertex $v \in V$ is mandatory, is to consider the query set $V \setminus \{v\}$. Showing that querying every vertex except v does not solve the problem implies that v is mandatory. Vice versa, if querying $V \setminus \{v\}$ solves the problem, then v is not mandatory. This proof idea also translates to different problems under explorable uncertainty.

Proof. If v is a minimum-weight vertex of hyperedge S with a non-trivial uncertainty interval I_v that 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 I_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, as querying it proves that all vertices in $S \setminus \{v\}$ have a larger weight than v. 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. □

Explicitly, Lemma 2.3.5 only enables us to identify mandatory vertices given full knowledge of the precise weights, but it also implies criteria to identify *known mandatory* vertices, i.e., vertices that are known to be mandatory given only the intervals, and precise weights revealed by previous queries.

We call a vertex *leftmost* in a hyperedge S if it has an interval of minimum lower limit among the vertices in S. Note that if a leftmost vertex v in S has a trivial uncertainty interval $I_v = \{w_v\}$, then no vertex in S can have a smaller precise weight than v and we therefore already know that we can orient S towards v. We say that a hyperedge is *solved*, if we already have sufficient information to orient the hyperedge. If a hyperedge has a leftmost vertex with a trivial uncertainty interval, then the hyperedge is solved. The following corollary follows directly from Lemma 2.3.5 and gives a characterization of known mandatory vertices.

Corollary 2.3.6. *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. If the interval of a leftmost vertex v in a not yet solved hyperedge S contains the precise weight of another vertex in S, then v is mandatory. In particular, if v is leftmost in (a not yet solved) S and $I_u \subseteq I_v$ for some $u \in S \setminus \{v\}$, then v is mandatory.*

The latter part of the corollary was also directly proven by Chaplick et al. [Cha+21]. Note that executing queries can change which vertices are leftmost in a hyperedge S and can also change the applicability of the corollary. This is, because the definition of leftmost and the corollary depend on the intervals and a query to a vertex v changes the uncertainty interval I_v from (L_v, U_v) to $\{w_v\}$.

Witness Sets We continue by giving a characterization of witness sets. In order to match the adversarial lower bound of 2 (cf. Theorem 2.2.2), we rely on identifying witness sets of size at most 2. If we always can identify a witness set of at most two as long as the instance is not solved yet, then we can implement the witness set algorithm for arbitrary query costs and apply Theorem 2.3.3 with $\rho = 2$. To this end, we can use the following characterization that was originally given by Kahan [Kah91].

Lemma 2.3.7 (Kahan [Kah91]). *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. Let $S \in E$ be a not yet solved hyperedge of H. 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.*

Proof. We show the lemma via proof by contradiction. Assume for the sake of contradiction that $\{u, v\}$ is as described in the lemma but *not* a witness set. Then, by the definition of witness sets (cf. Definition 2.3.1), there must exist a feasible query set $Q \subseteq V$ with $v, u \notin Q$. In particular, the query set $Q = V \setminus \{u, v\}$ must be feasible and, therefore, reveal sufficient information to orient the hyperedge S of the lemma towards a vertex v^* of minimum precise weight in S.

Assume without loss of generality, that v is leftmost in S. Consider the problem instance *after* querying Q. If $w_{v'} \in I_v$ for some $v' \in S \setminus \{v, u\}$, then v would be mandatory by Lemma 2.3.5 and $\{u, v\}$ would clearly be a witness set. Since v is leftmost in S, it therefore must hold $w_{v'} \geq U_v$ for all $v' \in S \setminus \{u, v\}$. Thus, no vertex in $S \setminus \{u, v\}$ can be of minimum precise weight in S.

As $I_v \cap I_u \neq \emptyset$, there exist both, a realization of the precise weights w_v and w_u with $w_v < w_u$ and a realization of precise weights with $w_v > w_u$. In one of those realizations we have to orient S towards v and in the other we have to orient S towards u. Without querying at least one of v and u, we cannot distinguish between those realizations and, therefore, do not have sufficient information to orient S. This implies that Q is not feasible; a contradiction. □

The Witness Set Algorithm Using the characterizations of mandatory vertices and witness sets (Lemmas 2.3.5 and 2.3.7), we can implement the witness set algorithm for hypergraph orientation under explorable uncertainty. Algorithm 3 formalizes this implementation. Lines 5 to 7 and Lines 9 to 13 implement Lines 4 to 8 of the abstract witness set Algorithm 2 using the characterizations of Lemmas 2.3.5 and 2.3.7. When applying the two lemmas, the algorithm always considers the *current instance*. That is, the instance with the uncertainty intervals *after* executing all previous queries. This allows us to exploit Theorem 2.3.3 and prove the following theorem. We remark that this result is also implied by the results in [GSS16], but only indirectly and with a pseudo-polynomial running time.

Theorem 2.3.8. *There exists a 2-competitive algorithm for hypergraph orientation under explorable uncertainty with arbitrary query costs.*

Proof. We prove the theorem by showing that Algorithm 3 is 2-competitive for hypergraph orientation under explorable uncertainty.

Consider an arbitrary instance for hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$, uncertainty intervals I_v for all $v \in V$ and query costs $c_v \geq 0$ for all $v \in V$. To prove that the algorithm indeed solves the given instance, we have to argue that we always can find a mandatory vertex with Lemma 2.3.5 or a witness set of size two with Lemma 2.3.7 as long as the instance is not solved yet. This then implies that the algorithm indeed terminates and, therefore, solves the instance. Assume for the sake of contradiction that we cannot apply Lemmas 2.3.5 and 2.3.7 but the instance is not solved yet. If we can neither apply Lemma 2.3.5 nor Lemma 2.3.7, then this means that the interval I_v of the leftmost vertex v (after executing all previous queries) of each not yet solved hyperedge $S \in E$ does not contain the precise weight w_u of a $u \in S \setminus \{v\}$ that has already been queried and is not intersected by the interval I_u of an $u \in S \setminus \{v\}$ that is still unqueried. Otherwise, we could apply one of the lemmas. But since v is leftmost in S, this means that v must be of minimum precise weight in S. Thus, we already know the orientation and the problem is already solved.

It remains to bound the query cost $c(\text{ALG})$ of Algorithm 3 for the instance. Since the algorithm implements the witness set algorithm and only considers witness sets of size one or two, we can apply Theorem 2.3.3 with $\rho = 2$ to conclude that $c(\text{ALG}) \leq 2 \cdot c(\text{OPT})$, where

Algorithm 3: Witness set algorithm for hypergraph orientation under explorable uncertainty with arbitrary query costs.

```
   Input: Instance of hypergraph orientation under explorable uncertainty: Hypergraph
          $H = (V, E)$ with uncertainty intervals $I_v$ and query costs $c_v \geq 0$ for all
          $v \in V$.
   Output: A feasible query set $Q$ for the given problem instance.
1  $Q \leftarrow \emptyset$;
2  $c'_v \leftarrow c_v$ for all $v \in V$;
3  while The problem is not solved yet do
4      while There exists a vertex $v \in V \setminus Q$ that is mandatory by Corollary 2.3.6 do
5          $c'_v \leftarrow 0$;
6          Query $v$;
7          $Q \leftarrow Q \cup \{v\}$;
8      if The problem is not solved yet then
9          $W \leftarrow$ A witness set $W \subseteq V \setminus Q$ with $|W| = 2$ identified by Lemma 2.3.7;
10         $\delta \leftarrow \min_{v \in W} c'_v$;
11         $c'_v \leftarrow c'_v - \delta$ for all $v \in W$;
12         Query $W' = \{v \in W \mid c'_v = 0\}$;
13         $Q \leftarrow Q \cup W'$;
14 return $Q$;
```

$c(\mathrm{OPT})$ is the optimal query cost for the instance. This implies 2-competitiveness and, thus, the theorem. □

2.3.4 A Witness Set Algorithm for Set Selection

We consider an implementation of the witness set algorithm for the set selection problem under explorable uncertainty. Recall that we are given a set of n uncertain weights w_i represented by uncertainty intervals $\mathcal{I} = \{I_1, \dots, I_n\}$ with $w_i \in I_i$ and a family of m sets $\mathcal{S} = \{S_1, \dots, S_m\}$ with $S \subseteq \mathcal{I}$ for all $S \in \mathcal{S}$. The intervals I_i are either open or trivial, i.e., $I_i = (L_i, U_i)$ or $I_i = \{w_i\}$, and can be queried at cost c_i to reveal the precise weight w_i. The goal is to determine a subset of minimum weight *and* the corresponding weight while minimizing the total query cost, where the weight of a subset S is $w(S) = \sum_{I_i \in S} w_i$.

Erlebach et al. [EHK16] gave a witness set algorithm for the variant of set selection under explorable uncertainty with uniform query costs where the goal is still to determine the subset of minimum weight but it is *not* required to determine the precise weight of that set. Their algorithm achieves a competitive ratio of $2d$ with $d = \max_{S \in \mathcal{S}} |S|$, i.e., d is the cardinality of the largest set. We adjust their algorithm to our problem variant and show that the adjusted algorithm achieves a competitive ratio of d, which matches the lower bound of Theorem 2.2.4. Furthermore, we state the algorithm for arbitrary weights.

Witness Sets We start by giving a characterization of witness sets. Since our problem variant has more constraints on feasible query sets, we can give a slightly simpler characterization than in [EHK16]. Recall that, for each $S \in \mathcal{S}$, the interval $I_S = (L_S, U_S)$ with $L_S = \sum_{I_i \in S} L_i$ and $U_S = \sum_{I_i \in S} U_i$ is guaranteed to contain the precise weight $w(S) = \sum_{I_i \in S} w_i$ of S (with the exception of when S contains only trivial intervals, then we have to define $I_S = [L_S, U_S] = \{w(S)\}$). The following lemma gives a characterization of witness sets based on the intervals I_S. The proof of the lemma heavily exploits that we have to determine the minimum set weight w^* and not only identify the set of minimum weight.

Lemma 2.3.9. *Consider a not yet solved instance of set selection under explorable uncertainty with the set of intervals $\mathcal{I}$ and the family of subsets $\mathcal{S}$. Let S be a set in $\mathcal{S}$ of minimum lower limit, i.e., $L_S = \min_{S' \in \mathcal{S}} L_{S'}$. Then, the non-trivial intervals in S form a witness set.*

Proof. Let $S \in \mathcal{S}$ be as described in the lemma and let w^* be the (potentially still uncertain) minimum set weight, i.e., $w^* = \min_{S \in \mathcal{S}} w(S)$. Furthermore, let $N(S) \subseteq S$ denote the subset of non-trivial intervals in S.

Since the instance is not solved yet, there cannot be a set $S' \in \mathcal{S}$ that only contains trivial intervals and has minimum lower limit in $\mathcal{S}$. If there was such a set, then $w(S') = L_{S'}$ and no other set in $\mathcal{S}$ could have a smaller weight than $w(S')$. This means that we would already know that S' is a set of minimum weight and, as S' only contains trivial intervals, would already have sufficient information to compute $w^* = w(S')$. This is a contradiction to the instance not being solved yet. Thus, such a set cannot exist and we must have $N(S) \neq \emptyset$.

We show the lemma via proof by contradiction. To that end, assume that there exists a feasible query set $Q \subseteq \mathcal{I}$ for the instance with $Q \cap N(S) = \emptyset$. Then, the query set $Q = \mathcal{I} \setminus N(S)$ must also be feasible. Since $Q \cap N(S) = \emptyset$, querying Q does not change the interval I_S of set S. Thus, even after querying Q, we still have that $I_S = (L_S, U_S)$ is non-trivial. As queries can only increase the lower limits of other sets $S' \in \mathcal{S} \setminus \{S\}$, set S remains of minimum lower limit even after querying Q. This means that, even after querying Q, set S can still potentially be the set of minimum weight. Since I_S is still non-trivial, this also implies that we still do not have sufficient information to compute w^*, even after querying Q. Thus, Q is not feasible and we arrive at a contradiction. $\square$

The Witness Set Algorithm Using the characterization of witness sets, we can again give an implementation of the witness set algorithm (cf. Algorithm 4). Since Lemma 2.3.9 implies that there always exists a witness set of size at most $d = \max_{S \in \mathcal{S}} |S|$ as long as the instance is not solved yet, the algorithm clearly solves the given instance. Thus, we can apply Theorem 2.3.3 to conclude the following theorem.

Theorem 2.3.10. *There exists a d-competitive algorithm for set selection under explorable uncertainty, where d is the cardinality of the largest given set.*

2.3.5 The Limits of the Witness Set Algorithm Beyond the Worst-Case

In the previous section, we have seen that implementations of the witness set algorithm achieve the best possible adversarial competitive ratios for set selection and hypergraph orientation (and by extension sorting) under explorable uncertainty. Furthermore, we will see in Chapter 5 that the same holds for the minimum spanning tree problem.

While this shows that the witness set algorithm is a powerful tool to achieve best possible adversarial guarantees, we can also observe that it is tailored towards this kind of worst-case analysis. By considering only witness sets of a cardinality that (at most) matches the corresponding worst-case lower bound, the algorithm will never worsen its competitive ratio beyond the lower bound, but it will also not significantly improve upon the worst-case lower bounds even if we consider the learning-augmented or stochastic setting.

In particular, applying the Theorems 2.3.2 and 2.3.3 as a black box can never lead to an improvement over the adversarial lower bounds because we would have to guarantee that we only query witness sets smaller than the adversarial lower bound. Clearly, such a guarantee would contradict the existence of the corresponding lower bound.

Algorithm 4: Witness set algorithm for set selection under explorable uncertainty with arbitrary query costs.

```
Input: Instance of the set selection under explorable uncertainty with the set of
       uncertainty intervals $\mathcal{I}$, the family of sets $\mathcal{S}$ and query costs $c_i \geq 0$ for all
       $I_i \in \mathcal{I}$.
Output: A feasible query set $Q$ for the given problem instance.
1 $Q \leftarrow \emptyset$;
2 $c'_i \leftarrow c_i$ for all $I_i \in \mathcal{I}$;
3 while The problem is not solved yet do
4     $W \leftarrow$ Witness set with $W \subseteq \mathcal{I} \setminus Q$ identified with Lemma 2.3.9;
5     $\delta \leftarrow \min_{I_i \in W} c'_i$;
6     $c'_i \leftarrow c'_i - \delta$ for all $I_i \in W$;
7     Query $W' = \{I_i \in W \mid c'_i = 0\}$;
8     $Q \leftarrow Q \cup W'$;
9 return $Q$;
```

However, using different techniques, breaking the adversarial lower bounds in settings beyond the worst-case remains possible. Consider for example the lower bound instance of Theorem 2.2.2 for hypergraph orientation. If we consider this instance in the stochastic setting and the probability that one of the weights is contained in the intersection of the intervals is very small, then the witness set algorithm will still have a stochastic competitive ratio close to 2. If we on the other hand would start by querying just one of the vertices and query the second one only if necessary, then the expected cost of our algorithm for the instance would improve significantly.

This small example illustrates that we need new algorithms that do not purely rely on witness sets in order to improve in settings that go beyond the worst-case. During the course of this thesis, we design several such algorithm for all our problems that either significantly extend the witness set framework or use completely novel techniques.

2.4 Related Work

While we will also summarize chapter-specific related work in all following chapters of this thesis, we conclude this chapter by giving a brief general overview of previous work in explorable uncertainty and related fields. The overview ignores the following papers as this thesis is based on them and their results are content of the following chapters: [Erl+22; MS23; Bam+21; Erl+23; Erl+20; MS21].

Previous works in the field of explorable uncertainty assume no knowledge of stochastic information and aim for algorithms that perform well even in a worst-case. The line of research on (adversarial) explorable uncertainty has been initiated by Kahan [Kah91] in the context of selection problems. In particular, he showed for the problem of identifying all maximum elements of a set of uncertain values that querying the intervals in order of non-increasing right endpoints requires at most one more query than the optimal query set. Subsequent work addressed finding the k-th smallest value in a set of uncertainty intervals [GSS16; Fed+03], caching problems in distributed databases [OW00], computing a function value [KT01], sorting [HL21], and classical combinatorial optimization problems, such as shortest path [Fed+07], the knapsack problem [Goe+15], scheduling problems [Dür+20; Ara+18; AE20; AE21; AD23], the MST problem and matroids [Hof+08; EH14; MMS17; FMM20; MS19; MÇ22].

Most related to our work are previous results on the MST problem and sorting with explorable uncertainty. For the MST problem with uncertain edge weights represented by open intervals, a 2-competitive deterministic algorithm was presented and shown to be best possible [Hof+08]. The algorithm is based on the concept of witness sets. The algorithm from [Hof+08] repeatedly identifies a witness set of size 2 that corresponds to two candidates for the maximum-weight edge in a cycle of the given graph, and queries both its elements. It is also known that randomization admits an improved competitive ratio of 1.707 for the MST problem with uncertainty [MMS17]. Both, a deterministic 2-competitive algorithm and a randomized 1.707-competitive algorithm, are known for the more general problem of finding the minimum base in a matroid [EHK16; MMS17], even for the case with non-uniform query costs [MMS17]. If the input graph is a cactus graph, then there is a best possible 1.5-competitive randomized algorithm for arbitrary query costs [MÇ22]. A different variant of the MST problem under explorable uncertainty considers uncertainty in the position of the vertices instead of in the edge weights. In this variant, the vertices are points in the euclidean space but their precise positions are uncertain within given (open or trivial) uncertainty sets. A vertex can be queried to reveal its precise position. The weight of an edge is the euclidean distance between its endpoints and the task is to query vertices until an MST with respect to those weights can be determined. Erlebach et al. [Hof+08] gave a 4-competitive algorithm and a matching lower bound for this problem with uniform query costs. As their algorithm is again a witness set algorithm, it seems likely that the result extends to non-uniform query costs via a local ratio technique. In terms of randomized algorithms, there is a lower bound of 2.5 on the competitive ratio and a 2.5-competitive algorithm for a graph class similar to cactus graphs and uniform query costs [MÇ22].

For the problem of sorting a single set of uncertain values, a 2-competitive algorithm exists (even with arbitrary query costs) and is best possible [HL21]. In the case of uniform query costs, the algorithm simply queries witness sets of size 2. In the case of arbitrary costs, it first queries a minimum-weight vertex cover of the interval graph corresponding to the instance and then executes any remaining queries that are still necessary. For uniform query costs, the competitive ratio can be improved to 1.5 using randomization [HL21].

While most of the works mentioned above consider the same objective as the problems considered in this thesis, i.e., to minimize the query costs, some of the scheduling problems [Dür+20; AE20; AE21; AD23] consider a combined objective that takes the query costs *and* the objective of the underlying scheduling problem into account. More precisely, they assume that the processing times of the jobs to be scheduled are uncertain but can be queried. For each job, an upper bound on the processing time is part of the input. If a job is *not* queried, then it has to be processed for as long as the upper bound demands. A query of a job might reduce the processing time of the job but also takes time that contributes to the scheduling objective. Thus, the scheduling objective includes both, the cost incurred by the processing of the jobs *and* the query costs.

While the majority of previous works on explorable uncertainty considers adaptive algorithms and queries that return precise uncertain values, there are some exceptions. Erlebach, Hoffmann and de Lima [EHL23] consider selection problems in a model with limited adaptivity. In their model, queries are executed in *rounds*. In each round, an algorithm can query up-to k elements in parallel and the goal is to minimize the number of rounds. Merino and Soto consider the MST problem in a completely non-adaptive setting [MS19]. Gupta, Sabharwal and Sen [GSS16] as well as Megow, Meißner and Skutella [MMS17] also consider queries that return smaller uncertainty intervals instead of precise values.

The learning-augmented results of this thesis are the first to consider explorable uncertainty in the recently proposed framework of online algorithms using (machine-learned) predictions [MV17; PSK18; LV21]. After work on revenue optimization [MV17] and online caching [LV21], Kumar et al. [PSK18] studied online algorithms with respect to consistency and robustness in the context of classical online problems, ski-rental and non-clairvoyant scheduling. They also studied the performance as a function of the prediction error. This work initiated a vast growing line of research. Studied problems include rent-or-buy problems [PSK18; GP19; WZ20], revenue optimization [MV17], scheduling and bin packing [PSK18; Ang+20; Mit20; Lat+20; ALT21; ALT22; LM22; Bam+22; LX21], caching and metrical task systems [LV21; Roh20; Ant+23; Wei20], matching [Kum+19; Ant+20], graph problems [Kum+19; LMS22; Ebe+22; APT22] and secretary problems [Düt+21; Ant+20]. We refer to [LM23] for a more complete list of related work in the field of learning-augmented algorithm design. Very recently and in a similar spirit as our work, Lu et al. [Lu+21] studied a generalized sorting problem with additional predictions. Their model strictly differs from ours, as they focus on bounds for the absolute number of pair-wise comparisons whereas we aim for query-competitive algorithms. Overall, learning-augmented online optimization is a highly topical concept which has not yet been studied in the explorable uncertainty model.

The only previous work we are aware of that explicitly considers stochastic explorable uncertainty is by Chaplick, Halldórsson, de Lima and Tonoyan [Cha+21]. Their main result is a dynamic program that minimizes the expected query cost for the problem of sorting a set of uncertain values. The stochastic competitive ratio of that problem remains open. They also consider the problem of finding the minimum in a set of uncertain values with arbitrary query costs and give a randomized 1.5-approximation of the, in expectation, best query strategy.

Also related to stochastic explorable uncertainty is the result by Maehara and Yamaguchi [MY20], who consider packing ILPs with (stochastic) uncertainty in the cost coefficients, which can be queried. They present a framework for solving several problems and bound the absolute number of iterations that it requires to solve them, instead of the competitive ratio. In terms of our problems, this result is related to the set selection problem and we further discuss it in Chapter 6. In another work, Wang et al. [WGW22] consider selection-type problems in a somewhat related model. In contrast to the explorable uncertainty setting, they consider different constraints on the set of queries that, in a way, imply a budget on the number of queries. They solve optimization problems with respect to this budget, which has a very different flavor than our setting of minimizing the number of queries.

While all previous works in explorable uncertainty have uncertain numerical input parameters, there also is a line of related works where the existence of certain entities is uncertain but can be queried [GV06a; Von07; Blu+20; Beh+19; AKL19; BBD22]. Most of these works consider scenarios where edges in a graph exist with a certain probability and a query of an edge reveals whether it actually exists. For example, Behnezhad et al. [BBD22] showed that vertex cover can be approximated within a factor of $(2+\epsilon)$ with only a constant number of queried edges per vertex, where the constant can depend on the probability that an edge exists.

Further works that are different from explorable uncertainty but somewhat related include research on the tradeoff between exploration and exploitation when coping with uncertainty in the input data. Here, stochastic models are often assumed, e.g., in work on multi-armed bandits [Tho33; BC12; GGW11] and the Weitzman's Pandora's box problem [Wei79], and more recently query-variants of combinatorial optimization problems; see, e.g.,[Sin18; Gup+19], and specific problems such as stochastic knapsack [DGV08; Ma18], orienteering [Gup+15; BN15], matching [Che+09; Ban+12], and probing problems [GN13; GNS16].

Chapter 3

Orienting (Hyper)graphs under Explorable Stochastic Uncertainty

This chapter considers the *hypergraph orientation problem under explorable uncertainty* in the stochastic setting. Given a hypergraph with uncertain vertex weights that follow known probability distributions, we study the problem of querying vertices of minimum total cost until the identity of a vertex with minimum weight can be determined for each hyperedge. Querying a vertex incurs a cost and reveals the precise weight of the vertex, drawn from the given probability distribution. Using stochastic competitive analysis, we compare the expected query cost of an algorithm with the expected cost of an offline optimal query set for the given instance.

For the general problem, we give a polynomial-time $f(\alpha)$-competitive algorithm, where $f(\alpha) \in [1.618 + \epsilon, 2]$ depends on the approximation ratio α for an underlying vertex cover problem. We also show that no algorithm using a similar approach can be better than 1.5-competitive.

Furthermore, we give polynomial-time $4/3$-competitive algorithms for orienting bipartite graphs with arbitrary query costs and for orienting hypergraphs with a single hyperedge and uniform query costs. We complement both of these results with matching lower bounds.

Bibliographic remark: This chapter is mainly based on joint work with E. Bampis, C. Dürr, T. Erlebach, M. de Lima and N. Megow [Bam+21]. Some minor structural results are based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+23; Erl+20]. Therefore, some parts correspond to or are identical with [Erl+23; Bam+21; Erl+20].

Contents

3.1 Introduction

In this chapter, we consider the hypergraph orientation problem under explorable uncertainty in the stochastic setting. Recall that we are given a hypergraph $H = (V, E)$ with uncertain vertex weights and our task is to orient each hyperedge $S \in E$ towards a vertex of minimum precise weight in S. Each vertex $v \in V$ has an initially uncertain weight w_v and is associated with an uncertainty interval I_v that is either open or trivial, i.e., $I_v = (L_v, U_v)$ or $I_v = \{w_v\}$. We call L_v and U_v the *lower and upper limit* of I_v. If I_v is trivial, then we define $L_v = U_v = w_v$. The precise weight w_v of a vertex v can be revealed by a query at cost $c_v \geq 0$, and our goal is to adaptively query vertices until we have sufficient information to orient all hyperedges. See Section 2.1.2 for an example problem instance.

In the stochastic setting, each vertex v additionally has a known continuous probability distribution[1] d_v over the uncertainty interval $I_v = (L_v, U_v)$, and the precise weight w_v of v is drawn independently from d_v. We assume that I_v is the minimal interval that contains the support of d_v, i.e., L_v is the largest value satisfying $\mathbb{P}[w_v \leq L_v] = 0$ and U_v is the smallest value satisfying $\mathbb{P}[w_v \geq U_v] = 0$. An 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 $Q \subseteq V$ of queries is called *feasible* if querying Q reveals sufficient information to determine the orientation of each hyperedge. If all vertices have the same query cost, we say that the query costs are *uniform* and assume w.l.o.g. that $c_v = 1$ for all $v \in V$. Otherwise, we speak of *arbitrary query costs*. For $S \subseteq V$, we define the query cost as $c(S) = \sum_{v \in S} c_v$. The objective of an algorithm is to minimize the expected cost of the queries it makes.

We analyze our algorithms in terms of their *stochastic competitive ratio* (see also Definition 2.2.6 in Chapter 2): For a problem instance J, let $\mathrm{ALG}(J)$ denote the query cost required by a deterministic algorithm ALG to solve J and let $\mathrm{OPT}(J)$ denote the minimal query cost required to solve J. For a fixed realization of vertex weights, $\mathrm{OPT}(J)$ refers to the cost of an optimal feasible query set that can be computed by an algorithm that knows all query results of the realization up-front before actually executing any queries but still has to compute a feasible query set. Since the precise weights are drawn from probability distributions, $\mathrm{ALG}(J)$ and $\mathrm{OPT}(J)$ are random variables with the expected values $\mathbb{E}[\mathrm{ALG}(J)]$ and $\mathbb{E}[\mathrm{OPT}(J)]$. The stochastic competitive ratio is defined as

$$\max_{J \in \mathcal{J}} \frac{\mathbb{E}[\mathrm{ALG}(J)]}{\mathbb{E}[\mathrm{OPT}(J)]},$$

[1]We assume the distribution is given in such a way that $\mathbb{P}[w_v \in (a, b)]$ can be computed in polynomial time for every $v \in V, a, b \in \mathbb{R}$. For all our algorithms it suffices to be given a probability matrix: rows correspond to vertices v, columns to elementary intervals (t_i, t_{i+1}), and entries to $\mathbb{P}[w_v \in (t_i, t_{i+1})]$, where $t_1, \dots, t_{2|V|}$ represent the sorted elements of $\{L_v, U_v | v \in V\}$.

where $\mathcal{J}$ is the set of all problem instances. We say that an algorithm is *ρ-competitive* if it has a stochastic competitive ratio of at most ρ. During this chapter, all competitive ratios refer to the stochastic setting unless explicitly stated otherwise.

Besides the general hypergraph orientation problem, we also consider the special case where we are given a graph $G = (V, E)$ instead of a hypergraph, called the *graph orientation* problem. If G is the interval graph defined by $\mathcal{I} = \{I_v \mid v \in V\}$, i.e., $\{u, v\} \in E$ if and only if $I_u \cap I_v \neq \emptyset$, then we speak of the *sorting* problem as orienting the graph corresponds to sorting the vertices by their precise weights.

3.1.1 Our Results

Our first main result (Section 3.3) is an algorithm for the graph orientation problem with competitive ratio $\frac{1}{2}(\alpha + \sqrt{8 - \alpha(4 - \alpha)})$, assuming we have an α-approximation for the vertex cover problem (which we need to solve on an induced subgraph of the given graph). This factor is always between $\phi \approx 1.618$ (for $\alpha = 1$), and 2 (for $\alpha = 2$). For $\alpha < 2$, this is an improvement compared to the adversarial lower bound of 2 (cf. Theorem 2.2.2 in Chapter 2). The algorithm has a preprocessing phase with two steps. First, we compute the probability that a vertex is *mandatory*, i.e., that it has to be queried by every algorithm in order to solve the instance. We query all vertices with a mandatory probability over a certain threshold. The second step uses an LP relaxation of the vertex cover problem to select some further vertices to query. Next, we compute an α-approximation of the vertex cover on a subgraph induced by the preprocessing and query the vertices in the resulting solution. The algorithm finishes with a postprocessing that only queries mandatory vertices. For the analysis, we show two main facts: (1) the expected optimal solution can be bounded by the expected optimal solutions for the subproblems induced by a partition of the vertices; (2) for the subproblem on which we compute a vertex cover in case of uniform query costs, the expected optimal solution can be bounded by applying the Kőnig-Egerváry theorem [Sch03] on a particular bipartite graph. When given arbitrary query costs, we utilize a technique of *splitting* the vertices in order to obtain a collection of disjoint stars with obvious vertex covers that imply a bound on the expected optimum.

We further show how to generalize the algorithm to hypergraphs (Section 3.5). Unfortunately, in this case it is #P-hard to compute the probability of a vertex being mandatory, but we can approximate the probability via sampling. This yields a randomized algorithm that attains, with high probability, a competitive ratio arbitrarily close to the expression given above for graphs. To generalize the previous approach, we need to solve the vertex cover problem on an induced subgraph of an auxiliary graph that contains, for each hyperedge of the given hypergraph, all edges between the vertex with the leftmost interval and the vertices whose intervals intersect that interval.

We also consider a natural alternative algorithm (Section 3.6) that starts with a particular vertex cover solution and then adaptively queries remaining vertices. We prove a competitive ratio of $4/3$ on special cases, namely, for bipartite graphs with arbitrary costs and for a single hyperedge with uniform costs, and complement these results with matching lower bounds.

All our algorithms first non-adaptively query a vertex cover of an auxiliary graph and afterwards adaptively query carefully selected remaining vertices until the instance is solved. We call such algorithms *vertex cover-based*. We prove that no vertex cover-based algorithm is better than 1.5-competitive for the general hypergraph orientation problem. Furthermore, we study how much adaptivity in the query strategy is actually necessary to achieve good competitive ratios. While our results for graph orientation require only two non-adaptive query rounds, we show that more than two rounds are necessary to achieve a competitive ratio better than $\log n$ for hypergraph orientation.

3.1.2 Related Work

Graph orientation problems are fundamental in the areas of graph theory and combinatorial optimization. Usually, graph orientation refers to the task of giving an orientation to edges in an undirected graph such that some given requirement is met. Different types of requirements have been investigated. While Robbins [Rob39] initiated research on connectivity and reachability requirements already in the 1930s, most work is concerned with degree-constraints; cf. overviews given by Schrijver [Sch03, Chap. 61] and Frank [Fra11, Chap. 9].

Our requirement, orienting each edge towards its vertex of minimum weight, becomes non-trivial and challenging when there is uncertainty in the vertex weights. While there are different ways of modeling uncertainty in the input data, the model of explorable uncertainty was introduced by Kahan [Kah91]. He considers the task of identifying the minimum element in a set of uncertainty intervals, which is equivalent to orienting a single hyperedge. Unlike in our model, no distributional information is known, and an adversary can choose weights in a worst-case manner from the intervals. Kahan [Kah91] showed that querying the intervals in order of non-decreasing left endpoints requires at most one more query than the optimal query set, thus giving a competitive ratio of 2. Further, he showed that this is best possible in the adversarial model.

Subsequent work addresses finding the k-th smallest value in a set of uncertainty intervals [GSS16; Fed+03], caching problems [OW00], computing a function value [KT01], and classical combinatorial optimization problems, such as shortest path [Fed+07], knapsack [Goe+15], scheduling problems [Dür+20; Ara+18; AE20], minimum spanning tree and matroids [Hof+08; EH14; MMS17; FMM20; MS19; Erl+22]. Recent work on sorting elements of a single or multiple non-disjoint sets is particularly relevant as it is a special case of the graph orientation problem [Erl+23; HL21]. For sorting a single set in the adversarial explorable uncertainty model, there is a 2-competitive algorithm and it is best possible, even for arbitrary query costs [HL21]. The adversarial competitive ratio can be improved to 1.5 for uniform query cost by using randomization [HL21]. Algorithms with limited adaptivity have been proposed in [EHL23].

Although the adversarial model is arguably pessimistic and real-world applications often come with some distributional information, surprisingly little is known on stochastic variants of explorable uncertainty. The only previous work we are aware of is by Chaplick et al. [Cha+21], in which they studied stochastic uncertainty for the problem of sorting a given set of uncertain values, and for the problem of determining the minimum element in a single set of uncertain values. They showed that the optimal decision tree (i.e., an algorithm that minimizes the expected query cost among all algorithms) for a given instance of the sorting problem can be computed in polynomial time. The competitive ratio of that algorithm remains open. For the minimum problem, they leave open whether an optimal decision tree can be determined in polynomial time, but give a 1.5-competitive algorithm and an algorithm that guarantees a bound slightly smaller than 1.5 on the expectation of the ratio between the query cost of the algorithm and the optimal query cost. Note that the expectation of this ratio is different to the stochastic competitive ratio, which we defined as the ratio of the expectations. The problem of scheduling with testing [LMS19] is also in the spirit of stochastic explorable uncertainty but less relevant here.

3.1.3 Outline

In Section 3.2, we start by introducing preliminary results that are necessary to design and analyze our algorithms, and also give lower bounds on both, the competitive ratio and the (expected) optimal solution cost. Afterwards, we introduce the class of *vertex cover-based* algorithms, which characterizes all algorithms that we use in this chapter and give a lower bound on the competitive ratio for this class of algorithms. In Section 3.3, we give a threshold algorithm that falls into this class of algorithms and analyze it for the graph orientation problem with uniform query costs. The subsequent Section 3.4 generalizes this result to arbitrary query costs. In Section 3.5, we generalize the results of the previous sections to hypergraphs. The key challenge when orienting hypergraphs is that the computation of certain probabilities that are necessary to execute our algorithms becomes $\#P$-complete. We give a sampling algorithm that allows us to approximate the necessary probabilities and use it to design a randomized algorithm that matches the results of the previous sections with a high probability. Finally, in Section 3.6, we give an algorithm that achieves the best expected query cost among all vertex cover-based algorithms. We analyze this algorithm for the special cases of orienting bipartite graphs with arbitrary query costs and orienting a single hyperedge with uniform query costs.

3.2 Preliminaries

The hypergraph orientation problem and the graph orientation problem have already been defined in Section 2.1 and we have seen some structural results for the problem in Section 2.3.3. For the sake of readability, we restate these results here. Afterwards, we extend them to the stochastic setting and discuss how to use them to lower bound the expected optimal query cost for a given instance.

To this end, we briefly restate the concept of mandatory vertices (cf. Sections 2.3.2 and 2.3.3) and show how the probability for a vertex to be mandatory can be computed. Next, we briefly restate the concept of *witness sets* (see also Section 2.3.1) and use it to define the *vertex cover instance* associated with an instance of the hypergraph orientation problem. Such vertex cover instances allow us to give more global lower bounds on the optimal query cost. Exploiting these bounds, we give a full characterization of the expected optimal query cost.

The vertex cover instance is not only useful to lower bound the expected optimum but also allows us to define a class of algorithms, the *vertex cover-based* algorithms. All algorithms considered in this chapter fall into this class.

Finally, we also give a lower bound on the stochastic competitive ratio showing that no algorithm can achieve a ratio better than $\frac{4}{3}$.

Basic definitions. Recall that we measure the performance of an algorithm by comparing the expected cost of the queries it executes to the expected optimal query cost. Formally, given a realization of the precise weights, we call a query set $Q \subseteq V$ *feasible* if querying Q permits one to identify the minimum-weight vertex in every hyperedge. We already proved the following characterization of feasible query sets in Chapter 2.

Lemma 2.1.1. *Consider an instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. A query set $Q \subseteq V$ is feasible if and only if it, for each hyperedge S, satisfies at least one of the following conditions:*

1. *Q contains all vertices v in S with non-trivial uncertainty intervals I_v that contain the minimum precise weight $w^* = \min_{u \in S} w_u$ of the vertices in S.*
2. *Let $v \in S$ be a vertex of minimum weight in S, i.e., $w_v = w^* = \min_{u \in S} w_u$. Q contains all vertices $u \in S \setminus \{v\}$ with intervals that intersect I_v and $w_u \geq U_v$ holds for all $u \in S \setminus \{v\}$.*

An *optimal query set* is a feasible query set of minimum query cost. Since this depends on the probabilistic realization of the precise weights, we denote by $\mathbb{E}[\mathrm{OPT}]$ the expected query cost of an offline optimal query set. Note that each realization of precise weights might have a different optimal query set, so $\mathbb{E}[\mathrm{OPT}]$ is not the expected cost of a certain query set but the expected query cost over the different optimal query sets of all realizations. Similarly, we denote by $\mathbb{E}[\mathrm{ALG}]$ the expected query cost of the query set queried by an algorithm ALG. The supremum of $\mathbb{E}[\mathrm{ALG}]/\,\mathbb{E}[\mathrm{OPT}]$, over all instances of the problem, is called the *competitive ratio* of A. Alternatively, one could compare $\mathbb{E}[\mathrm{ALG}]$ against the cost $\mathbb{E}[A^*]$ of an optimal adaptive algorithm A^*. However, in explorable uncertainty, it is standard to compare against the optimal query set, and, since $\mathbb{E}[\mathrm{OPT}]$ is a lower bound on $\mathbb{E}[A^*]$, all our algorithmic results translate to this alternative setting.

We say that a vertex $v \in S$ is *leftmost* in a hyperedge $S \in E$ if the uncertainty interval I_v of v has minimum lower limit among the intervals of the vertices in S, i.e., $L_v = \min_{u \in S} L_u$. If a hyperedge $S \in E$ contains a leftmost vertex v with a trivial uncertainty interval, then clearly no other vertex in S can have a smaller weight than v and we already know the orientation of S. Thus, we can assume without loss of generality that no hyperedge contains a leftmost vertex with a trivial uncertainty interval, since otherwise we could simply remove the hyperedge. Let v be a leftmost vertex in a hyperedge $S \in E$. Then we can also assume that $I_v \cap I_u \neq \emptyset$ for all $u \in S \setminus \{v\}$, because otherwise the vertex u could be removed from the hyperedge S as we would already know that $w_v < w_u$ and that u cannot be of minimum weight in S. For the special case of graphs, this means that we assume $I_v \cap I_u \neq \emptyset$ for each $\{u, v\} \in E$, since otherwise we could simply remove the edge. The following assumption summarizes these observations.

Assumption 3.2.1. *We assume without loss of generality that all problem instances for hypergraph orientation under explorable uncertainty satisfy the following properties:*

1. *No hyperedge $S \in E$ has a leftmost vertex v with a trivial uncertainty interval I_v.*
2. *If v is leftmost in a hyperedge S, then $I_v \cap I_u \neq \emptyset$ for all $u \in S \setminus \{v\}$.*

3.2.1 Preprocessing, Mandatory Vertices and Mandatory Probabilities

Our algorithms exploit that some vertices are part of *every* feasible query set. If we can identify such a vertex, then we can query it without ever worsening the competitive ratio. A vertex v is called *mandatory* if it belongs to every feasible query set for the given realization. For example, if for some edge $\{u, v\}$, vertex u has already been queried and its value w_u belongs to the non-trivial interval I_v, then v is known to be mandatory as it is impossible to decide whether $w_v < w_u$ or $w_v \geq w_u$ without querying v. We restate the following lemma that fully characterizes mandatory vertices (see Section 2.3.3 for the corresponding proof).

Lemma 2.3.5. *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. A vertex $v \in V$ with a non-trivial uncertainty interval I_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 a minimum-weight vertex u of S.*

The lemma directly implies the following corollary that was also directly proven in [Cha+21, Section 3]. Recall that a hyperedge S is *solved* if we already have sufficient information to orient S. In particular, if there exists a left-most vertex v in a hyperedge S with a trivial uncertainty interval, then S is solved as no vertex in S can have a smaller weight than v.

Corollary 2.3.6. *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. If the interval of a leftmost vertex v in a not yet solved hyperedge S contains the precise weight of another vertex in S, then v is mandatory. In particular, if v is leftmost in (a not yet solved) S and $I_u \subseteq I_v$ for some $u \in S \setminus \{v\}$, then v is mandatory.*

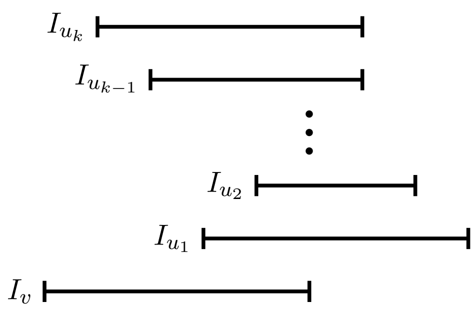


FIGURE 3.1: Illustration of the structure of the uncertainty intervals of the vertices in a hyperedge $S = \{v, u_1, \dots, u_k\}$ of a preprocessed instance.

Thus, if there is a leftmost vertex v of a not yet solved hyperedge $S \in E$ with $I_u \subseteq I_v$ for some $u \in S \setminus \{v\}$, then every algorithm can query such a vertex, without worsening the competitive ratio. Similarly, if a not yet solved hyperedge contains vertices u and v such that v has not been queried yet and is the leftmost vertex, while a query of u has revealed that $w_u \in I_v$, then it follows from Lemma 2.3.5 that v is mandatory for every realization of the unqueried vertices. This also means that if we query a vertex that is mandatory by Corollary 2.3.6, then the information revealed by the query can lead to another vertex becoming mandatory by Corollary 2.3.6. Thus, it can make sense to iteratively and exhaustively apply the corollary until we cannot use it to identify anymore mandatory vertices. In particular, this can be done as a preprocessing step and most of our algorithms assume without loss of generality that the input instance is already preprocessed in this manner.

Definition 3.2.2. *An instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$ is* preprocessed, *if it does not contain vertices that can be identified as mandatory by using Corollary 2.3.6. That is, for every hyperedge $S \in E$, there is no leftmost vertex $v \in S$ with $I_u \subseteq I_v$ for an $u \in S \setminus \{v\}$.*

The assumption of a preprocessed input instance is particularly useful because it allows us to further characterize the structure of the uncertainty intervals as in the following lemma. Figure 3.1 illustrates the structure as described in the lemma for a single hyperedge. We remark that the structure from the lemma holds for an initial preprocessed instance. Once we execute queries, it can change again. However, even then an algorithm can retain the structure by querying mandatory vertices according to Corollary 2.3.6.

Lemma 3.2.3. *Consider a preprocessed instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. Then, each hyperedge $S \in E$ satisfies the following properties:*

1. *S has a unique leftmost vertex v with a non-trivial uncertainty interval.*
2. *$I_u \cap I_v \neq \emptyset$ and $I_u \setminus I_v \neq \emptyset$ for all $u \in S \setminus \{v\}$.*

Proof. Consider an arbitrary hyperedge $S \in E$. By Assumption 3.2.1, S does not have a leftmost vertex with a trivial uncertainty interval. Thus, there must be a leftmost vertex $v \in S$ with a non-trivial uncertainty interval $I_v = (L_v, U_v)$. Assume there is a second leftmost vertex $v' \in S$ with $v \neq v'$. Since v and v' are both leftmost, we have $L_v = L_{v'}$. But then, either $I_v \subseteq I_{v'}$ or $I_{v'} \subseteq I_v$ must hold; a contradiction to the instance being preprocessed. This implies that S satisfies the first property of the lemma.

Next, consider a vertex $u \in S \setminus \{v\}$. Assumption 3.2.1 directly implies $I_u \cap I_v \neq \emptyset$. Since v is the unique leftmost vertex of S, we have $L_v < L_u$. Thus, $I_u \setminus I_v = \emptyset$ would imply $I_u \subseteq I_v$, a contradiction to the instance being preprocessed. □

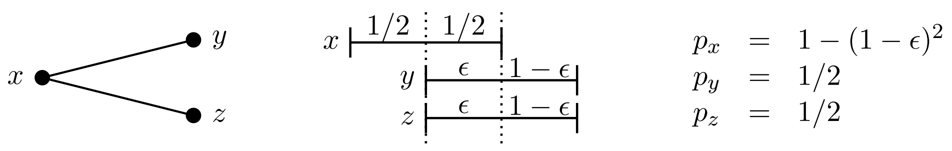


FIGURE 3.2: Instance and mandatory probabilities used in the proof of Theorem 3.2.5.

Corollary 2.3.6 gives us a criterion to identify mandatory vertices based only on the graph and the structure of the intervals and allows us to preprocess instances. The full characterization of mandatory vertices (cf. Lemma 2.3.5) on the other hand, cannot necessarily be directly applied by an algorithm to identify more mandatory vertices, as it depends on the unknown precise vertex weights. Since we consider the stochastic setting however, we might be able to use that characterization to compute the probability that a vertex v is mandatory. For each vertex $v \in V$, we denote this *mandatory probability* by p_v. Querying vertices $v \in V$ with a high mandatory probability p_v will be a key element of our algorithms. For graphs, p_v is easy to compute as, by Lemma 2.3.5, v is mandatory if and only if I_v is initially non-trivial and $w_u \in I_v$ for some neighbor vertex u. Hence, we can observe the following.

Observation 3.2.4. *Consider an instance of graph orientation under stochastic explorable uncertainty with graph $G = (V, E)$ and uncertainty intervals I_v for all $v \in V$. The mandatory probability of a vertex v with a non-trivial uncertainty interval I_v can be written as $p_v = 1 - \prod_{u:\{u,v\}\in E} \mathbb{P}[w_u \notin I_v]$ and can be computed in polynomial time.*

For hypergraphs, however, we show in Section 3.5 that the computation of p_v is #P-hard, even if all hyperedges have size 3. Luckily it is not difficult to get a good estimate of the probabilities to be mandatory for hypergraphs using sampling, as we show in Section 3.5.

3.2.2 Lower Bounds on the Stochastic Competitive Ratio

Using the previously defined mandatory probabilities, we give the following lower bounds on the stochastic competitive ratio.

Theorem 3.2.5. *Every algorithm for the graph orientation problem under explorable stochastic uncertainty has competitive ratio at least $\frac{4}{3}$, even for uniform query costs and even if no restriction on the running time of the algorithm is imposed. For the hypergraph orientation problem under explorable stochastic uncertainty, the lower bound holds even when orienting a single hyperedge. For arbitrary query costs, this lower bound holds even for orienting a single (non-hyper) edge.*

Proof. We first show the lower bound for the graph orientation problem with uniform query costs. Consider three vertices x, y, z, with $I_x = (0, 2)$ and $I_y = I_z = (1, 3)$, and uniform query costs $c_x = c_y = c_z = 1$. The only edges are $\{x, y\}$ and $\{x, z\}$. The probabilities are such that $\mathbb{P}[w_x \in (1, 2)] = \frac{1}{2}$ and $\mathbb{P}[w_y \in (1, 2)] = \mathbb{P}[w_z \in (1, 2)] = \epsilon$, for some $0 < \epsilon \ll \frac{1}{2}$; see Figure 3.2. If $w_x \in (0, 1]$, which happens with probability $\frac{1}{2}$, querying only x gives us sufficient information to orient both edges. If $w_x \in (1, 2)$ and $w_y, w_z \in [2, 3)$, which happens with probability $\frac{1}{2}(1 - \epsilon)^2$, querying y and z gives sufficient information to orient both edges. Otherwise, all three vertices must be queried to identify the orientations of the edges. This gives us

$$\mathbb{E}[\mathrm{OPT}] = \frac{1}{2} \cdot 1 + \frac{1}{2}(1-\epsilon)^2 \cdot 2 + \frac{1}{2}\left(1 - (1-\epsilon)^2\right) \cdot 3 = 2 - \frac{(1-\epsilon)^2}{2},$$

which tends to $\frac{3}{2}$ as ϵ approaches 0. Since y and z are identical and we can assume without loss of generality that an algorithm always queries a vertex first that it knows to be mandatory (if there is such a vertex), we only have three possible algorithms to consider:

1. First query x. If $w_x \in (1,2)$, then query y and z. The expected query cost is 2.
2. First query y. If $w_y \in (1,2)$, then query x, and query z if $w_x \in (1,2)$. If $w_y \in [2,3)$, then query z, and query x if $w_z \in (1,2)$. The expected query cost is $1 + \frac{3}{2}\epsilon + (1-\epsilon)(1+\epsilon)$, which tends to 2 as ϵ approaches 0.
3. First query y. Whatever happens, query x, then query z if $w_x \in (1,2)$. The expected query cost is $\frac{5}{2}$, so this is never better than the previous options.

With either choice (even randomized), the competitive ratio tends to at least $\frac{4}{3}$ as $\epsilon \to 0$.

For the hypergraph orientation problem, consider the instance that consists of a single hyperedge $\{x, y, z\}$ with the uncertainty intervals and distributions as used in the lower bound for graph orientation; see Figure 3.2. For this instance, the expected optimal query cost is the same as in the previous lower bound. Furthermore, it suffices to consider the same three algorithms with the same expected query costs. Thus, the lower bound translates.

For arbitrary query costs, consider a single edge $\{x, y\}$ with costs $c_x = 1$ and $c_y = 2$. We again use the uncertainty intervals and distributions given in Figure 3.2. The expected optimal query cost is

$$\mathbb{E}[\mathrm{OPT}] = \frac{1}{2} \cdot c_x + \frac{1}{2} \cdot (1-\epsilon) \cdot c_y + \frac{1}{2} \cdot (1-(1-\epsilon)) \cdot (c_x + c_y) = \frac{c_x + c_y}{2} + \frac{\epsilon c_x}{2} = 1.5 - \frac{\epsilon}{2}.$$

We only have two decision trees to consider:

1. First query x. If $w_x \in (1,2)$, query y. The expected query cost is $c_x + \frac{1}{2} \cdot c_y = 2$.
2. First query y. If $w_y \in (1,2)$, query x. The expected query cost is $c_y + \epsilon \cdot c_x = 2 + \epsilon$, which tends to 2 as ϵ approaches 0.

With either choice (even randomized), the competitive ratio tends to at least $\frac{4}{3}$ as $\epsilon \to 0$.

□

3.2.3 Witness Sets and the Vertex Cover Instance

Another key concept of our algorithms is to exploit *witness sets* [Bru+05; Hof+08]. Recall that a subset $W \subseteq V$ is a *witness set* if $W \cap Q \neq \emptyset$ for all feasible query sets Q. Witness sets can be used to lower bound the optimal query cost. We restate the following characterization of witness sets for hypergraph orientation (see Section 2.3.3 for a proof).

Lemma 2.3.7 (Kahan [Kah91])**.** *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. Let $S \in E$ be a not yet solved hyperedge of H. 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.*

As we have seen in Section 2.3.3, we can use Lemma 2.3.7 within the witness set algorithm to achieve an adversarial competitive ratio of 2. However, our goal is to achieve better competitive ratios in the stochastic setting and, therefore, we cannot rely on using such techniques as a blackbox.

Instead, we design new techniques that rely on stronger lower bounds on the expected optimal query cost. Lemma 2.3.7 allows us to locally lower bound the optimal query cost for a single hyperedge. To lower bound the optimal query cost for the complete hypergraph, we define the following vertex cover instance.

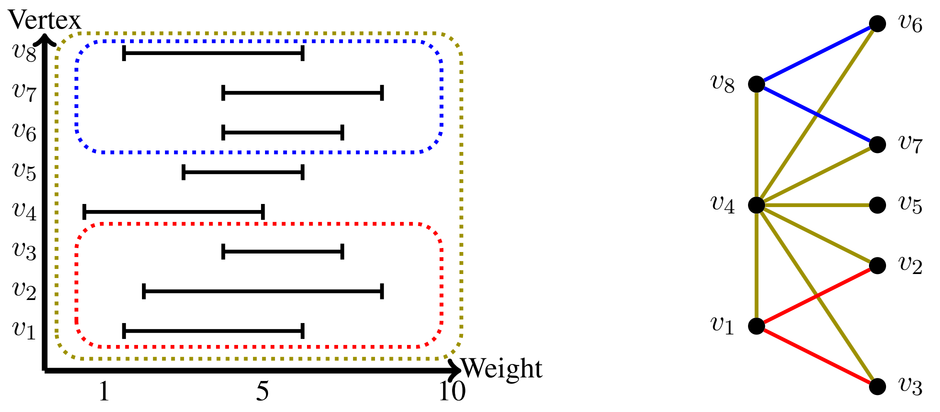


FIGURE 3.3: Example instance for the hypergraph orientation problem and the corresponding vertex cover instance. For a Hypergraph $H = (V, E)$ with the vertices $V = \{v_1, \dots, v_8\}$ and hyperedges $E = \{\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}, \{v_6, v_7, v_8\}, \{v_1, v_2, v_3\}\}$, the figure shows the uncertainty intervals (left) and the corresponding vertex cover instance (right). The differently colored edges indicate the vertex cover instances for the single hyperedges.

Definition 3.2.6. *Consider an instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. The* vertex cover instance *of H is the graph $\bar{G} = (V, \bar{E})$ with $\{v, u\} \in \bar{E}$ if and only if there is a not yet solved hyperedge $S \in E$ such that $v, u \in S$, v is leftmost in S and $I_v \cap I_u \neq \emptyset$. For the special case of a graph G instead of a hypergraph H, it holds that $\bar{G} = G$.*

See Figure 3.3 for an example vertex cover instance. Since each edge of the vertex cover instance $\bar{G}$ is a witness set by Lemma 2.3.7, we can observe that each feasible query set Q is a vertex cover of $\bar{G}$. This leads to the following observation, which will turn out to be very helpful for lower bounding the expected optimal query cost.

Observation 3.2.7. *Given an instance of hypergraph orientation with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$, consider the vertex cover instance $\bar{G} = (V, \bar{E})$. Let VC denote a minimum-weight vertex cover of $\bar{G}$ using the query costs c_v as weights for the vertices. Then, $c(VC) \leq \mathrm{OPT}$ for every realization of precise weights.*

For preprocessed instances, the vertex cover instance has another useful property: If querying a vertex cover of the vertex cover instance does not solve the problem, then we can solve it by only querying vertices that are known to be mandatory by Corollary 2.3.6. The following lemma formalizes this property.

Lemma 3.2.8. *Given a preprocessed instance of hypergraph orientation with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$, let Q be an arbitrary vertex cover of $\bar{G}$. After querying Q, for each hyperedge $S \in E$, we either know the orientation of S or can determine it by exhaustively querying according to Corollary 2.3.6.*

Proof. Consider an arbitrary hyperedge S and let v be leftmost in S. As the instance is preprocessed, the leftmost vertex is unique by Lemma 3.2.3. If the interval of v is not intersected by any I_u with $u \in S \setminus \{v\}$ before querying Q, then we clearly already know the orientation. Thus, assume otherwise. Since the instance is preprocessed, we, by Lemma 3.2.3, have $I_u \not\subseteq I_v$ and $I_v \not\subseteq I_u$ for all $u \in S \setminus \{v\}$ (before querying Q). By definition, a vertex cover of the vertex cover instance $\bar{G}$ contains either (i) v or (ii) $S \setminus \{v\}$.

Consider case (i) and let u be leftmost in $S \setminus \{v\}$. If querying v reveals $w_v \notin I_u$, then v has minimum weight in S and we know the orientation of S. If querying v reveals $w_v \in I_u$, then the query reduces I_v to $\{w_v\}$ and u becomes mandatory by Corollary 2.3.6 as

$I_v = \{w_v\} \subseteq I_u$ and u is leftmost in S after querying v. After querying u (to exhaustively apply Corollary 2.3.6), we can repeat the argument with the vertex leftmost in $S \setminus \{u, v\}$. Thus, exhaustively applying Corollary 2.3.6 gives us the orientation.

Consider case (ii). If querying $S \setminus \{v\}$ reveals that there exists no vertex $u \in S \setminus \{v\}$ with $w_u \in I_v$, then v must be of minimum weight in S. Otherwise, the uncertainty interval of some $u \in S \setminus \{v\}$ was reduced to $\{w_u\} \subseteq I_v$ and we can apply Corollary 2.3.6. After that, all elements of S are queried and we clearly know the orientation. □

3.2.4 Lower Bounds on the Expected Optimal Query Cost

To analyze our algorithms and achieve improved stochastic competitive ratios, we require stronger lower bounds on $\mathbb{E}[\mathrm{OPT}]$. Let $\mathcal{R}$ be the set of all possible realizations of precise weights and let $\mathrm{OPT}(R)$ for $R \in \mathcal{R}$ be the optimal query cost for realization R. By Observation 3.2.7, the minimum weight of a vertex cover of $\bar{G}$ (using the query costs as weights) is a lower bound on the optimal query cost for each realization and, thus, on $\mathbb{E}[\mathrm{OPT}]$. This observation in combination with Lemma 2.3.5 also gives us a way to identify an optimal query set for a fixed realization, by using the knowledge of the exact vertex weights.

For a graph $G = (V, E)$ and a subset $U \subseteq V$, we use $G[U]$ to refer to the subgraph of G induced by U.

Lemma 3.2.9. *Consider a preprocessed instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. For a fixed realization R, let M be the set of vertices that are mandatory for the realization of precise weights (cf. Lemma 2.3.5), and let VC_M be a minimum-weight vertex cover of $\bar{G}[V \setminus M]$ (using the query costs as weights). Then, $M \cup VC_M$ is an optimal query set for realization R.*

Proof. Consider a preprocessed instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$, uncertainty intervals I_v for all $v \in V$, and a fixed realization of precise weights w_v for all $v \in V$.

Clearly, $c(M) + c(VC_M) \leq \mathrm{OPT}(R)$ as all vertices in M are mandatory and all edges in $\bar{G}[V \setminus M]$ are witness sets by Lemma 2.3.7. Furthermore, set $Q = M \cup VC_M$ is a vertex cover for $\bar{G}$ *and* contains all mandatory vertices. By Lemma 3.2.8, this suffices to conclude that Q is feasible, which implies the lemma. □

To analyze the performance of our algorithms, we compare the expected cost of the algorithms to the expected cost of the optimal solution. By Lemma 3.2.9, $c(M) + c(VC_M)$ is the minimum query cost for a fixed realization R of a preprocessed instance, where $M \subseteq V$ is the set of mandatory elements in the realization and VC_M is a minimum-weight vertex cover for the subgraph $\bar{G}[V \setminus M]$ of the vertex cover instance $\bar{G} = (V, \bar{E})$ induced by $V \setminus M$. Thus, the optimal solution for a fixed realization is completely characterized by the set of mandatory vertices in the realization. Using this, we can characterize the expected optimal query cost for a preprocessed instance as $\mathbb{E}[\mathrm{OPT}] = \sum_{M \subseteq V} p(M) \cdot c(M) + \sum_{M \subseteq V} p(M) \cdot c(VC_M)$, where $p(M)$ denotes the probability that M is the set of mandatory elements. Using that $\sum_{M \subseteq V} p(M) \cdot c(M) = \sum_{v \in V} p_v \cdot c(v)$ holds since both terms describe the expected cost for querying mandatory vertices, we can derive the following characterization of $\mathbb{E}[\mathrm{OPT}]$:

$$\mathbb{E}[\mathrm{OPT}] = \sum_{v \in V} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M).$$

A key technique for our analysis is to lower bound $\mathbb{E}[\mathrm{OPT}]$ by partitioning the optimal solution into subproblems and discarding dependencies between elements in different subproblems. The following definition and lemma make this more concrete.

Definition 3.2.10. *Consider a preprocessed instance of hypergraph orientation under explorable stochastic uncertainty with hypergraph $H = (V, E)$. For a realization R and any subset $S \subseteq V$, let $\mathrm{OPT}_S = \min_{Q \in \mathcal{Q}} c(Q \cap S)$, where $\mathcal{Q}$ is the set of all feasible query sets for realization R.*

Lemma 3.2.11. *Consider a preprocessed instance of hypergraph orientation under explorable stochastic uncertainty with hypergraph $H = (V, E)$. Let $S_1, \dots, S_k$ be a partition of V. Then $\mathbb{E}[\mathrm{OPT}] \geq \sum_{i=1}^{k} \mathbb{E}[\mathrm{OPT}_{S_i}]$.*

Proof. We start the proof by characterizing $\mathbb{E}[\mathrm{OPT}_{S_i}]$ for each $i \in \{1, \dots, k\}$. Let $R \in \mathcal{R}$ an arbitrary realization and let $M \subseteq V$ denote the set of vertices that are mandatory in realization R. By Lemma 3.2.9, the optimal solution for that realization needs to contain all mandatory elements of S_i, and resolve all remaining dependencies between vertices of S_i, i.e., query a minimum-weight vertex cover $VC_M^{S_i}$ for the subgraph $\bar{G}[S_i \setminus M]$. Thus, we arrive at

$$\mathbb{E}[\mathrm{OPT}_{S_i}] = \sum_{v \in S_i} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M^{S_i}). \tag{3.2.1}$$

By summing Equation (3.2.1) over all $i \in \{1, \dots, k\}$, we obtain the lemma:

$$\begin{aligned}\sum_{i=1}^{k} \mathbb{E}[\mathrm{OPT}_{S_i}] &= \sum_{i=1}^{k} \left(\sum_{v \in S_i} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M^{S_i}) \right) \\ &= \sum_{v \in V} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot \left(\sum_{i=1}^{k} c(VC_M^{S_i}) \right) \\ &\leq \sum_{v \in V} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M) = \mathbb{E}[\mathrm{OPT}],\end{aligned}$$

where the second equality follows from $S_1, \dots, S_k$ being a partition. The inequality follows from $\sum_{i=1}^{k} c(VC_M^{S_i})$ being the cost of a minimum weighted vertex cover for a subgraph of $\bar{G}[V \setminus M]$, while $c(VC_M)$ is the minimum cost for a vertex cover of the whole graph. □

3.2.5 Vertex Cover-based Algorithms

Using the vertex cover instance, we can not only lower bound $\mathbb{E}[\mathrm{OPT}]$, but also define a class of algorithms that exploits Observation 3.2.7.

Definition 3.2.12. *An algorithm for solving a preprocessed instance of hypergraph orientation under explorable uncertainty is called* vertex cover-based *if it implements the following pattern:*

1. *Non-adaptively query a vertex cover VC of $\bar{G}$.*

2. *Iteratively query mandatory vertices until the minimum-weight vertex of each hyperedge is known: For each hyperedge $S \in E$ for which the minimum weight is still unknown, query the vertices in order of left interval endpoints until the vertex of minimum weight is found.*

By definition of the second step, each vertex cover-based algorithm clearly orients each hyperedge. Furthermore, the proof of Lemma 3.2.8 implies that each vertex queried in the second step is indeed mandatory for all realizations that are consistent with the currently known information, i.e., the weights of the previously queried vertices.

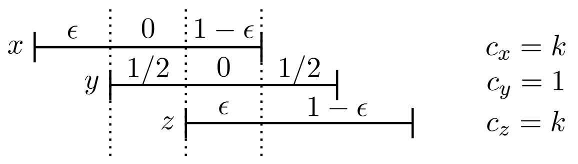


FIGURE 3.4: Lower bound example for orienting a single hyperedge.

All the algorithms we propose in this chapter are vertex cover-based and we show that this framework can be used to improve upon the adversarial lower bound of 2. However, we also give lower bounds for algorithms that follow this pattern.

Theorem 3.2.13. *No vertex cover-based algorithm has competitive ratio better than $\frac{3}{2}$ for hypergraph orientation under explorable stochastic uncertainty. This result holds even in the following special cases:*

1. *The graph has only a single hyperedge but the query costs are not uniform.*
2. *The query costs are uniform and the vertex cover instance $\bar{G}$ is bipartite.*
3. *The instance is a non-bipartite graph orientation instance with uniform query costs.*

Proof. **Case 1.** Consider the instance consisting of a single hyperedge $\{x, y, z\}$ with intervals and query costs as shown in Figure 3.4. Each vertex cover-based algorithm has to query a vertex cover of $\bar{G}$ in the first stage. As the edge set of the vertex cover instance consists of the two edges $\{x, y\}$ and $\{x, z\}$, each algorithm either queries $A_1 = \{x\}$, $A_2 = \{y, z\}$ or a superset of one of them in the first stage. Since querying supersets of A_1 or A_2 in the first stage never improves in comparison to just querying A_1 or A_2, we discard that option. For ϵ tending to zero, the expected cost of the two-stage algorithms are $\mathbb{E}[A_1] = \mathbb{E}[A_2] = k + 1 + \frac{1}{2} \cdot k = \frac{3}{2} \cdot k + 1$, while the expected optimum is $\mathbb{E}[\text{OPT}] = 1 + \frac{1}{2} \cdot k + \frac{1}{2} \cdot k = k + 1$. Therefore, for k tending to infinity, the competitive ratio approaches $\frac{3}{2}$.

Case 2. The argumentation is analogous to the proof of the first case but uses the instance in Figure 3.5. The hyperedges are $S_1, \ldots, S_k$, with $S_i = \{x_i, y, z_1, \ldots, z_k\}$.

Case 3. The argumentation is analogous to the proof of Case 1 but uses the instance in Figure 3.6. □

3.2.6 Hardness of the Offline Problem

We conclude the section by proving that the *offline variant* of hypergraph orientation under explorable uncertainty is NP-hard. The offline variant is the problem of computing a feasible query set of minimum query costs for a fixed realization of precise weights and given *full knowledge of the precise weights*.

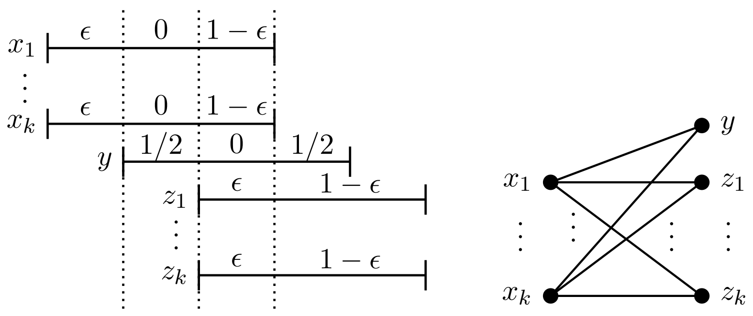


FIGURE 3.5: Lower bound example for the hyperedge orientation problem. The vertex cover instance is shown in the complete bipartite graph at the right.

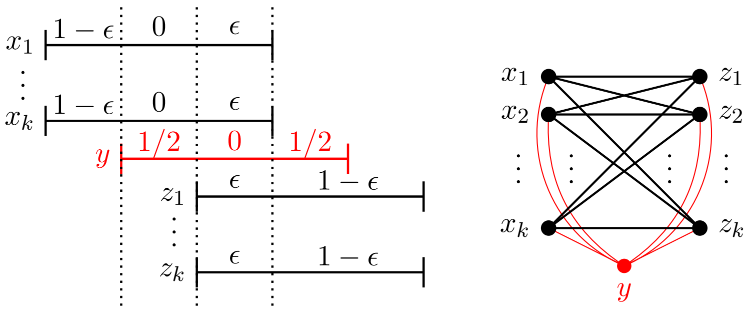


FIGURE 3.6: Lower bound example for the graph orientation problem with uniform query costs. The subgraph induced by $\{x_1, \dots x_k, z_1, \dots, z_k\}$ is complete bipartite, and y is a universal vertex (adjacent to all others).

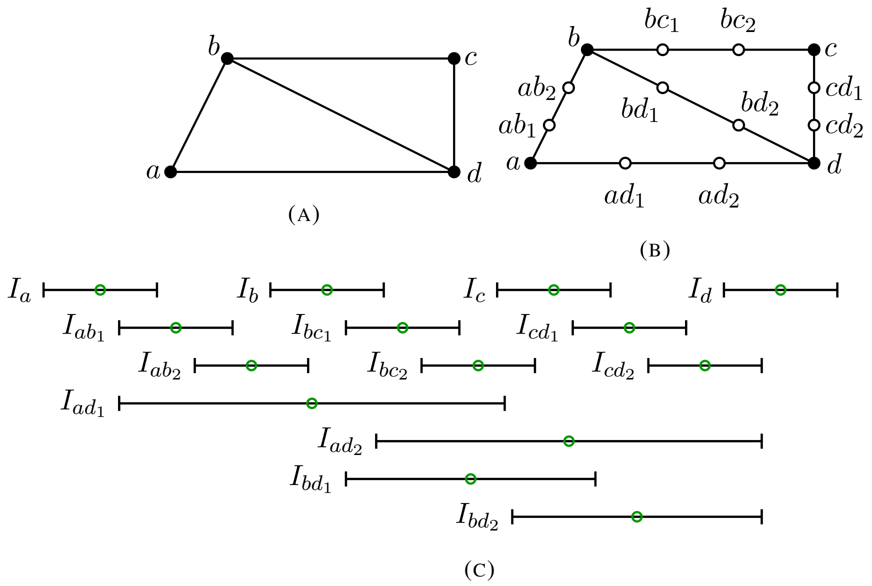


FIGURE 3.7: NP-hardness reduction for the hypergraph orientation problem from the vertex cover problem on 2-subdivision graphs. (a) A graph and (b) its 2-subdivision. (c) The corresponding instance for the hypergraph orientation problem.

Theorem 3.2.14. *The offline variant of hypergraph orientation under explorable uncertainty is NP-hard.*

Proof. The proof uses a reduction from the vertex cover problem for 2-subdivision graphs, which is NP-hard [Pol74]. A 2-subdivision graph is a graph H which can be obtained from some graph G by replacing each edge by a path of length four (with three edges and two new vertices). The graph in Figure 3.7b is a 2-subdivision of the graph in Figure 3.7a.

Given a graph H which is a 2-subdivision of a graph G, we construct an instance of graph orientation under uncertainty $G' = (V', E')$ as follows. First, we set $G' = H$, i.e., we use the 2-subdivision as the input graph for our graph orientation instance. It remains to define the intervals and precise weights.

For the original vertices of G, we define the intervals in a way such that no two intervals intersect each other. For an edge $e = \{u, v\}$ of G, let v_1 and v_2 denote the vertices introduced to replace e when creating the 2-subdivision H. We define the intervals I_{v_1} and I_{v_2} such that two intervals of $I_v, I_{v_1}, I_{v_2}, I_u$ intersect if and only if the corresponding vertices define an edge in $G' = H$. As illustrated in Figure 3.7c, placing the intervals in such a way is clearly possible.

Finally, we can assign precise weights such that, for any edge $\{u, v\} \in E'$, neither $w_v \in I_u$ nor $w_u \in I_v$. By definition of the intervals, this is possible. To orient each edge, we have to query at least one endpoint. By definition of the precise weights, querying one endpoint is sufficient to orient any edge. Clearly, on such instances, every solution to the offline variant of the graph orientation problem corresponds to a vertex cover of H, and vice-versa. □

Note that the reduction used to prove the theorem preserves the approximation factor, i.e., an α-approximation for the offline variant of hypergraph orientation implies an α-approximation for vertex cover on 2-subdivision graphs. Since Chlebík and Chlebíková [CC07] showed hardness of approximation for vertex cover on 2-subdivision graphs, we can conclude the following corollary.

Corollary 3.2.15. *The offline variant of hypergraph orientation under explorable uncertainty is APX-hard.*

3.3 A Threshold Algorithm for Orienting Graphs

In this section, we introduce a vertex cover-based algorithm for the graph orientation problem with uniform query costs. The techniques and analysis used in this section can be generalized to hypergraph orientation and arbitrary weights; we do so in the subsequent sections.

As a subproblem, our algorithm solves a vertex cover problem. This problem is NP-hard and 2-approximation algorithms are known [YG80]. For several special graph classes, there are improved algorithms [Hal02]. Using an α-approximation for vertex cover as a black box, we achieve a competitive ratio between $\phi \approx 1.618$ ($\alpha = 1$) and 2 ($\alpha = 2$) as a function depending on α. This function is shown in Figure 3.8.

The main idea behind Algorithm 5 is motivated by the lower bounds on $\mathbb{E}[\mathrm{OPT}]$ from the previous section. For one, we know that vertices with a high mandatory probability are likely to be part of OPT. Intuitively, querying such vertices cannot be too bad for our competitive ratio. Thus, Algorithm 5 selects vertices with a mandatory probability larger than a certain threshold $d \in (0, 1]$ (cf. Line 1) as part of the vertex cover that is queried in the first phase of the vertex cover-based algorithm (cf. Line 5). Then, we know by Observation 3.2.7 that a minimum vertex cover of the vertex cover instance is a lower bound on $\mathbb{E}[\mathrm{OPT}]$ (for graph orientation, the vertex cover instance is just the input graph). So the algorithm approximates a minimum vertex cover for the subgraph that remains after removing the already selected vertices with high mandatory probabilities (cf. Lines 2-4) and queries it in the first phase (cf. Line 5). For technical reasons that ease the analysis, the algorithm executes a preprocessing of the vertex

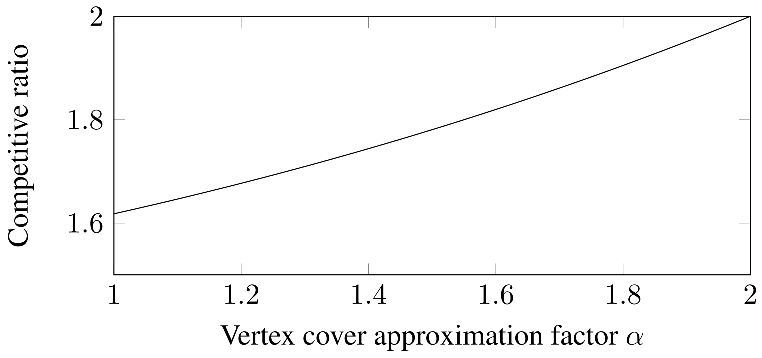


FIGURE 3.8: Competitive ratio of THRESHOLD for different approximation factors α of the vertex cover problem blackbox.

Algorithm 5: THRESHOLD

```
Input: Preprocessed instance of graph orientation under stochastic explorable
       uncertainty with graph $G = (V, E)$, distributions $d_v$ and uncertainty intervals
       $I_v$ for each $v \in V$. Threshold parameter $d \in (0, 1]$ and an $\alpha$-approximation
       black box for the vertex cover problem.
1 Let $M = \{v \in V \mid p_v \geq d\}$;
2 Solve (LP) for $G[V \setminus M]$ and let $x^*$ be an optimal basic feasible solution;
3 Let $V_1 = \{v \in V \mid x^*_v = 1\}$ and similarly $V_{1/2}, V_0$ ;
4 Use the $\alpha$-approximation black box to approximate a vertex cover $VC'$ for $G[V_{1/2}]$;
5 Query $Q = M \cup V_1 \cup VC'$;                /* $Q$ is a vertex cover */
6 Query the mandatory elements of $V \setminus Q$ ;
```

cover instance by using the following classical LP relaxation, for which each optimal basic feasible solution is half-integral [NJ75], before actually approximating the vertex cover:

$$\begin{array}{lll} \min & \sum_{v \in V} c_v \cdot x_v & \\ \text{s.t.} & x_v + x_u \geq 1 & \forall \{u, v\} \in E \\ & x_v \geq 0 & \forall v \in V \end{array} \tag{LP}$$

After querying the vertex cover in Line 5, the algorithm, as every vertex cover-based algorithm, only queries remaining vertices if they are mandatory (cf. Line 6). Since the algorithm already queried all vertices with a high mandatory probability (cf. Line 1), we can bound the expected number of queries in Line 6. Formalizing these ideas, we analyze our algorithm to show the following theorem.

Theorem 3.3.1. *Given an α-approximation with $1 \leq \alpha \leq 2$ for the vertex cover problem (on the induced subgraph $G[V_{1/2}]$, see Line 4),* THRESHOLD *with parameter $d \in (0, 1]$ achieves a competitive ratio of $\max\{\frac{1}{d}, \alpha + (2 - \alpha) \cdot d\}$ for the graph orientation problem under explorable stochastic uncertainty with uniform query costs. Optimizing d yields a competitive ratio of $\frac{1}{2}(\alpha + \sqrt{8 - \alpha(4 - \alpha)})$. The running time of* THRESHOLD *is polynomial in the input.*

Proof. Since Q is a vertex cover for G, querying it in Line 5 and resolving all remaining dependencies in Line 6 solves the graph orientation problem by Lemma 3.2.8. Note that $V \setminus Q$ is an independent set in G, and, thus, after querying Q we exactly know which elements of $V \setminus Q$ are mandatory. Hence, it is known after Line 5 which nodes in $V \setminus Q$ are mandatory, and they can be queried in Line 6 in arbitrary order (or in parallel).

As we argued in Section 3.2, the mandatory probabilities for a graph orientation instance can be computed in polynomial time and, therefore, Line 1 can be executed in polynomial time. Since the α-approximation blackbox has polynomial running time by assumption, all other steps of the algorithm, and thus the complete algorithm, can be executed in polynomial time.

We continue by showing the competitive ratio of $\max\{\frac{1}{d}, \alpha + (2 - \alpha) \cdot d\}$. Algebraic transformations show that the optimal choice for the threshold is $d(\alpha) = 2/(\alpha + \sqrt{8 - \alpha(4 - \alpha)})$. The desired competitive ratio for THRESHOLD with $d = d(\alpha)$ follows.

The algorithm queries set Q, and all other vertices only if they are mandatory, hence

$$\mathbb{E}[\text{ALG}] = |Q| + \sum_{v \in V \setminus Q} p_v = |M| + |V_1| + |VC'| + \sum_{v \in V_0} p_v + \sum_{v \in V_{1/2} \setminus VC'} p_v. \tag{3.3.1}$$

The expected optimal cost can be lower bounded by partitioning and Lemma 3.2.11:

$$\mathbb{E}[\text{OPT}] \geq \mathbb{E}[\text{OPT}_M] + \mathbb{E}[\text{OPT}_{V_1 \cup V_0}] + \mathbb{E}[\text{OPT}_{V_{1/2}}]. \tag{3.3.2}$$

In the remaining part of the proof, we compare $\mathbb{E}[ALG]$ with $\mathbb{E}[\mathrm{OPT}]$ component-wise.

We can lower bound $\mathbb{E}[\mathrm{OPT}_M]$ by $\sum_{v \in M} p_v$ using Equation (3.2.1). By definition of M, it holds that $\mathbb{E}[\mathrm{OPT}_M] \geq \sum_{v \in M} p_v \geq d \cdot |M|$. Thus,

$$|M| \leq \frac{1}{d} \cdot \mathbb{E}[\mathrm{OPT}_M]. \tag{3.3.3}$$

Next, we compare $|V_1| + \sum_{v \in V_0} p_v$ with $\mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}]$. For this purpose, let $G[V_1 \cup V_0]$ be the subgraph of G induced by $V_1 \cup V_0$, and let $G'[V_1 \cup V_0]$ be the bipartite graph that is created by removing all edges between elements of V_1 from $G[V_1 \cup V_0]$. Note that there cannot be edges between vertices of V_0 in $G[V_1 \cup V_0]$ by definition of V_0. The following statement can be shown by slightly adjusting arguments from [NJ75, Theorem 2].

Claim 3.3.2. *We claim that V_1 is a minimum vertex cover for $G'[V_1 \cup V_0]$.*

Proof. Assume otherwise. Then there is a vertex cover VC^* for $G'[V_1 \cup V_0]$ with $c(VC^*) < c(V_1)$. Since we consider uniform query costs c, we have $c(VC^*) = |VC^*|$ and $c(V_1) = |V_1|$. Let $R_1 = V_1 \setminus VC^*$ and let $A_0 = V_0 \cap VC^*$, then $VC^* = (V_1 \setminus R_1) \cup A_0$. Since $c(VC^*) < c(V_1)$, we get $c(R_1) > c(A_0)$. We define a solution x' for the LP Relaxation of Line 2 in THRESHOLD as follows:

$$x'_v = \begin{cases} \frac{1}{2} & \text{if } v \in R_1 \cup A_0 \\ x^*_v & \text{otherwise,} \end{cases}$$

where x^* refers to the solution computed in Line 2 of THRESHOLD. The objective value $c(x')$ of x' is $c(x') = c(x^*) - \frac{1}{2} \cdot c(R_1) + \frac{1}{2} \cdot c(A_0)$, where $c(x^*)$ is the objective value of x^*. We argue that x' is feasible for the LP relaxation, which contradicts the optimality of x^*.

All edges that are only incident to elements of $V_{1/2} = V \setminus (V_1 \cup V_0)$ are still covered, because the values of the corresponding variables were not changed. Each edge between some $u \in V_{1/2}$ and some $v \in V_1$ is still covered because $x'_u = \frac{1}{2}$ and $x'_v \geq \frac{1}{2}$ holds by definition of x'. Edges between elements of V_1 are covered for the same reason. Since there are no edges between elements of V_0, it remains to consider an edge between some $u \in V_0$ and some $v \in V_1$. If $v \in V_1 \cap VC^*$, then $x'_v = 1$, and the edge is covered. If $v \in V_1 \setminus VC^* = R_1$, then $u \in A_0$ because VC^* is a vertex cover for $G'_{V_1 \cup V_0}$. By definition of x', we have $x'_v = \frac{1}{2}$ and $x'_u = \frac{1}{2}$, and, therefore, the edge is covered. This implies that x' is feasible, which contradicts the optimality of x^*. Thus, V_1 is a minimum weighted vertex cover for $G'_{V_1 \cup V_0}$. □

Claim 3.3.2 allows us to apply the famous Kőnig-Egerváry theorem [Sch03] on the bipartite graph $G'[V_1 \cup V_0]$ and its minimum vertex cover V_1. By the theorem there is a matching h mapping each $v \in V_1$ to a distinct $h(v) \in V_0$ with $\{v, h(v)\} \in E$. Denoting $S = \{h(v) \mid v \in V_1\}$, we can infer $\mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}] \geq \mathbb{E}[\mathrm{OPT}_{V_1 \cup S}] + \mathbb{E}[\mathrm{OPT}_{V_0 \setminus S}]$.

Any feasible solution must query at least one endpoint of all edges of the form $\{v, h(v)\}$ as those are witness sets by Lemma 2.3.7 and Assumption 3.2.1. This implies $\mathbb{E}[\mathrm{OPT}_{V_1 \cup S}] \geq |V_1|$. Since additionally $p_{h(v)} \leq d$ for each $h(v) \in S$ by definition of the algorithm, we get

$$\sum_{v \in V_1} \left(1 + p_{h(v)}\right) \leq (1 + d) \cdot |V_1| \leq (1 + d) \cdot \mathbb{E}[OPT_{V_1 \cup S}]. \tag{3.3.4}$$

By lower bounding $\mathbb{E}[OPT_{V_0 \setminus S}]$ with $\sum_{v \in V_0 \setminus S} p_v$ and using (3.3.4), we get

$$\begin{aligned} |V_1| + \sum_{v \in V_0} p_v &= \sum_{v \in V_1} \left(1 + p_{h(v)}\right) + \sum_{v \in V_0 \setminus S} p_v \\ &\leq (1 + d) \cdot \mathbb{E}[\mathrm{OPT}_{V_1 \cup S}] + \mathbb{E}[\mathrm{OPT}_{V_0 \setminus S}] \leq (1 + d) \cdot \mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}]. \end{aligned} \tag{3.3.5}$$

Finally, consider the term $|VC'| + \sum_{v \in V_{1/2} \setminus VC'} p_v$. Let VC^* be a minimum cardinality vertex cover for $G[V_{1/2}]$. Then, it holds $|VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$. This is, because in the optimal basic feasible solution to the LP relaxation x^*, each vertex in $V_{1/2}$ has a value of $\frac{1}{2}$. A vertex cover with $|VC^*| < \frac{1}{2} \cdot |V_{1/2}|$ would contradict the optimality of x^*. The following part of the analysis crucially relies on $|VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$, which is the reason why THRESHOLD executes the LP relaxation-based preprocessing before applying the α-approximation.

The expected cost of the algorithm for the subgraph $G[V_{1/2}]$ is $|VC'| + \sum_{v \in I'} p_v \leq |VC'| + d \cdot |I'|$ with $I' = V_{1/2} \setminus VC'$ as the algorithm queries VC' in Line 5 and the vertices in I' only if they are mandatory in Line 6. Since $|VC'| \geq |VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$, there is a tradeoff between the quality of $|VC'|$ and the additional cost of $d \cdot |I'|$. If $|VC'|$ is close to $\frac{1}{2} \cdot |V_{1/2}|$, then it is close to $|VC^*|$ but, on the other hand, $|I'|$ then is close to $\frac{1}{2} \cdot |V_{1/2}|$, which means that the additional cost $d \cdot |I'|$ is high. Vice versa, if the cost for $|VC'|$ is high because it is larger than $\frac{1}{2} \cdot |V_{1/2}|$, then $|I'|$ is close to zero and the additional cost $d \cdot |I'|$ is low. We exploit this tradeoff and upper bound $\frac{|VC'|+d \cdot |I'|}{|VC^*|}$ in terms of the approximation factor α of the vertex cover approximation. Assume that the approximation factor α is tight, i.e., $|VC'| = \alpha \cdot |VC^*|$. Since $d \leq 1$, this is the worst case for the ratio $\frac{|VC'|+d \cdot |I'|}{|VC^*|}$. (In other words, if the approximation factor was not tight, we could replace α by the approximation factor that is actually achieved and carry out the following calculations with that smaller value of α instead, yielding an even better bound.) Using $|VC'| = \alpha \cdot |VC^*|$ and $|VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$, we can derive

$$|I'| = |V_{1/2}| - |VC'| = |V_{1/2}| - \alpha \cdot |VC^*| \leq (2 - \alpha) \cdot |VC^*|.$$

For the cost of the algorithm for subgraph $G[V_{1/2}]$, we get

$$|VC'| + d \cdot |I'| \leq \alpha \cdot |VC^*| + d \cdot (2 - \alpha) \cdot |VC^*| = (\alpha + (2 - \alpha) \cdot d) \cdot |VC^*|.$$

Since $|VC^*| \leq \mathbb{E}[\mathrm{OPT}_{V_{1/2}}]$, this implies

$$|VC'| + d \cdot |I'| \leq (\alpha + (2 - \alpha) \cdot d) \cdot \mathbb{E}[\mathrm{OPT}_{V_{1/2}}]. \tag{3.3.6}$$

Combining Equations (3.3.1), (3.3.3), (3.3.5), and (3.3.6), we can upper bound the cost of the algorithm:

$$\begin{aligned}
\mathbb{E}[\mathrm{ALG}] &= |M| + |V_1| + \sum_{v \in V_0} p_v + |VC'| + \sum_{v \in V_{1/2} \setminus VC'} p_v \\
&\leq \frac{1}{d} \cdot \mathbb{E}[\mathrm{OPT}_M] + (1 + d) \cdot \mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}] + (\alpha + (2 - \alpha) \cdot d) \cdot \mathbb{E}[\mathrm{OPT}_{V_{1/2}}] \\
&\leq \max\left\{\frac{1}{d}, (1 + d), (\alpha + (2 - \alpha) \cdot d)\right\} \cdot \mathbb{E}[\mathrm{OPT}],
\end{aligned}$$

where the last inequality follows from the lower bound on $\mathbb{E}[\mathrm{OPT}]$ in Equation (3.3.2). Observe that for any $d \in (0, 1]$ and $\alpha \in [1, 2]$, it holds that $(\alpha + (2 - \alpha) \cdot d) \geq (1 + d)$. We conclude that $\mathbb{E}[ALG] \leq \max\{\frac{1}{d}, (\alpha + (2 - \alpha) \cdot d)\} \cdot \mathbb{E}[\mathrm{OPT}]$, which implies the theorem. □

We show that the above analysis for THRESHOLD is tight.

Theorem 3.3.3. *The analysis of* THRESHOLD *is tight. More precisely, there is no threshold $d \in (0, 1]$ such that the competitive ratio of* THRESHOLD *is less than the Golden ratio $\phi = (1 + \sqrt{5})/2$.*

Proof. Consider THRESHOLD with some threshold $d \in (0, 1]$. We give two instances of the graph orientation problem with uniform query costs and show that the algorithm has either

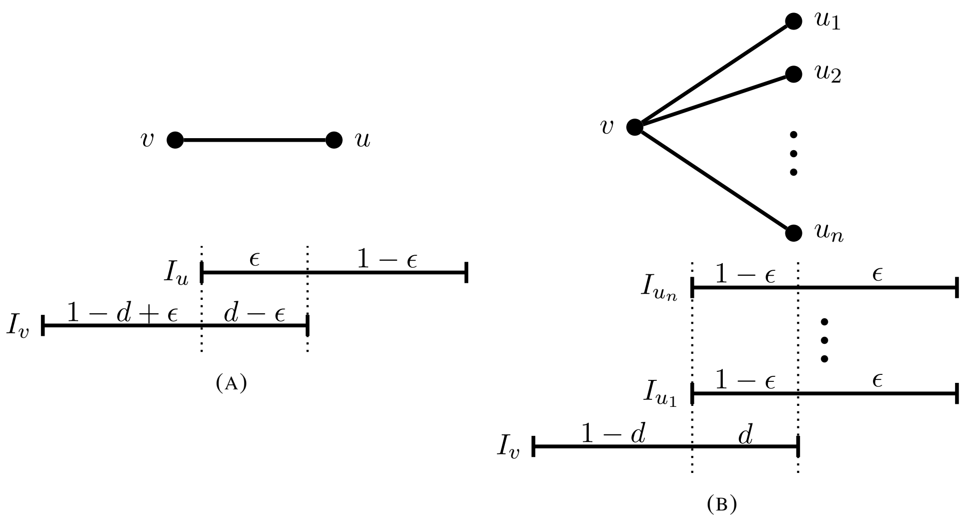


FIGURE 3.9: Instance of graph orientation under explorable stochastic uncertainty as used in the proof of Theorem 3.3.3

a competitive ratio of at least $1 + d - \epsilon$, for an arbitrarily small $\epsilon > 0$, or at least $1/d$. This implies the lower bound.

As the first instance, consider a single edge $\{u, v\}$ and mandatory probabilities $p_u = d - \epsilon$ and $p_v = \epsilon$ (see Figure 3.9 (A) for an illustration). Threshold finds a basic feasible solution for the vertex cover LP relaxation for the edge $\{u, v\}$, say $x_u^{\text{LP}} = 0$ and $x_v^{\text{LP}} = 1$. It queries v, and then it still has to query u with probability $p_u = d - \epsilon$. Thus, the expected number of queries is $1 + d - \epsilon$. The expected optimal number of queries on the other hand is not larger than $1 + \epsilon$ as this expected cost can be achieved by the algorithm that starts by querying u and only queries v if it is mandatory. This implies the competitive ratio of THRESHOLD is at least $\frac{1+d-\epsilon}{1+\epsilon}$, which tends towards $1 + d$ for $\epsilon \to 0$.

As a second instance, consider a star with the center v and n edges $\{v, u_i\}$ for $i \in \{1, 2, \ldots, n\}$. Let $p_v = 1$ and $p_{u_i} = d$ (see Figure 3.9 (B) for an illustration). Then the algorithm queries v and all u_i due to the threshold d, whereas it would have been sufficient to query v first and then continue with querying u_i only if needed, that is, with probability $p_{u_i} = d$ for each i. The algorithm has cost $n + 1$, while the optimal cost is at most $1 + n \cdot d$. Thus, the competitive ratio is at least $(n + 1)/(1 + n \cdot d)$, which tends to $1/d$ for $n \to \infty$. □

3.4 Threshold Algorithm for Arbitrary Query Costs

In this section, we generalize the analysis of the threshold algorithm for orienting graphs to arbitrary query costs. Even for arbitrary query costs, the algorithm remains largely the same. In contrast to the uniform problem variant, the adjusted algorithm solves a weighted version of (LP), and a weighted vertex cover problem of $\bar{G}[V_{1/2}]$. In both instances, the query costs are used as weights.

However, the analysis of the uniform variant does not directly translate to the weighted setting. In particular, when analyzing the subgraph $\bar{G}[V_1 \cup V_0]$, we applied the unweighted Kőnig-Egerváry theorem. For each $v \in V_1$, this gave us a distinct partner vertex $h(v) \in V_0$ with $\{v, h(v)\} \in \bar{E}$. THRESHOLD queries v, and $h(v)$ only if necessary, leading to an expected query cost of $1 + d$ for the vertices $\{v, h(v)\}$. Since OPT has to query at least one of $\{v, h(v)\}$, this gave us the local competitive ratio of $1 + d$.

In the weighted setting, this argumentation does not work anymore. Intuitively, instead of charging the complete query cost c_v of an $v \in V_1$ to a single partner $h(v)$, Kőnig-Egerváry now distributes the query cost c_v to multiple vertices in V_0. Also, multiple vertices in V_1 might distribute query cost to the same $u \in V_0$.

Therefore, we require new tools in the analysis. To this end, we introduce new fractional lower bounds on $\mathbb{E}[\mathrm{OPT}]$. Afterwards, we use these lower bounds to analyze THRESHOLD for arbitrary query costs.

3.4.1 Fractional Lower Bounds on the Expected Optimum

A key part of our analysis for uniform query costs was the application of Lemma 3.2.11, which states $\mathbb{E}[\mathrm{OPT}] \geq \sum_{i=1}^{k} \mathbb{E}[\mathrm{OPT}_{S_i}]$ for any partition $S_1, \dots, S_k$ of V. For the case of arbitrary query costs, we rely on fractionally assigning a vertex v to multiple parts of the partition and applying a corresponding variant of Lemma 3.2.11. To this end, we define a *fractional partition* of V.

Definition 3.4.1. *Given a hypergraph $H = (V, E)$, let $\mathcal{S}$ be a family of subsets of V and let $f\colon V \times \mathcal{S} \to [0, 1]$. We say that $(\mathcal{S}, f)$ is a* partial fractional partition *of V if $\bigcup_{S\in\mathcal{S}} S \subseteq V$, $\sum_{S\in\mathcal{S}} f(v, S) \leq 1$ for all $v \in V$, and $f(v, S) = 0$ for all $v \in V$ and $S \in \mathcal{S}$ with $v \notin S$. If additionally $\bigcup_{S\in\mathcal{S}} S = V$ and $\sum_{S\in\mathcal{S}} f(v, S) = 1$ for all $v \in V$, then $(\mathcal{S}, f)$ is a* fractional partition.

Intuitively, the function f of a fractional partition $(\mathcal{S}, f)$ assigns fractions of each $v \in V$ to the subsets $S \in \mathcal{S}$ with $v \in S$. In particular, we want to distribute the query cost c_v of a vertex v fractionally over the sets $S \in \mathcal{S}$ with $v \in S$. We define the fractional costs $c'_{S,f}(v) = f(v, S) \cdot c_v$ for vertex v and the different sets S of the fractional partition $(\mathcal{S}, f)$. For subsets $U \subseteq V$, let $c'_{S,f}(U) = \sum_{v\in U} c'_{S,f}(v)$. By definition, the fractional costs of a vertex v sum up to at most c_v, i.e., $\sum_{S\in\mathcal{S}} c'_{S,f}(v) \leq c_v$. If $(\mathcal{S}, f)$ is *not* partial, then $\sum_{S\in\mathcal{S}} c'_{S,f}(v) = c_v$. Using these definitions, we show a generalized version of Lemma 3.2.11 for fractional partitions.

Definition 3.4.2. *Consider a preprocessed instance of hypergraph orientation under explorable stochastic uncertainty with hypergraph $H = (V, E)$. Let $(\mathcal{S}, f)$ be a partial fractional partition of H. For a realization R of precise weights and any subset $S \in \mathcal{S}$, let $\mathrm{OPT}_{S,f} = \min_{Q\in\mathcal{Q}} c'_{S,f}(Q \cap S)$, where $\mathcal{Q}$ is the set of all feasible query sets for realization R.*

Lemma 3.4.3. *Consider a preprocessed instance of hypergraph orientation under explorable stochastic uncertainty with hypergraph $H = (V, E)$. Let $(\mathcal{S}, f)$ be a partial fractional partition of H. Then, $\mathbb{E}[\mathrm{OPT}] \geq \sum_{S\in\mathcal{S}} \mathbb{E}[\mathrm{OPT}_{S,f}]$.*

Proof. We start the proof by characterizing $\mathbb{E}[\mathrm{OPT}_{S,f}]$ for each $S \in \mathcal{S}$. Let $R \in \mathcal{R}$ be a realization in which M is the set of mandatory elements.

In line with Lemma 3.2.9, the solution corresponding to $\mathrm{OPT}_{S,f}$ needs to contain all mandatory elements of S, and resolve all remaining dependencies between vertices of S, i.e., query a minimum-weight vertex cover VC^S_M for the subgraph $\bar{G}[S \setminus M]$ with respect to to the fractional vertex costs $c'_{S,f}(v) = f(v, S) \cdot c_v$. Thus, we get

$$\begin{aligned}\mathbb{E}[\mathrm{OPT}_{S,f}] &= \sum_{v\in S} p_v \cdot c'_{S,f}(v) + \sum_{M\subseteq V} p(M) \cdot c'_{S,f}(VC^{S_i}_M). \\ &= \sum_{v\in S} p_v \cdot f(v,S) \cdot c_v + \sum_{M\subseteq V} p(M) \cdot \left(\sum_{v\in VC^S_M} f(v,S) \cdot c_v \right),\end{aligned}$$

where $p(M)$ denotes the probability that M is the set of mandatory vertices. Summing over all $S \in \mathcal{S}$, we obtain the lemma:

$$\begin{aligned}\sum_{S\in\mathcal{S}} \mathbb{E}[\mathrm{OPT}_{S,f}] &= \sum_{S\in\mathcal{S}} \left(\sum_{v\in S} p_v \cdot f(v,S)\cdot c_v + \sum_{M\subseteq V} p(M)\cdot \left(\sum_{v\in VC^S_M} f(v,S)\cdot c_v \right)\right) \\ &\leq \sum_{v\in V} p_v\cdot c_v + \sum_{M\subseteq V} p(M)\cdot\left(\sum_{S\in\mathcal{S}} \sum_{v\in VC^S_M} f(v,S)\cdot c_v \right) \\ &\leq \sum_{v\in V} p_v \cdot c_v + \sum_{M\subseteq V} p(M)\cdot c(VC_M) = \mathbb{E}[\mathrm{OPT}],\end{aligned}$$

where the first inequality follows from $(\mathcal{S}, f)$ being a partial fractional partition. The second inequality follows from $\left(\sum_{S\in\mathcal{S}}\sum_{v\in VC^S_M} f(v,S)\cdot c_v\right)$ being the cost of a minimum weighted vertex cover for a relaxation of $\bar{G}[V \setminus M]$. □

3.4.2 A Threshold Algorithm for Arbitrary Query Costs

We consider an adjusted variant of THRESHOLD for graph orientation with arbitrary query costs that, in contrast to THRESHOLD for the uniform problem variant, solves a weighted version of (LP), and a weighted vertex cover problem in Line 4. In both cases, we use the query costs c_v as weights for the vertices in the vertex cover instances. Since these changes do not affect the pseudocode, we still refer to Algorithm 5 in the following theorem and proof.

Theorem 3.4.4. *Given an α-approximation with $1 \leq \alpha \leq 2$ for the weighted vertex cover problem (on the induced subgraph $G[V_{1/2}]$, see Line 4), the adjusted* THRESHOLD *with parameter d achieves a competitive ratio of* $\max\{\frac{1}{d}, \alpha + (2-\alpha)\cdot d\}$ *for graph orientation under explorable stochastic uncertainty with arbitrary query costs. Optimizing d yields a competitive ratio of $\frac{1}{2}(\alpha + \sqrt{8-\alpha(4-\alpha)})$. The running time of* THRESHOLD *is polynomial in the input size.*

Proof. The expected cost of the adjusted THRESHOLD for arbitrary query costs is

$$\begin{aligned}\mathbb{E}[ALG] &= c(Q) + \sum_{v\in V\setminus Q} c_v\cdot p_v \\ &= c(M) + c(V_1) + c(VC') + \sum_{v\in V_0} c_v\cdot p_v + \sum_{v\in V_{1/2}\setminus VC'} c_v \cdot p_v\end{aligned}$$

as the algorithm starts by querying $Q = M \cup V_1 \cup VC'$ (cf. Line 5) and queries the remaining vertices only if they are mandatory (cf. Line 6).

Analogously to the analysis of THRESHOLD for uniform query costs, we compare $\mathbb{E}[ALG]$ and $\mathbb{E}[\mathrm{OPT}]$ component-wise. Remember that $\mathbb{E}[\mathrm{OPT}]$ is characterized by Equation (3.3.2):

$$\mathbb{E}[\mathrm{OPT}] \geq \mathbb{E}[\mathrm{OPT}_M] + \mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}] + \mathbb{E}[\mathrm{OPT}_{V_{1/2}}].$$

The bounds in Equations (3.3.3) and (3.3.6) regarding $\mathbb{E}[\mathrm{OPT}_M]$ and $c(M)$, and $\mathbb{E}[\mathrm{OPT}_{V_{1/2}}]$ and $c(VC') + \sum_{v\in V_{1/2}\setminus VC'} c_v\cdot p_v$, respectively, can be generalized straightforwardly by just plugging in the more general query cost function into the analysis of Theorem 3.3.1. Only generalizing Equation (3.3.5) requires more effort.

To that end, we compare $c(V_1) + \sum_{v\in V_0} c_v \cdot p_v$ and $\mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}]$. According to Claim 3.3.2, V_1 is a minimum weighted vertex cover for the bipartite graph $G'[V_1 \cup V_0]$

that is created by removing all edges within the partitions V_1 and V_0 from the subgraph induced by $V_1 \cup V_0$. In contrast to the uniform query cost variant, we cannot apply the Kőnig-Egerváry theorem for unweighted vertex covers. Instead, the theorem gives us a function $\pi : E \to \mathbb{R}$ with $\sum_{\{u,v\}\in E} \pi(u,v) \leq c_v$ for each $v \in V_1 \cup V_0$. By duality theory, the constraint is tight for each $v \in V_1$, and $\pi(u,v) = 0$ holds if both u and v are in V_1 (or if there is no edge between v and u). Thus, we can interpret π as a function that distributes the complete query cost c_v of each $v \in V_1$ to the neighbors of v outside of V_1. For each $u \in V_0$, let τ_u denote the remaining part of c_u that is not used by π to cover the weight of any $v \in V_1$, i.e., $\tau_u = c_u - \sum_{\{u,v\}\in E} \pi(u,v)$. We can rewrite $c(V_1) + \sum_{u\in V_0} c_u \cdot p_u$ as

$$\sum_{v\in V_1} \left(c_v + \sum_{u\in V_0} \pi(u,v) \cdot p_u \right) + \sum_{u\in V_0} \tau_u \cdot p_u.$$

Instead of directly comparing this expression to $\mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}]$, we first use π and τ to create a fractional partition of $V_1 \cup V_0$ and exploit Lemma 3.4.3 to further lower bound $\mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}]$. For each $v \in V_1$, let $H_v = \{v\} \cup \{u \mid u \in V_0 \wedge \pi(u,v) > 0\}$ and define $\mathcal{S} = \{H_v \mid v \in V_1\} \cup \{V_0\}$. Furthermore let $f\colon V \times \mathcal{S} \to [0,1]$ with

$$f(u, H_v) = \begin{cases} 1 & \text{if } u = v \\ \frac{\pi(u,v)}{c_u} & \text{if } u \in H_v \setminus \{v\} \\ 0 & \text{otherwise} \end{cases}$$

and

$$f(u, V_0) = \begin{cases} \frac{\tau_u}{c_u} & \text{if } u \in V_0 \\ 0 & \text{otherwise.} \end{cases}$$

Clearly $(\mathcal{S}, f)$ is a partial fractional partition of $V_0 \cup V_1$ and, thus,

$$\begin{aligned} \mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}] &\geq \sum_{v\in V_1} \mathbb{E}[\mathrm{OPT}_{H_v,f}] + \mathbb{E}[\mathrm{OPT}_{V_0,f}] \\ &\geq \sum_{v\in V_1} \mathbb{E}[\mathrm{OPT}_{H_v,f}] + \sum_{u\in V_0} f(u,V_0) \cdot p_u \cdot c_u \\ &= \sum_{v\in V_1} \mathbb{E}[\mathrm{OPT}_{H_v,f}] + \sum_{v\in V_0} p_u \cdot \tau_u \end{aligned}$$

holds by Lemmas 3.4.3 and 3.2.9.

We continue by further lower bounding $\mathbb{E}[\mathrm{OPT}_{H_v,f}]$. Observe that the subgraph of the vertex cover instance induced by H_v is a star with the center v and the leaves $H_v \setminus \{v\}$. With respect to the fractional costs $c'_{H_v,f}(u) = f(u,H_v) \cdot c_u$ for all $u \in H_v$, the cost of a minimum weighted vertex cover for this subgraph is c_v. This is because $c'_{H_v,f}(v) = f(v,H_v) \cdot c_v = c_v$ and $c'_{H_v,f}(H_v \setminus \{v\}) = \sum_{u\in H_v\setminus\{v\}} f(u,H_v) \cdot c_u = \sum_{u\in H_v\setminus\{v\}} \pi(u,v) = c_v$, where the last equality holds by definition of π.

Thus, $\mathbb{E}[\mathrm{OPT}_{H_v,f}] \geq c_v$ holds by Observation 3.2.7, and, therefore,

$$\begin{aligned} \mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}] &\geq \sum_{v\in V_1} \mathbb{E}[\mathrm{OPT}_{H_v,f}] + \sum_{v\in V_0} p_u \cdot \tau_u \\ &\geq c(V_1) + \sum_{u\in V_0} \tau_u \cdot p_u. \end{aligned}$$

Finally, we compare $c(V_1) + \sum_{v \in V_0} c_v \cdot p_v$ against this lower bound on $\mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}]$ by using that $\sum_{u \in V_0} \pi(u,v) \leq c_v$ holds for all $v \in V_1$ and that $p_u \leq d$ holds for all $u \in V_0$:

$$\begin{aligned}
&c(V_1) + \sum_{u \in V_0} c_u \cdot p_u \\
\leq &\sum_{v \in V_1} \left(c_v + \sum_{u \in V_0} \pi(u,v) \cdot d \right) + \sum_{u \in V_0} \tau_u \cdot p_u \\
\leq &\sum_{v \in V_1} (1+d) \cdot c_v + \sum_{u \in V_0} \tau_u \cdot p_u \\
\leq &(1+d) \cdot \mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}].
\end{aligned}$$

The rest of the analysis follows the same pattern as for the uniform query costs. □

3.5 A Threshold Algorithm for Orienting Hypergraphs

In this section, we generalize THRESHOLD to hypergraphs. The algorithms and analyses of the previous sections exploit several properties of the graph orientation problem that do not directly transfer to hypergraphs.

To achieve a polynomial running time, we used that the mandatory probabilities can easily be computed for graphs. For hypergraphs, we show that computing these probabilities becomes #P-hard, and show how to approximate them via sampling.

Then, the algorithm exploits that, for graphs, the vertex cover instance $\bar{G}$ (cf. Definition 3.2.6) is equal to the input graph G. This means that after querying the vertex cover in Line 5 of Algorithm 5, we, for each edge e, either know the orientation of e or the remaining unqueried endpoint of e is mandatory by Lemma 2.3.5. The algorithm can use this to identify all vertices that became mandatory after Line 5 and non-adaptively query them (cf. Line 6). This clearly solves the problem as it queries the only remaining unqueried endpoint for each edge with a still unknown orientation. For hypergraphs, this is not the case anymore and the algorithm has to operate more adaptively after querying the vertex cover.

To conclude the section, we show that this additional adaptivity is indeed necessary by giving a lower bound for algorithms that employ less adaptivity.

3.5.1 Computing Mandatory Probabilities

Recall that we denote by p_v the probability that a vertex v is mandatory. For graphs, p_v is easy to compute as, by Lemma 2.3.5, v is mandatory if and only if $w_u \in I_v$ for some neighbor vertex u. Hence, $p_v = 1 - \prod_{u:\{u,v\} \in E} \mathbb{P}[w_u \notin I_v]$. For hypergraphs, however, we can show that the computation of p_v is #P-hard, even if all hyperedges have size 3.

Theorem 3.5.1. *Computing the mandatory probabilities p_v for a hypergraph $H = (V,E)$ is #P-hard, even if all hyperedges have size* 3.

Proof. The proof consists of a reduction from the #P-hard problem of counting the number of vertex covers of a given graph $G = (V,E)$ [Vad01]. We construct the following instance of the hypergraph orientation problem.

Let v' be a new vertex, which does not belong to V. We construct the hypergraph $H = (V \cup \{v'\}, E')$, where every edge $\{u,v\} \in E$ from the original graph corresponds to an hyperedge in E' of the form $\{u,v,v'\}$. The value w_v associated to every vertex $v \in V$ follows a distribution on the interval $I_v = (i\varepsilon, 2+i\varepsilon)$ for an arbitrary ε with $0 < \varepsilon < 1/n$, where i is the index of v for some arbitrary fixed ordering of V. The role of the shift by ε is to avoid that two intervals contain each other. The distribution on I_v is constructed such that w_v is less than 1 with probability $1/2$ and at least 1 with probability $1/2$. The weight associated to vertex v' belongs to the interval $(1, 2 + (n+1)\varepsilon)$.

What is the probability that v' is mandatory? By Lemma 2.3.5, vertex v' is mandatory if and only if there exists a hyperedge $\{u, v, v'\}$ with $w_v, w_u \in I_{v'} = (1, 2 + (n+1)\varepsilon)$. Thus, for every realization of the vertices, it is only crucial which of them have weight less than 1. Let $T \subseteq V$ be the set of the vertices with a value below the threshold 1. Every potential set T has the same probability $1/2^n$.

Now, v' is mandatory due to a set $\{u, v, v'\}$ if and only if both values w_u, w_v are at least 1. This means that v' is *not* mandatory if and only if the set T defined by the realization is a vertex cover for the graph G. In other words the probability that v' is mandatory equals $1 - \ell/2^n$, where ℓ is the number of vertex covers of G. As a consequence, computing this probability is #P-hard. □

Luckily, it is not difficult to get a good estimate of the probabilities to be mandatory for hypergraphs using sampling.

Lemma 3.5.2. *There is a polynomial-time randomized algorithm that, given a hypergraph $H = (V, E)$, a vertex $v \in V$, and parameters $\epsilon, \delta \in (0, 1)$, produces a value y such that $|y - p_v| \geq \epsilon$ with probability at most δ. Its time complexity is $O(|V| \cdot |E| \cdot \ln(1/\delta)/\epsilon^2)$.*

Proof. The algorithm consists of independently drawing $k = \lceil \ln(2/\delta)/(2\epsilon^2) \rceil$ realizations of the vertex values, and determining for each realization whether v is mandatory by using Lemma 2.3.5. The output y is the fraction of realizations where this event happened.

For the analysis, let X_v^i be the random variable indicating whether v is mandatory in the i-th realization. These variables are independent and have mean p_v. Using Hoeffding's concentration [Hoe94] bound, we have

$$\mathbb{P}\left[\left|\frac{1}{k}\sum_{i=1}^{k} X_v^i - p_v\right| > \epsilon\right] < 2\exp(-2k\epsilon^2).$$

The choice of k is motivated by the following equivalent inequalties.

$$\begin{aligned} 2\exp(-2k\epsilon^2) &\leq \delta \\ -2k\epsilon^2 &\leq \ln(\delta/2) \\ k &\geq \ln(2/\delta)/(2\epsilon^2). \end{aligned}$$

For the time complexity, we observe that verifying if a vertex is mandatory in a sampled realization can be done in time $O(|V| \cdot |E|)$: We can iterate over the hyperedges E and, for each hyperedge $S \in E$, decide in time $\mathcal{O}(|V|)$ whether the vertices in S render v mandatory by Lemma 2.3.5.

The practical implementation of the sampling algorithm makes use of the given probability matrix. Let $t_1, \ldots, t_{2|V|}$ be the sorted elements of $\{L_v, U_v | v \in V\}$, defining elementary intervals of the form (t_i, t_{i+1}). The given probability matrix specifies for every given vertex $v \in V$ the probability that its weight w_v belongs to a given interval (t_i, t_{i+1}). Instead of sampling the actual weights, we only sample the elementary intervals to which they belong. Hence, for a fixed sample, we know for each hyperedge S in which elementary interval M its minimum weight lies. In this setting, the vertices that are mandatory because of S can be determined as follows, again using Lemma 2.3.5:

- If at least two weights of vertices in G belong to M, then all vertices $v \in S$ with $M \subseteq I_v = (L_v, U_v)$ are mandatory.

Algorithm 6: THRESHOLD for Hypergraphs

```
Input: Preprocessed instance of hypergraph orientation under stochastic explorable
       uncertainty with hypergraph $H = (V, E)$, distributions $d_v$ and uncertainty
       intervals $I_v$ for each $v \in V$. Threshold parameter $d \in (0, 1]$ and an
       $\alpha$-approximation black box for the vertex cover problem
1 For each $v \in V$, approximate mandatory probability $p_v$ using Lemma 3.5.2;
2 Let $M = \{v \in V \mid p_v \geq d\}$;
3 Solve (LP) for $\bar{G}[V \setminus M]$ and let $x^*$ be an optimal basic feasible solution;
4 Let $V_1 = \{v \in V \mid x^*_v = 1\}$ and similarly $V_{1/2}, V_0$ ;
5 Use the $\alpha$-approximation black box to approximate a vertex cover $VC'$ for $\bar{G}[V_{1/2}]$;
6 Query $Q = M \cup V_1 \cup VC'$;          /* $Q$ is a vertex cover for $\bar{G}$ */
7 for $S \in E$ do
8     while The orientation of $F$ is not known yet do
9         Query the leftmost vertex of $S$;
```

- Otherwise, let $m \in S$ be the unique vertex whose weight lies in M. We then have:
  - Every other vertex $u \in S$ with $M \subseteq I_u$ is mandatory.
  - If some vertex $u \in S$ has its weight in I_m, then m is also mandatory.

The union of the mandatory vertices of all hyperedges then gives the set of all mandatory vertices. □

3.5.2 A Threshold Algorithm for Orienting Hypergraphs

The threshold algorithm for hypergraphs, Algorithm 6, uses the sampling algorithm of Lemma 3.5.2 to approximate the mandatory probabilities of all vertices (cf. Line 1). Until Line 6, the algorithm mostly behaves like the threshold algorithm for graphs, but it computes the vertex cover on the vertex cover instance $\bar{G}$ instead of the input graph.

Afterwards, starting at Line 7, the algorithm iterates through the hyperedges and, if the orientation of a hyperedge is not yet known, repeatedly queries the leftmost vertex of the hyperedge until the orientation is known. By resolving the remaining hyperedges in this way, we only query mandatory vertices by Lemma 2.3.5 as shown in the proof of Lemma 3.2.8.

Theorem 3.5.3. *For $\epsilon, \delta > 0$ and given an α-approximation with $1 \leq \alpha \leq 2$ for the vertex cover problem (on the induced subgraph $\bar{G}[V_{1/2}]$ of the vertex cover instance given by Definition 3.2.6, cf. Line 5), the* THRESHOLD *algorithm for hypergraphs solves hypergraph orientation under explorable stochastic uncertainty with competitive ratio*

$$R = \frac{1}{2}\left(\alpha + \sqrt{\alpha^2 + 4(2-\alpha)(1 + \alpha\epsilon + (2-\alpha)\epsilon^2)} + (4 - 2\alpha)\epsilon\right)$$

with probability at least $1 - \delta$. Its running time is upper bounded by the complexities of the sampling procedure and the vertex cover black box procedure.

Proof. The majority of the analysis directly transfers from the proofs of Theorem 3.3.1 and Theorem 3.4.4. In Line 2 of Algorithm 6, we use a different threshold $d(\alpha) = 1/R + \epsilon$, which algebraic transformations prove to be the optimal choice.

Apart from that, the analysis only differs in the use of the approximated mandatory probabilities. We use a random estimation Y_v of p_v, instead of the precise probability, using the procedure described in Lemma 3.5.2 with parameters ϵ and δ' such that $1 - \delta = (1 - \delta')^n$.

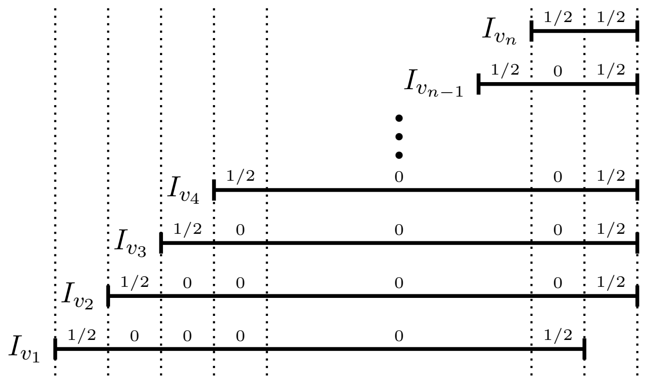


FIGURE 3.10: Instance of hypergraph orientation under explorable stochastic uncertainty with a single hyperedge $\{v_1, v_2, \ldots, v_n\}$ as used in the proof of Theorem 3.5.4.

As a result, with probability at least $1-\delta$, we have that for every vertex v, the estimation Y_v has absolute error at most ϵ. In case of this event, we obtain the following bound on the cost (which is optimized for the chosen value of d), namely $\mathbb{E}[\text{ALG}] \leq \max\{\frac{1}{d-\epsilon}, (1+d+\epsilon), (\alpha+(2-\alpha)(d+\epsilon))\} \cdot \mathbb{E}[\text{OPT}]$, by repeating the analysis of Theorem 3.3.1 with the approximated probabilities. □

3.5.3 Bounds on the Necessary Adaptivity

In contrast to the algorithm for graphs, the THRESHOLD algorithm for hypergraphs uses additional adaptivity after querying the vertex cover. We remark that this is necessary for all vertex cover-based algorithms. To show this, we define a *a strict two-stage algorithm* to be an algorithm with two stages that non-adaptively queries a set V_1 of vertices in the first stage and then, after receiving the answers to the queries of V_1, determines a set V_2 of vertices that it queries non-adaptively in the second stage. The algorithm must guarantee that, after the second stage, it has sufficient information to solve the problem. Our next theorem shows that strict two-stage algorithms cannot have a good competitive ratio.

Theorem 3.5.4. *No strict two-stage algorithm can have competitive ratio $o(\log n)$ for hypergraph orientation under explorable stochastic uncertainty, even for a single hyperedge with uniform query costs.*

Proof. Consider an instance of hypergraph orientation under stochastic explorable uncertainty with a single hyperedge $\{v_1, \ldots, v_n\}$ and uncertainty intervals

- $I_{v_1} = (1, n+1)$ and
- $I_{v_i} = (i, n+2)$ for $2 \leq i \leq n$.

The probability distributions are such that, for each uncertainty interval (a, b), the probability is $1/2$ for the precise weight to be in $(a, a+1)$, and $1/2$ for the precise weight to be in $(b-1, b)$. So the weight of each interval is either at its left end or at its right end, and each of these events happens with probability $1/2$. See Figure 3.10 for an illustration.

If we query the vertices in order of their left interval endpoints until the instance is solved, we are done as soon as one interval has its weight at its left end. So each vertex has probability $1/2$ to be the last one we need to query, and the expected number of queries is 2. So $\mathbb{E}[\text{OPT}] \leq 2$.

Now consider an arbitrary strict two-stage algorithm ALG. Assume that the algorithm queries k intervals in the first stage.

Algorithm 7: BESTVC

```
Input: Preprocessed instance of hypergraph orientation under explorable stochasitc
       uncertainty with hypergraph $H = (V, E)$ and $I_v$ and $p_v$ for each $v \in V$.
1 Compute a minimum-weight vertex cover $VC$ for $\bar{G}$ using weights $\bar{c}_v = (1 - p_v) \cdot c_v$;
2 Query $VC$;
3 for $S \in E$ do
4     while The orientation of $S$ is not known yet do
5         Query the leftmost vertex of $S$;
```

Consider the event E that each of the vertices queried by ALG in the first stage has its weight at the right end. The event E occurs with probability $1/2^k$ and, if it occurs, the algorithm must query all $n - k$ remaining vertices in the second stage. Otherwise, it could happen that all queries in the second stage also yield a weight at the right end of the corresponding interval, and then it is impossible to decide whether an unqueried vertex has minimum weight. Thus, the expected cost of ALG is

$$\mathbb{E}[\text{ALG}] = k + \frac{1}{2^k}(n-k) = \frac{n}{2^k} + k\left(1 - \frac{1}{2^k}\right)$$

If $k \geq (1/2)\log n$, the term $k(1 - 1/2^k)$ is in $\Omega(\log n)$. If $k < (1/2)\log n$, the term $n/2^k$ is in $\Omega(\sqrt{n})$. So, in any case, $\mathbb{E}[\text{ALG}] = \Omega(\log n)$ while $\mathbb{E}[\text{OPT}] \leq 2$. $\square$

3.6 Vertex Cover-Based Algorithms: Special Cases

In this section, we further characterize the, in expectation, best vertex cover-based algorithm and show that it achieves an improved competitive ratio for the problems of orienting a bipartite graph and orienting a single hyperedge with uniform query costs.

Consider an arbitrary vertex cover-based algorithm ALG. It queries a vertex cover VC in the first stage and continues with elements of $V \setminus VC$ only if they are mandatory. Thus,

$$\begin{aligned}
\mathbb{E}[\text{ALG}] &= c(VC) + \sum_{v \in V \setminus VC} p_v \cdot c_v \\
&= \sum_{v \in VC} (p_v \cdot c_v + (1 - p_v) \cdot c_v) + \sum_{v \in V \setminus VC} p_v \cdot c_v \\
&= \sum_{v \in V} p_v \cdot c_v + \sum_{v \in VC} (1 - p_v) \cdot c_v.
\end{aligned}$$

Since the term $\sum_{v \in V} p_v \cdot c_v$ is independent of VC, ALG is the best possible vertex cover-based algorithm if the vertex cover VC minimizes $\sum_{v \in VC}(1 - p_v) \cdot c_v$. We refer to this algorithm as BESTVC and formalize it with Algorithm 7.

To implement BESTVC, we need the exact value p_v, for all $v \in V$, and an optimal algorithm for computing a weighted vertex cover. As mentioned in Section 3.2, the first problem is #P-hard in hypergraphs, but it can be solved exactly in polynomial time for graphs. The weighted vertex cover problem can be solved optimally in polynomial time for bipartite graphs.

In general, BESTVC has competitive ratio at least 1.5 (cf. Theorem 3.2.13). However, we show in the following that it is $4/3$-competitive for two special cases: Orienting a bipartite graph with arbitrary query costs and orienting a single hyperedge with uniform query costs.

It remains open whether BESTVC still outperforms THRESHOLD if the vertex cover is only approximated with a factor $\alpha > 1$.

3.6.1 Orienting Bipartite Graphs with Arbitrary Query Costs

We start by considering BESTVC for orienting bipartite graphs. In particular, we prove the following theorem, which is tight due to Theorem 3.2.5.

Theorem 3.6.1. BESTVC *is $\frac{4}{3}$-competitive for the bipartite graph orientation problem under explorable stochastic uncertainty.*

Our proof of the theorem relies on reducing the problem to the special case of orienting a weighted star with probabilities such that both vertex cover-based algorithms for orienting the star (querying the center first and querying the leaves first, respectively) have equal expected cost. We prove the following lemma in the subsequent section and use it as a blackbox to show Theorem 3.6.1.

Lemma 3.6.2. *Consider the problem of orienting a star $G = (V, E)$ with center $v \in V$. Let L and R be the vertex cover-based algorithms that query $VC_1 = \{v\}$ and $VC_2 = V \setminus \{v\}$ in the first stage, respectively. If $\mathbb{E}[L] = \mathbb{E}[R]$, then L is $\frac{4}{3}$-competitive.*

We show now how to use this lemma as a blackbox in order to analyse BESTVC for bipartite graphs.

Proof of Theorem 3.6.1. Lemma 3.6.2 states that BESTVC is $\frac{4}{3}$-competitive for the problem of orienting stars if both vertex cover-based algorithms (either querying the leaves or the center first) have the same expected cost. In this proof, we divide a given bipartite instance into subproblems that fulfill these requirements, and use the result for stars to infer $\frac{4}{3}$-competitiveness for bipartite graphs.

Let VC be a minimum-weight vertex cover (with weights $\bar{c}_v = c_v \cdot (1 - p_v)$) as computed by BESTVC in the first phase. In the remainder of the proof, we use the term *weight of vertex* v to refer to $\bar{c}_v = (1 - p_v) \cdot c_v$. By the Kőnig-Egerváry theorem (e.g., [Sch03]), there is a function $\pi : E \to \mathbb{R}$ with $\sum_{\{u,v\} \in E} \pi(u, v) \leq c_v \cdot (1 - p_v)$ for each $v \in V$. By duality theory, the constraint is tight for each $v \in VC$, and $\pi(u, v) = 0$ holds if both u and v are in VC (or if there is no edge between u and v). Thus, we can interpret π as a function that distributes the weight of each $v \in VC$ to its neighbors outside of VC.

For each $v \in VC$ and $u \in V \setminus VC$, let $\lambda_{u,v} := \frac{\pi(u,v)}{(1-p_u) \cdot c_u}$ denote the fraction of the weight of u that is used by π to cover the weight of v. Moreover, for $u \in V \setminus VC$, let $\tau_u := 1 - \sum_{\{u,v\} \in E} \lambda_{u,v}$ be the fraction of the weight of u that is not used by π to cover the weight of any $v \in VC$. Then, we can write the expected query cost of BESTVC as follows:

$$\mathbb{E}[\text{BESTVC}] = \sum_{v \in VC} \left(c_v + \sum_{u \in V \setminus VC} p_u \cdot \lambda_{u,v} \cdot c_u \right) + \sum_{u \in V \setminus VC} p_u \cdot \tau_u \cdot c_u. \tag{3.6.1}$$

In order to bound this expected cost of BESTVC in terms of $\mathbb{E}[\text{OPT}]$, we first use λ and τ to define a fractional partition $(\mathcal{S}, f)$ as defined in Definition 3.4.2 of the vertex set V and use Lemma 3.4.3 to derive a lower bound on $\mathbb{E}[\text{OPT}]$. For each $v \in VC$, let $H_v = \{v\} \cup \{u \mid u \in V \setminus VC \text{ with } \lambda_{u,v} > 0\}$, and define $\mathcal{S} = \{H_v \mid v \in VC\} \cup \{V \setminus VC\}$. We complete the definition of the fractional partition $(\mathcal{S}, f)$ by defining f as

$$f(u, H_v) = \begin{cases} 1 & \text{if } u = v \\ \lambda_{u,v} & \text{if } u \in H_v \setminus \{v\} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$f(u, V \setminus VC) = \begin{cases} \tau_u & \text{if } u \in V \setminus VC \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, $(\mathcal{S}, f)$ is a fractional partition of V. By Lemma 3.4.3 and Equation (3.2.1), we get the following lower bound on $\mathbb{E}[\mathrm{OPT}]$:

$$\begin{aligned} \mathbb{E}[\mathrm{OPT}] &\geq \sum_{S \in \mathcal{S}} \mathbb{E}[\mathrm{OPT}_{S,f}] \\ &\geq \sum_{v \in VC} \mathbb{E}[\mathrm{OPT}_{H_v,f}] + \mathbb{E}[\mathrm{OPT}_{V \setminus VC,f}] \\ &\geq \sum_{v \in VC} \mathbb{E}[\mathrm{OPT}_{H_v,f}] + \sum_{u \in V \setminus VC} p_u \cdot f(u, V \setminus VC) \cdot c_u \\ &= \sum_{v \in VC} \mathbb{E}[\mathrm{OPT}_{H_v,f}] + \sum_{u \in V \setminus VC} p_u \cdot \tau_u \cdot c_u. \end{aligned}$$

Comparing Equation (3.6.1) with this lower bound on $\mathbb{E}[\mathrm{OPT}]$, we notice that the term $\sum_{u \in V \setminus VC} p_u \cdot \tau_u \cdot c_u$ shows up in both inequalities. Thus, to bound $\mathbb{E}[\textsc{BestVC}]$ in terms of $\mathbb{E}[\mathrm{OPT}]$, it only remains to bound $\sum_{v \in VC}(c_v + \sum_{u \in V \setminus VC} p_u \cdot \lambda_{u,v} \cdot c_u)$ in terms of $\sum_{v \in VC} \mathbb{E}[\mathrm{OPT}_{H_v,f}]$. We do so component-wise and compare $c_v + \sum_{u \in V \setminus VC} p_u \cdot \lambda_{u,v} \cdot c_u$ to $\mathbb{E}[\mathrm{OPT}_{H_v,f}]$ for each $v \in VC$.

The value $\mathbb{E}[\mathrm{OPT}_{H_v,f}]$ corresponds to the expected optimum for the subproblem which considers the subgraph induced by H_v with query costs $c'_u = f(u, H_v) \cdot c_u$ for all $u \in H_v$ and using the original mandatory probabilities p_u for all $u \in H_v$. The subgraph is a star with center v and leaves $H_v \setminus \{v\}$. Considering the two vertex cover-based algorithms L and R on this instance, where L is the algorithm that queries $VC_1 = \{v\}$ in the first stage and R is the algorithm that queries $VC_2 = H_v \setminus \{v\}$ in the first stage, we can observe that the expected costs of L and R are $c'_v + \sum_{u \in H_v \setminus \{v\}} p_u \cdot c'_u = c_v + \sum_{u \in H_v \setminus \{v\}} p_u \cdot \lambda_{u,v} \cdot c_u$ and $p_v \cdot c'_v + \sum_{u \in H_v \setminus \{v\}} c'_u = p_v \cdot c_v + \sum_{u \in H_v \setminus \{v\}} \lambda_{u,v} \cdot c_u$, respectively. By definition of λ, we have $(1 - p_v) \cdot c_v = \sum_{u \in H_v \setminus \{v\}}(1 - p_u) \cdot \lambda_{u,v} \cdot c_u$, which implies

$$\begin{aligned} c'_v + \sum_{u \in H_v \setminus \{v\}} p_u \cdot c'_u &= c_v + \sum_{u \in H_v \setminus \{v\}} p_u \cdot \lambda_{u,v} \cdot c_u \\ &= p_v \cdot c_v + \sum_{u \in H_v \setminus \{v\}} \lambda_{u,v} \cdot c_u \\ &= p_v \cdot c'_v + \sum_{u \in H_v \setminus \{v\}} c'_u. \end{aligned}$$

Thus, L and R have the same expected cost for the subproblem. This allows us to apply Lemma 3.6.2 and conclude that both, L and R, are $\frac{4}{3}$-competitive for the subproblem. This implies

$$c'_v + \sum_{u \in H_v \setminus \{v\}} p_u \cdot c'_u = c_v + \sum_{u \in H_v \setminus \{v\}} p_u \cdot \lambda_{u,v} \cdot c_u \leq \frac{4}{3} \cdot \mathbb{E}[\mathrm{OPT}_{H_v,f}].$$

We remark that applying the lemma requires p_v to be independent of each p_u with $u \in H_v$; otherwise, the subproblem does not correspond to the star orientation problem. As the input graph is bipartite, such independence follows by definition.

Using this inequality, the lower bound on $\mathbb{E}[\mathrm{OPT}]$ and Equation (3.6.1), we conclude that BESTVC is $4/3$-competitive:

$$\begin{aligned}\mathbb{E}[\textsc{BestVC}] &= \sum_{v\in VC}\Big(c_v + \sum_{u\in V\setminus VC} p_u\cdot\lambda_{uv}\cdot c_u\Big) + \sum_{u\in V\setminus VC} p_u\cdot\tau_u\cdot c_u\\ &\le \frac{4}{3}\cdot\sum_{v\in VC}\mathbb{E}[\mathrm{OPT}_{H_v,f}] + \sum_{u\in V\setminus VC} p_u\cdot\tau_u\cdot c_u\\ &\le \frac{4}{3}\cdot\mathbb{E}[\mathrm{OPT}].\end{aligned}$$

□

3.6.2 Orienting a Special Star with Arbitrary Query Costs

The proof of the main result of the previous section uses Lemma 3.6.2 as a blackbox. In this section, we proceed by proving the lemma.

Lemma 3.6.2. *Consider the problem of orienting a star $G = (V, E)$ with center $v \in V$. Let L and R be the vertex cover-based algorithms that query $VC_1 = \{v\}$ and $VC_2 = V \setminus \{v\}$ in the first stage, respectively. If $\mathbb{E}[L] = \mathbb{E}[R]$, then L is $\frac{4}{3}$-competitive.*

Proof. For each $v \in V$, let X_v be an indicator variable denoting whether v is mandatory. Note that $p_v = \mathbb{P}[X_v = 1]$. The analysis relies on X_v being independent from X_u for each $u \in V \setminus \{v\}$. In the graph orientation problem, a vertex u is mandatory if the exact weight of at least one v with $\{u, v\} \in E$ is contained in I_u. Thus, vertex u being mandatory depends only on the direct neighbors of u. Since we consider a start with center v and leaves $V \setminus \{v\}$, v does not share any neighbors with any $u \in V \setminus \{v\}$. Thus, X_v and X_u are independent. This also implies that X_v and L are independent. In the following let $B_v = V \setminus \{v\}$ denote the set of leaves. Thus, for the expected costs of the algorithms, we have

$$\mathbb{E}[L] = c_v + \sum_{u\in B_v} p_u\cdot c_u = p_v\cdot c_v + c(B_v) = \mathbb{E}[R]. \tag{3.6.2}$$

We start our analysis by handling a special case.

Special Case $c(B_v) \le c_v$. If $c(B_v) \le c_v$, querying the leaves is always the best strategy if the center is not mandatory, i.e., if $X_v = 0$, independently of whether the elements of B_v are mandatory or not. Similarly, if the center is mandatory, i.e., if $X_v = 1$, querying the center first is always the best strategy. Thus, we can write $\mathbb{E}[\mathrm{OPT}]$ as

$$\begin{aligned}\mathbb{E}[\mathrm{OPT}] &= p_v\cdot\mathbb{E}[L \mid X_v = 1] + (1-p_v)\cdot\mathbb{E}[R \mid X_v = 0]\\ &= p_v\cdot\mathbb{E}[L] + (1-p_v)\cdot c(B_v)\\ &= p_v\cdot\mathbb{E}[L] + (1-p_v)\cdot(\mathbb{E}[L] - p_v\cdot c_v)\\ &= \mathbb{E}[L] - p_v\cdot(1-p_v)\cdot c_v,\end{aligned}$$

where the second equality uses the fact that X_v and L are independent, and the third equality uses $\mathbb{E}[L] = \mathbb{E}[R]$. Since $0 \le p_v \le 1$, we get $p_v\cdot(1-p_v)\cdot c_v \le \frac{c_v}{4}$. This directly implies

$$\frac{\mathbb{E}[L]}{\mathbb{E}[\mathrm{OPT}]} = \frac{\mathbb{E}[L]}{\mathbb{E}[L] - p_v\cdot(1-p_v)\cdot c_v} \le \frac{c_v}{c_v - p_v\cdot(1-p_v)\cdot c_v} \le \frac{c_v}{c_v - \frac{1}{4}\cdot c_v} \le \frac{4}{3},$$

where the second inequality uses that $\mathbb{E}[L] \geq c_v$ and that the left fraction is maximized when $\mathbb{E}[L]$ is minimized. This implies that we achieve the desired competitive ratio for the special case $c(B_v) \leq c_v$. For the remainder of the analysis, we assume $c(B_v) > c_v$.

Characterizing $\mathbb{E}[L]$ and $\mathbb{E}[\mathrm{OPT}]$. We begin by characterizing $\mathbb{E}[L]$ in terms of conditional expectations using the law of total expectation:

$$\mathbb{E}[L] = p_v \cdot \mathbb{E}[L \mid X_v = 1] + (1 - p_v) \cdot \mathbb{E}[L \mid X_v = 0].$$

Since X_v and L are independent, it follows that $\mathbb{E}[L \mid X_v = 1] = \mathbb{E}[L]$. If the center is mandatory, i.e., $X_v = 1$, we know that both OPT and L have the same expected value. That is, $\mathbb{E}[\mathrm{OPT} \mid X_v = 1] = \mathbb{E}[L \mid X_v = 1]$.

Thus, we are more interested in the case when $X_v = 0$. For this case, we know that

$$\mathbb{E}[L \mid X_v = 0] = \mathbb{E}[L] = c_v + \sum_{u \in B_v} p_u \cdot c_u = \mathbb{E}[R] = p_v \cdot c_v + c(B_v).$$

But whether algorithm L is the best possible strategy depends on how much query cost of the leaves is mandatory. We have two cases. In the first case, where a realization satisfies $\sum_{u \in B_v}(1 - X_u) \cdot c_u \geq c_v$, querying the center first is a better strategy than querying the leaves first, even if the center is not mandatory. This is because $\sum_{u \in B_v}(1 - X_u) \cdot c_u$ is the query cost of leaves that are *not* mandatory. Every feasible query set has to query either v or all such leaves, but it is not necessary to query both (if v is also not mandatory). As the non-mandatory leaves are more expensive than v, querying the center instead is optimal. In the second case, where the realization satisfies $\sum_{u \in B_v}(1 - X_u) \cdot c_u < c_v$, querying the leaves is the better strategy if additionally $X_v = 0$, using the same argumentation as for the first case.

Consider the case where $\sum_{u \in B_v}(1 - X_u) \cdot c_u \geq c_v$. Recall that $c(B_v)$ denotes the total query cost of the leaves. Since algorithm L only queries the leaves that are mandatory, the query cost for L on the leaves is at most $c(B_v) - \sum_{u \in B_v}(1 - X_u) \cdot c_u \leq c(B_v) - c_v$. As L additionally queries the center v, the total query cost of L is at most $c(B_v)$. Thus, we can write the expected cost for this case as $\mathbb{E}[L \mid X_v = 0 \wedge \sum_{u \in B_v}(1 - X_u) \geq c_v] = c(B_v) - s$, where s with $c(B_v) \geq s \geq 0$ describes the slack between $c(B_v)$ and $\mathbb{E}[L \mid X_v = 0 \wedge \sum_{u \in B_v}(1 - X_u) \cdot c_u \geq c_v]$. Define $q_L := \mathbb{P}\left[\sum_{u \in B_v}(1 - X_u) \cdot c_u \geq c_v\right]$, then we can further characterize $\mathbb{E}[L \mid X_v = 0]$ by again using the total law of expectation:

$$\begin{aligned}
\mathbb{E}[L \mid X_v = 0] = q_L \cdot \mathbb{E}&\left[L \middle| X_v = 0 \wedge \sum_{u \in B_v}(1 - X_u) \cdot c_u \geq c_v\right] \\
&+ (1 - q_L) \cdot \mathbb{E}\left[L \middle| X_v = 0 \wedge \sum_{u \in B_v}(1 - X_u) \cdot c_u < c_v\right] \\
&= q_L \cdot (c(B_v) - s) + (1 - q_L) \cdot \mathbb{E}\left[L \middle| X_v = 0 \wedge \sum_{u \in B_v}(1 - X_u) \cdot c_u < c_v\right].
\end{aligned}$$

Recall, again, that if $\sum_{u \in B_v}(1 - X_u) \cdot c_u \geq c_v$, then L is optimal (as argued above). This implies that, if $q_L = 1$, then $\mathbb{E}[L] = \mathbb{E}[\mathrm{OPT}]$ and the lemma immediately follows. Thus, in the following, assume $q_L < 1$. This in particular means that we can safely divide by $(1 - q_L)$.

Using the above characterization of $\mathbb{E}[L \mid X_v = 0]$ and the fact that $\mathbb{E}[L \mid X_v = 0] = \mathbb{E}[L] = \mathbb{E}[R] = p_v \cdot c_v + c(B_v)$ holds by the independence of L and X_v, we can derive

$$\begin{aligned}
&p_v \cdot c_v + c(B_v) = q_L \cdot (c(B_v) - s) + (1 - q_L) \cdot \mathbb{E}\left[L \middle| X_v = 0 \wedge \sum_{u \in B_v} (1 - X_u) \cdot c_u < c_v\right] \\
\Leftrightarrow &(1 - q_L) \cdot \mathbb{E}\left[L \middle| X_v = 0 \wedge \sum_{u \in B_v} (1 - X_u) \cdot c_u < c_v\right] = p_v \cdot c_v + (1 - q_L) \cdot c(B_v) + q_L \cdot s \\
\Leftrightarrow &\mathbb{E}\left[L \middle| X_v = 0 \wedge \sum_{u \in B_v} (1 - X_u) \cdot c_u < c_v\right] = c(B_v) + \frac{1}{1 - q_L} \cdot p_v \cdot c_v + \frac{q_L}{1 - q_L} \cdot s.
\end{aligned}$$

Plugging in this equality into the previous characterization for $\mathbb{E}[L \mid X_v = 0]$, we get

$$\begin{aligned}
\mathbb{E}[L \mid X_v = 0] &= q_L \cdot (c(B_v) - s) + (1 - q_L) \cdot \mathbb{E}\left[L \middle| X_v = 0 \wedge \sum_{u \in B_v} (1 - X_u) \cdot c_u < c_v\right] \\
&= q_L \cdot (c(B_v) - s) + (1 - q_L) \cdot \left(c(B_v) + \frac{1}{1 - q_L} \cdot p_v \cdot c_v + \frac{q_L}{1 - q_L} \cdot s\right).
\end{aligned}$$

Finally, we use this, and again the independence of L and X_v, to characterize $\mathbb{E}[L]$:

$$\begin{aligned}
\mathbb{E}[L] &= p_v \cdot \mathbb{E}[L \mid X_v = 1] + (1 - p_v) \cdot \mathbb{E}[L \mid X_v = 0] \\
&= p_v \cdot \mathbb{E}[L] + (1 - p_v) \cdot \mathbb{E}[L \mid X_v = 0] \\
&= p_v \cdot \mathbb{E}[L] + (1 - p_v) \cdot q_L \cdot (c(B_v) - s) \\
&\quad + (1 - p_v) \cdot (1 - q_L) \cdot \left(c(B_v) + \frac{1}{1 - q_L} \cdot p_v \cdot c_v + \frac{q_L}{1 - q_L} \cdot s\right).
\end{aligned}$$

To characterize $\mathbb{E}[\text{OPT}]$, observe that the only case where OPT and L differ is the case where $X_v = 0$ and $\sum_{u \in B_v} (1 - X_u) \cdot c_u < c_v$. For that case, OPT queries only the leaves at cost $c(B_v)$. Thus, we can write $\mathbb{E}[\text{OPT}]$ as

$$\mathbb{E}[\text{OPT}] = p_v \cdot \mathbb{E}[L] + (1 - p_v) \cdot q_L \cdot (c(B_v) - s) + (1 - p_v) \cdot (1 - q_L) \cdot c(B_v),$$

where we also use that $c(B_v) - s$ is the expected optimal cost conditioned on $X_v = 0$ and $\sum_{u \in B_v} (1 - X_u) \cdot c_u \geq c_v$, as we already argued.

Upper bounding the competitive ratio. We now use the characterizations of $\mathbb{E}[L]$ and $\mathbb{E}[\text{OPT}]$ to show $\frac{4}{3}$-competitiveness. Consider the difference between $\mathbb{E}[L]$ and $\mathbb{E}[\text{OPT}]$, then

$$\begin{aligned}
\mathbb{E}[L] - \mathbb{E}[\text{OPT}] &= p_v \cdot \mathbb{E}[L] + (1 - p_v) \cdot q_L \cdot (c(B_v) - s) \\
&\quad + (1 - p_v) \cdot (1 - q_L) \cdot \left(c(B_v) + \frac{1}{1 - q_L} \cdot p_v \cdot c_v + \frac{q_L}{1 - q_L} \cdot s\right) \\
&\quad - p_v \cdot \mathbb{E}[L] - (1 - p_v) \cdot q_L \cdot (c(B_v) - s) - (1 - p_v) \cdot (1 - q_L) \cdot c(B_v) \\
&= (1 - p_v) \cdot (1 - q_L) \cdot \left(c(B_v) + \frac{1}{1 - q_L} \cdot p_v \cdot c_v + \frac{q_L}{1 - q_L} \cdot s\right) \\
&\quad - (1 - p_v) \cdot (1 - q_L) \cdot c(B_v) \\
&= (1 - p_v) \cdot (p_v \cdot c_v + q_L \cdot s).
\end{aligned}$$

We can use this difference to upper bound the competitive ratio by

$$\frac{\mathbb{E}[L]}{\mathbb{E}[\text{OPT}]} \leq \frac{\mathbb{E}[L]}{\mathbb{E}[L] - (1 - p_v) \cdot p_v \cdot c_v - (1 - p_v) \cdot q_L \cdot s}.$$

We continue by upper bounding $q_L \cdot s$. Consider again the characterization of $\mathbb{E}[L]$. Under the conditions $X_v = 0$ and $\sum_{v \in B_v}(1 - X_u) \cdot c_u < c_v$, we already showed the expected value of L is larger than $c(B_v)$ by $\frac{1}{1-q_L} \cdot p_v \cdot c_v + \frac{q_L}{1-q_L} \cdot s$. By Equation (3.6.2), the difference between $c(B_v)$ and the cost of L can never be larger than c_v. Thus,

$$\begin{aligned}
&\frac{1}{1-q_L} \cdot p_v \cdot c_v + \frac{q_L}{1-q_L} \cdot s \leq c_v \\
&\implies p_v \cdot c_v + q_L \cdot s \leq c_v \cdot (1 - q_L) \\
&\implies q_L \cdot s \leq c_v \cdot (1 - p_v) - c_v \cdot q_L \\
&\implies q_L \cdot (c_v + s) \leq c_v \cdot (1 - p_v) \\
&\implies q_L \leq c_v \cdot \frac{1 - p_v}{c_v + s}.
\end{aligned}$$

This implies

$$\begin{aligned}
\frac{\mathbb{E}[L]}{\mathbb{E}[\mathrm{OPT}]} &\leq \frac{\mathbb{E}[L]}{\mathbb{E}[L] - (1-p_v) \cdot p_v \cdot c_v - (1-p_v) \cdot q_L \cdot s} \\
&\leq \frac{\mathbb{E}[L]}{\mathbb{E}[L] - (1-p_v) \cdot p_v \cdot c_v - \frac{c_v \cdot (1-p_v)^2 \cdot s}{c_v + s}}.
\end{aligned}$$

Moreover, we can observe that this upper bound decreases for increasing $\mathbb{E}[L]$, so we only need a lower bound on $\mathbb{E}[L]$. In the characterization of $\mathbb{E}[L]$, the expected cost of L under the condition $X_v = 0$ and $\sum_{u \in B_v}(1 - X_u) \cdot c_u \geq c_v$ is $c(B_v) - s$. Since the cost of L can never be smaller than c_v as L starts by querying v, we get $c(B_v) - s \geq c_v$ and, thus, $c(B_v) \geq c_v + s$. So the expected cost of the algorithm is $\mathbb{E}[L] = \mathbb{E}[R] = c(B_v) + c_v \cdot p_v \geq c_v + s + p_v \cdot c_v$. By plugging this into the ratio we obtain:

$$\frac{\mathbb{E}[L]}{\mathbb{E}[\mathrm{OPT}]} \leq \frac{c_v + p_v \cdot c_v + s}{c_v + p_v \cdot c_v + s - (1-p_v) \cdot p_v \cdot c_v - \frac{c_v \cdot (1-p_v)^2 \cdot s}{c_v + s}}$$

We can now use

$$\frac{c_v \cdot (1-p_v)^2 \cdot s}{c_v + s} = \frac{c_v \cdot (1-p_v)^2 \cdot s}{c_v \cdot (1 + \frac{s}{c_v})} = \frac{(1-p_v)^2 \cdot s}{1 + \frac{s}{c_v}}$$

to write the ratio as

$$\begin{aligned}
\frac{\mathbb{E}[L]}{\mathbb{E}[\mathrm{OPT}]} &\leq \frac{c_v + c_v \cdot p_v + s}{c_v + p_v \cdot c_v + s - (1-p_v) \cdot p_v \cdot c_v - \frac{(1-p_v)^2 \cdot s}{1 + \frac{s}{c_v}}} \\
&= \frac{1 + p_v + \frac{s}{c_v}}{1 + p_v + \frac{s}{c_v} - (1-p_v) \cdot p_v - \frac{(1-p_v)^2 \cdot \frac{s}{c_v}}{1 + \frac{s}{c_v}}}.
\end{aligned}$$

Assume $c_v > 0$; we can do this w.l.o.g. because we can query free elements in a preprocessing step. Let $s' = \frac{s}{c_v}$. Then we have that

$$\frac{\mathbb{E}[L]}{\mathbb{E}[\mathrm{OPT}]} \leq \frac{1 + p_v + s'}{1 + p_v + s' - (1-p_v) \cdot p_v - \frac{(1-p_v)^2 \cdot s'}{1 + s'}} = f(p_v, s').$$

Showing $\frac{4}{3}$-competitiveness. To complete the proof, we show that $f(p_v, s') \le \frac{4}{3}$ for any $0 \le p_v \le 1$ and $s' \ge 0$. We can rewrite

$$\begin{aligned}
f(p_v, s') &= \frac{(1+s')(1+p_v+s')}{1+s'+(1+s')p_v+(1+s')s'-(1+s')p_v+(1+s')p_v^2-(1-p_v)^2\cdot s'} \\
&= \frac{(1+s')(1+p_v+s')}{1+s'+(1+s')s'+(1+s')p_v^2-(1-p_v)^2\cdot s'} \\
&= \frac{(1+s')(1+p_v+s')}{1+s'+s'+s'^2+p_v^2+s'p_v^2-(1-p_v)^2\cdot s'} \\
&= \frac{(1+s')(1+p_v+s')}{1+s'+s'+s'^2+p_v^2+s'p_v^2-s'+p_v s'-s'p_v^2} \\
&= \frac{(1+s')(1+p_v+s')}{1+s'+s'^2+p_v^2+2p_v s'} \\
&= \frac{(1+s')(1+p_v+s')}{1+s'+(p_v+s')^2}.
\end{aligned}$$

We fix the value of s' and look for critical points, so we consider the partial derivative of $f(p_v, s')$ on p_v, which is

$$\frac{\partial}{\partial p_v} f(p_v, s') = -\frac{(1+s')(p_v^2+2p_v(1+s')+s'^2+s'-1)}{((p_v+s')^2+1+s')^2}.$$

The denominator is clearly always greater than zero.

First, let us consider $s' \ge \frac{1}{\phi} = \frac{\sqrt{5}-1}{2}$. If $s'^2+s'-1 \ge 0$, which always holds for $s' \ge \frac{1}{\phi}$, then clearly $\frac{\partial}{\partial p_v} f(p_v, s') \le 0$, so $f(p_v, s')$ is non-increasing and is maximized when $p_v = 0$. Let $g(s') = f(0, s') = \frac{(1+s')^2}{1+s'+s'^2}$; then $\frac{d}{ds'} g(s') = \frac{1-s'^2}{(1+s'+s'^2)^2}$, and the only critical point is obtained by taking $\frac{d}{ds'} g(s') = 0$, which holds when $s' = 1$. It is clear that $\frac{d}{ds'} g(s') \ge 0$ for $0 \le s' \le 1$ and $\frac{d}{ds'} g(s') \le 0$ for $s' \ge 1$, so $g(s')$ has a global maximum when $s' = 1$. Thus, the maximum value of $f(p_v, s')$ for $s' \ge \frac{1}{\phi}$ and $p_v \ge 0$ is $f(0,1) = \frac{4}{3}$.

Now assume $0 \le s' < \frac{1}{\phi}$. Let us look at the critical points by taking $\frac{\partial}{\partial p_v} f(p_v, s') = 0$. Since the denominator is always positive and $s' \ge 0$, we only have a zero when

$$N(p_v, s') = p_v^2 + 2p_v(1+s') + s'^2 + s' - 1 = 0,$$

which by the quadratic formula yields $p_v = -1 - s' \pm \sqrt{s'+2}$. Since we need $p_v \ge 0$, we have $p_v = -1 - s' + \sqrt{s'+2}$. We claim that this is always a maximum point. Clearly, $\frac{\partial}{\partial p_v} f(p_v, s') \ge 0$ whenever $N(p_v, s') \le 0$ and vice-versa, but $\frac{\partial}{\partial p_v} N(p_v, s') = 2(1+p_v+s')$, which is non-negative for $p_v, s' \ge 0$, so $N(p_v, s')$ is non-decreasing. We can conclude that $\frac{\partial}{\partial p_v} f(p_v, s') \ge 0$ for $0 \le p_v \le -1 - s' + \sqrt{s'+2}$ and $\frac{\partial}{\partial p_v} f(p_v, s') \le 0$ for $-1 - s' + \sqrt{s'+2} \le p_v \le 1$, so we have a maximum point at $p_v = -1 - s' + \sqrt{s'+2}$. Finally, let

$$h(s') = f(-1 - s' + \sqrt{s'+2}, s') = \frac{(1+s')\sqrt{s'+2}}{2s'+4-2\sqrt{s'+2}}.$$

We have that $\frac{d}{ds'}h(s') = \frac{1}{4\sqrt{s'+2}}$, which is always non-negative for $s' \geq 0$. Thus, $h(s')$ is a non-decreasing function, so its maximum for $0 \leq s' < \frac{1}{\phi}$ is attained when s' tends to $\frac{1}{\phi}$. Therefore, for $0 \leq s' < \frac{1}{\phi}$ and $p_v \geq 0$, we have that $f(p_v, s') \leq h(\frac{1}{\phi}) = \frac{\sqrt{5}+3}{4} < \frac{4}{3}$. $\square$

3.6.3 Orienting a Single Hyperedge with Uniform Query Costs

Next, we analyze BESTVC for the special case of orienting a single hyperedge with uniform query costs.

Theorem 3.6.3. BESTVC *has a competitive ratio at most* $\min\{\frac{4}{3}, \frac{n+1}{n}\}$ *for the hypergraph orientation problem on a single preprocessed hyperedge with* $n+1 \geq 2$ *vertices and uniform query costs. For a hyperedge with only two vertices, the algorithm is* 1.207*-competitive.*

Proof. Let $S = \{v_0, v_1, \dots, v_n\}$ be the single hyperedge. We assume that $v_0, v_1, \dots, v_n$ are ordered by non-decreasing left endpoints of their uncertainty intervals $I_{v_0}, \dots, I_{v_n}$. That is, v_0 is the leftmost vertex of S and, by our assumptions in Section 3.2, all intervals $I_{v_1}, \dots, I_{v_n}$ intersect I_{v_0}, but none of them is contained in I_{v_0}.

In this setting, by Definition 3.2.6, the vertex cover instance is a star with center v_0 and leaves $v_1, \dots, v_n$. Thus, we only have to consider two vertex cover-based algorithms: the algorithm L that queries vertex cover $\{v_0\}$ in the first stage, and the algorithm R that queries $\{v_1, \dots, v_n\}$ in the first stage. Note that, while the vertex cover instance is a star, we cannot apply the analysis of Lemma 3.6.2 in Section 3.6.2: This is because that analysis requires the mandatory probabilities of v_0 and each v_i with $1 \leq i \leq n$ to be independent, which is not the case here.

If $n \geq 3$, it is sufficient to consider algorithm L. The algorithm simply queries the vertices in order of left endpoints, starting with v_0, until we can identify the minimum element. If v_0 is also queried by OPT, we have $L = \text{OPT}$. Otherwise, OPT must query all of $v_1, \dots, v_n$, and the competitive ratio is at most $\frac{n+1}{n} \leq \frac{4}{3}$. Since L is $\frac{4}{3}$-competitive in this case, so is BESTVC.

So it remains to consider the cases $n = 1$ and $n = 2$. For $n = 1$, the configuration of the intervals is shown in Figure 3.11. Here, p is the probability that the weight of v_0 is contained in I_1, and q is the probability that the weight of v_1 is contained in I_0. Note that in this case p and q correspond to the mandatory probabilities of v_1 and v_0.

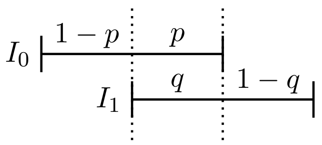


FIGURE 3.11: Configuration for a hyperedge with two vertices.

We can assume that $q \geq p$ (otherwise, we swap I_0 and I_1 and flip the x-axis). Since OPT has to query at least one vertex and only queries a second one if both are mandatory, we have $\mathbb{E}[\text{OPT}] = 1 + pq$. Algorithm L queries I_0 first and has $\mathbb{E}[L] = (1-p) \cdot 1 + p \cdot 2 = 1 + p$. The ratio $\frac{1+p}{1+pq}$ for $0 \leq p \leq 1$ and $p \leq q \leq 1$ is maximized for $q = p$, in which case it is $\frac{1+p}{1+p^2}$. The derivative of $f(p) = \frac{1+p}{1+p^2}$ is $f'(p) = \frac{1-2p-p^2}{(1+p^2)^2}$, which is positive for $0 \leq p < -1+\sqrt{2}$, equal to 0 for $p = -1+\sqrt{2}$, and negative for $1+\sqrt{2} < p \leq 1$. Hence, the maximum of $f(p)$ in the range $0 \leq p \leq 1$ is attained at $p_0 = -1+\sqrt{2} \approx 0.4142$ with value $f(p_0) = \frac{1+\sqrt{2}}{2} \approx 1.207$. Thus, BESTVC is 1.207-competitive for $n = 1$.

It remains to consider case $n = 2$. The configuration of the intervals is shown in Figure 3.12. Note that the order of the right endpoints of I_{v_1} and I_{v_2} is irrelevant for the proof.

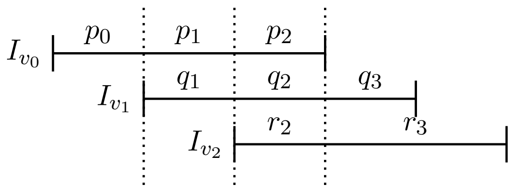


FIGURE 3.12: Configuration for three intervals.

Again, we only have to consider the two vertex cover-based algorithms L and R. We have:

$$\begin{aligned}\mathbb{E}[\mathrm{OPT}] &= p_0 \cdot 1 + p_1 \cdot 2 + p_2(q_1 \cdot 2 + q_2 \cdot 3 + q_3(r_2 \cdot 3 + r_3 \cdot 2)) \\ \mathbb{E}[L] &= p_0 \cdot 1 + p_1 \cdot 2 + p_2(q_1 \cdot 2 + q_2 \cdot 3 + q_3 \cdot 3) \\ \mathbb{E}[R] &= 3 - q_3 r_3\end{aligned}$$

By definition, BESTVC has an expected value of at most $\min\{\mathbb{E}[L], \mathbb{E}[R]\}$, and, thus, the competitive ratio is at most

$$\frac{\min\{\mathbb{E}[L], \mathbb{E}[R]\}}{\mathbb{E}[\mathrm{OPT}]}.$$

First, we want to remove some parameters to simplify our calculations. Note that if we move ε probability from q_2 to q_1, then $\mathbb{E}[\mathrm{OPT}]$ and $\mathbb{E}[L]$ decrease by $p_2\varepsilon$, while $\mathbb{E}[R]$ is unchanged. So we can assume that $q_2 = 0$. Furthermore, if we move ε probability from p_1 to p_0, then $\mathbb{E}[\mathrm{OPT}]$ and $\mathbb{E}[L]$ decrease by ε, while $\mathbb{E}[R]$ is unchanged. So we can assume that $p_1 = 0$. This gives

$$\begin{aligned}\mathbb{E}[\mathrm{OPT}] &= p_0 \cdot 1 + p_2(q_1 \cdot 2 + q_3(r_2 \cdot 3 + r_3 \cdot 2)) \\ \mathbb{E}[L] &= p_0 \cdot 1 + p_2(q_1 \cdot 2 + q_3 \cdot 3) \\ \mathbb{E}[R] &= 3 - q_3 r_3.\end{aligned}$$

Now we rename the parameters as

$$p := p_2, \quad 1 - p = p_0, \quad q := q_1, \quad 1 - q = q_3, \quad r := r_2, \quad 1 - r = r_3.$$

The equations then become

$$\begin{aligned}\mathbb{E}[\mathrm{OPT}] &= 1 + p + p(1 - q)r \\ \mathbb{E}[L] &= 1 + p(2 - q) \\ \mathbb{E}[R] &= 2 + r + q - qr.\end{aligned}$$

The corresponding picture with renamed parameters is shown in Figure 3.13.

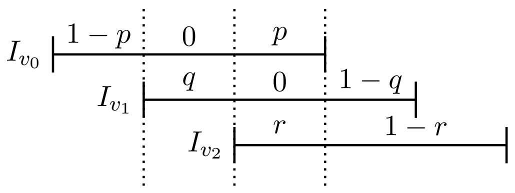


FIGURE 3.13: Configuration for three intervals assuming $p_1 = q_2 = 0$.

Case 1: $\mathbb{E}[L] \leq \mathbb{E}[R]$. We want to show

$$\mathbb{E}[L] \leq \frac{4}{3} \cdot \mathbb{E}[\mathrm{OPT}]. \tag{3.6.3}$$

This inequality (3.6.3) can be transformed as follows

$$\begin{aligned} 3 \cdot \mathbb{E}[L] &\leq 4 \cdot \mathbb{E}[\mathrm{OPT}] \\ \Leftrightarrow 3 + 3p(2-q) &\leq 4 + 4p + 4p(1-q)r \\ \Leftrightarrow 2p + 4pqr &\leq 1 + 4pr + 3pq. \end{aligned}$$

This is clearly satisfied for $p = 0$, so we can divide by p and get

$$2 + 4qr \leq \frac{1}{p} + 4r + 3q. \tag{3.6.4}$$

With respect to p, this inequality is most difficult to satisfy if p is as large as possible. To see how large p can be, we look back at our assumption for the current case, $\mathbb{E}[L] \leq \mathbb{E}[R]$. Expanding this assumption gives

$$1 + p(2-q) \leq 2 + r + q - qr.$$

This can be reformulated as

$$p \leq \frac{1 + r + q - qr}{2 - q}. \tag{3.6.5}$$

We now distinguish two cases depending on whether the right-hand side of (3.6.5) is larger than 1 or not.

Case 1.1: $\frac{1+r+q-qr}{2-q} \geq 1$. This condition is equivalent to

$$\begin{aligned} 1 + r + q - qr &\geq 2 - q \\ \Leftrightarrow r + 2q &\geq 1 + qr. \end{aligned} \tag{3.6.6}$$

In this case, we can set p to 1. Instead of inequality (3.6.4), it now suffices to show

$$\begin{aligned} 2 + 4qr &\leq 1 + 4r + 3q \\ \Leftrightarrow 1 + 4qr &\leq 4r + 3q. \end{aligned} \tag{3.6.7}$$

By (3.6.6) we have

$$1 + 4qr = 1 + qr + 3qr \leq r + 2q + 3qr.$$

Thus, to show (3.6.7) it suffices to show

$$\begin{aligned} r + 2q + 3qr &\leq 4r + 3q \\ \Leftrightarrow 3qr &\leq 3r + q. \end{aligned}$$

This is obviously satisfied as $3qr \leq 3r$ follows from $q \leq 1$.

Case 1.2: $\frac{1+r+q-qr}{2-q} < 1$. This condition is equivalent to

$$\begin{aligned} 1+r+q-qr &< 2-q \\ \Leftrightarrow r+2q &< 1+qr. \\ \Rightarrow 4qr+4q^2r^2 &> 4r^2q+8rq^2 \end{aligned} \tag{3.6.8}$$

In the last step, we have multiplied the inequality with $4qr$. In this case, we can set p to $\frac{1+r+q-qr}{2-q}$. Instead of Inequality (3.6.4) it now suffices to show

$$\begin{aligned} 2+4qr &\leq \frac{2-q}{1+r+q-qr}+4r+3q \\ \Leftrightarrow (1+r+q-qr)(2+4qr-4r-3q) &\leq 2-q \\ \Leftrightarrow 2-2r-q-5qr-4r^2-3q^2+7q^2r+8qr^2-4q^2r^2 &\leq 2-q \\ \Leftrightarrow 7q^2r+8qr^2 &\leq 2r+qr+4r^2+3q^2+(4q^2r^2+4qr). \end{aligned}$$

By (3.6.8), it suffices to show that

$$7q^2r+8qr^2 \leq 2r+qr+4r^2+3q^2+4r^2q+8rq^2.$$

This can be transformed into

$$4qr^2 \leq 2r+qr+4r^2+3q^2+rq^2.$$

This clearly holds because $4qr^2 \leq 4r^2$ follows from $q \leq 1$.

Case 2: $\mathbb{E}[L] \geq \mathbb{E}[R]$. We want to show

$$\mathbb{E}[R] \leq \frac{4}{3} \cdot \mathbb{E}[\mathrm{OPT}]. \tag{3.6.9}$$

This inequality can be transformed to

$$\begin{aligned} 6+3r+3q-3qr &\leq 4+4p+4p(1-q)r \\ \Leftrightarrow 2+3r+3q+4pqr &\leq 3qr+4p+4pr \\ \Leftrightarrow p(4+4r-4qr) &\geq 2+3r+3q-3qr, \end{aligned} \tag{3.6.10}$$

Expanding our assumption $\mathbb{E}[L] \geq \mathbb{E}[R]$ gives

$$2+r+q-qr \leq 1+p(2-q).$$

This can be reformulated as

$$\begin{aligned} 1+r+q+pq &\leq 2p+qr \\ \Leftrightarrow p(2-q) &\geq 1+r+q-qr \\ \Leftrightarrow p &\geq \frac{1+r+q-qr}{2-q}. \end{aligned} \tag{3.6.11}$$

We observe that this is only possible if the right-hand side of (3.6.11) is at most 1, so we must have the following for this to be possible.

$$\begin{aligned} 1+r+q-qr &\leq 2-q \\ r+2q &\leq 1+qr \end{aligned}$$

However, we will not need to use this fact in the remainder of the proof.

Inequality (3.6.10) is most difficult to satisfy if p is as small as possible, so in view of (3.6.11) we can set $p = \frac{1+r+q-qr}{2-q}$ and then prove the following inequality to establish (3.6.10).

$$\begin{aligned} \frac{1+r+q-qr}{2-q}(4+4r-4qr) &\geq 2+3r+3q-3qr \\ \Leftrightarrow 2r+5qr+4r^2+3q^2+4q^2r^2 &\geq 8qr^2+7q^2r \qquad (3.6.12) \end{aligned}$$

We show that (3.6.12) holds by showing the following inequalities

$$\begin{aligned} 5qr+4r^2 &\geq 8qr^2, \qquad (3.6.13) \\ 2r+3q^2+4q^2r^2 &\geq 7q^2r. \qquad (3.6.14) \end{aligned}$$

Inequality (3.6.13) holds because $5qr \geq 5qr^2$ and $4r^2 \geq 4qr^2$, so $5qr+4r^2 \geq 9qr^2 \geq 8qr^2$. To show (3.6.14), we distinguish two cases for r.

Case 2.1: $r \leq \frac{3}{5}$. In this case, $3q^2 \geq 5q^2r$ and hence $2r+3q^2 \geq 2q^2r+5q^2r = 7q^2r$. Thus, (3.6.14) holds.

Case 2.2: $r > \frac{3}{5}$. In this case, we have $4q^2r^2 \geq 4q^2r \cdot \frac{3}{5} \geq 2q^2r$, and hence $2r+3q^2+4q^2r^2 \geq 2q^2r+3q^2r+2q^2r = 7q^2r$, so (3.6.14) also holds in this case. □

The previous analysis improves upon a $(n+1)/n$-competitive algorithm by Chaplick et al. [Cha+21] in case that the hyperedge has two or three vertices. It is not hard to show a matching lower bound for two vertices and, due to Theorem 3.2.5, the algorithm also achieves the best possible competitive ratio for three vertices.

3.7 Concluding Remarks

In this chapter, we presented algorithms for the (hyper)graph orientation problem under stochastic explorable uncertainty and showed that the adversarial lower bounds can be broken in expectation. Open questions include determining the competitive ratio of BestVC for the general (hyper)graph orientation problem under explorable stochastic uncertainty, and investigating how the algorithm behaves if it has to rely on an α-approximation to solve the vertex cover subproblem. Our analysis suggests that, to achieve a competitive ratio better than 1.5, algorithms have to employ more adaptivity than vertex cover-based algorithms; exploiting this possibility remains an open problem.

Finally, it would be interesting to characterize the vertex cover instances arising in our Threshold algorithm. By definition of Threshold, no vertex in such instances has a very high mandatory probability. In the case of graph orientation, this means for every edge $\{u,v\}$ that a significant fraction of the probability mass in the distributions of u and v must be outside the intersection $I_u \cap I_v$ of the corresponding uncertainty intervals I_v and I_u. Since this is not possible for every combination of input graph and uncertainty intervals, it restricts the class of graphs that occur in the vertex cover instances of THRESHOLD. In addition to the relevance from a combinatorial point of view, such a characterization may allow an improved α-approximation algorithm for those instances.

Chapter 4

Sorting and Hypergraph Orientation under Uncertainty with Predictions

In this chapter, we consider learning-augmented algorithms for hypergraph orientation under explorable uncertainty and the special case of sorting a set of uncertainty intervals. Given a hypergraph with uncertain vertex weights, we study the problem of querying vertices until the identity of a vertex with minimum weight can be determined for each hyperedge. Since queries are costly, we aim at minimizing the number of queries. In the learning-augmented setting, we assume access to untrusted predictions for the uncertain vertex weights. Our algorithms provide improved performance guarantees for accurate predictions while maintaining worst-case guarantees that match the best possible guarantees without access to predictions. For hypergraph orientation, for any integral $\gamma \geq 2$, we give an algorithm that achieves a competitive ratio of $1 + 1/\gamma$ for correct predictions and γ for arbitrarily wrong predictions. For sorting, we achieve an optimal solution for accurate predictions while still being 2-competitive for arbitrarily wrong predictions. These tradeoffs are 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.

Bibliographic remark: This chapter is mainly based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+23] that will also appear in the proceedings of IJCAI 2023. Some results are based on a different joint work with the same group of authors [Erl+22; Erl+20]. Therefore, some parts correspond to or are identical with [Erl+23; Erl+22; Erl+20].

Contents

4.1 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 [LV21] for caching and has fostered an overwhelming number of results for numerous problems, including online graph problems [Kum+19; LMS22; Ebe+22; APT22] and scheduling problems [PSK18; Ang+20; Mit20; Lat+20; ALT21; ALT22; LM22; Bam+22; LX21]. In this and the consecutive chapter, we consider learning-augmented algorithms for problems under explorable uncertainty and start by designing learning-augmented algorithms for hypergraph orientation under explorable uncertainty.

To this end, we briefly restate the problem definition as initially given in Section 2.1.2 of Chapter 2. In the *hypergraph orientation problem under explorable uncertainty*, we are given a hypergraph $H = (V, E)$. Each vertex $v \in V$ is associated with an *uncertainty interval* I_v and an, initially unknown, *precise weight* $w_v \in I_v$. Each uncertainty interval I_v is either open or trivial, i.e., $I_v = (L_v, U_v)$ or $I_v = \{w_v\}$. We call L_v and U_v the *lower and upper limit* of v. If I_v is trivial, then we define $L_v = U_v = w_v$. A query of v comes at cost c_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 a 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 a 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 Q of minimum query cost $c(Q) = \sum_{v \in Q} c_v$. In this chapter, we only consider *uniform query costs*, i.e., we have $c_v = c_u$ for all $u, v \in V$. In this case, our objective is to minimize the *number of queries* and to find a feasible query set of minimum cardinality. See Section 2.1.2 for an example instance and feasible query set.

An important special case of the hypergraph orientation problem is when the input graph is a simple graph that is exactly the interval graph induced by the uncertainty intervals $\mathcal{I} = \{I_v \mid v \in V\}$, i.e., $\{u, v\} \in E$ for $u \neq v$ if and only if $I_v \cap I_u \neq \emptyset$. This special case corresponds to the problem of sorting a set of unknown values represented by a set of uncertainty intervals and, therefore, we refer to it as *sorting under uncertainty*.

As in the previous chapters, we analyze our algorithms for hypergraph orientation under explorable uncertainty in terms of their competitive ratio as defined in Section 2.2. In contrast to the previous chapter, we resort to *worst-case* competitive analysis. Recall that, for a given problem instance, OPT denotes an offline optimal feasible query set. In case of uniform query costs, an algorithm is *ρ-competitive* if it executes, for any problem instance, at most $\rho \cdot |\mathrm{OPT}|$ queries. The *competitive ratio* of an algorithm is the smallest ρ such that the algorithm is still ρ-competitive. 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 studied since the model of explorable uncertainty has been proposed [Kah91]. 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 [Kah91; HL21; Bam+21]; see also Sections 2.2.2 and 2.3.3. For sorting, the competitive ratio can be improved to 1.5

using randomization [HL21]. In case of general hypergraph orientation, the deterministic lower bound of 2 translates to a randomized lower bound of 1.5 but, to our knowledge, there are no randomized algorithms for that problem yet. 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 orienting a single hyperedge, as shown in Chapter 3. For the stochastic sorting problem, the algorithm with the optimal expected query cost is known, but its competitive ratio remains open [Cha+21].

In this chapter, 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 [LV21; PSK18] (see also Section 2.2.3): An algorithm is α-consistent if it is α-competitive when the predictions are correct, i.e., $\overline{w}_v = w_v$ for all $v \in V$, 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 of the previous chapter) that are more fine-grained than guarantees in the pure adversarial setting.

4.1.1 Our Results

We show how to utilize possibly erroneous predictions for hypergraph orientation and sorting under 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 transition between consistency and robustness for the sorting problem. However, for the general hypergraph orientation, we prove that this measure is too crude to allow for improvements upon the lower bound of 2 for the setting without predictions. We therefore introduce the *hop distance* k_h, a general error metric for problems under explorable uncertainty that takes the structure of the uncertainty intervals into account. Furthermore, we propose another 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 than k_h, and we show it to be more restrictive in the sense that $k_M \leq k_h$. We give formal definitions of the three error measures in Section 4.2. For the sorting problem, we obtain an algorithm with competitive ratio $\min\{1 + \frac{k}{|\mathrm{OPT}|}, 2\}$, 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 $\min\{(1+\frac{1}{\gamma-1})(1+\frac{k_M}{|\text{OPT}|}),\gamma\}$, 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 $\min\{(1+\frac{1}{\gamma})(1+\frac{k_h}{|\text{OPT}|}),\gamma\}$, for any integral $\gamma \geq 2$, which is also best possible for $k_h = 0$ and large k_h. While the consistency and robustness tradeoff 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.

The parameter γ of these algorithmic results can be thought of as a confidence parameter regulating how much the algorithm should trust the given predictions: If one expects predictions of high accuracy, then a large value of γ should be selected to exploit the expected high accuracy. On the other hand, if one expects unreliable predictions, then it might be better to select a small value of γ to maintain a strong robustness against bad predictions. Further, we note that the parameter γ is in both results restricted to integral values since it determines sizes of query sets, but a generalization to reals $\gamma \geq 2$ is possible via randomization 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 the NP-hardness of the offline version of the problem (cf. Section 3.2.6 in Chapter 3).

4.1.2 Outline

We start the chapter in Section 4.2 by restating some preliminary results on hypergraph orientation under explorable uncertainty that were already presented in Chapters 2 and 3, and extend them to the learning-augmented setting with access to predictions. Furthermore, we introduce and discuss several prediction errors that measure the quality of the given predictions. During the course of the first section, we also give lower bounds on the best possible consistency and robustness tradeoffs.

Afterwards, in Section 4.3, we consider learning-augmented algorithms for the general hypergraph orientation problem under explorable uncertainty with predictions and give two algorithms, one for each of the two main error measures we are considering, that achieve the best possible tradeoff with respect to their respective error metric. These algorithms combine the idea of the witness set algorithm (cf. Section 2.3.3) and the idea of vertex cover-based algorithms as introduced in the previous chapter (cf. Chapter 3). In order to achieve a consistency that improves upon the worst-case lower bound of two while still maintaining a good robustness, we extend these concepts by exploiting the additional information given by the predictions.

For the main algorithmic result of this chapter in Section 4.4, we consider sorting under explorable uncertainty with predictions and give a single algorithm achieving the optimal consistency and robustness tradeoff, while at the same time simultaneously matching the optimal error-dependency for all three error measures. Building on the ideas of the algorithms for the more general hypergraph orientation problem, we exploit the particular structure of sorting instances and employ a charging scheme based on a clique partition to achieve improved results for the special case.

Finally, in Section 4.5, we prove learnability results for the predictions with respect to the different error measures. To that end, we use the framework of PAC learnability [Val84] to show that we can approximately learn the predictions that in expectation minimize the respective error measure. We give more formal definitions in Section 4.5.

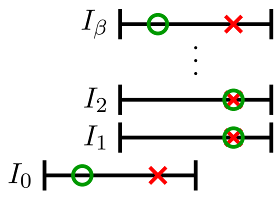


FIGURE 4.1: Lower bound example as used in the proof of Theorem 5.2.1. Shows the uncertainty intervals of an instance of hypergraph orientation under explorable uncertainty with prediction that consists of a single hyperedge with the vertices $\{0, 1, \ldots, \beta\}$. The green circles illustrate the precise weights while the red crosses illustrate the predicted weights.

4.2 Preliminaries, Tradeoff Lower Bounds and Error Measures

During this chapter, we consider hypergraph orientation under explorable uncertainty with predictions. So we are given a hypergraph $H = (V, E)$, uncertainty intervals I_v for all $v \in V$ *and* predicted weights $\overline{w}_v$ for all vertices v. As usual, our goal is to query vertices until we have sufficient information to orient each hyperedge and to minimize the number of queries. Since the predictions can be inaccurate, we still have to execute queries to guarantee that we find a correct orientation with respect to the precise weights. In contrast to the previous chapters, however, we now can use the predicted weights to select our query strategy.

An algorithm for the hypergraph orientation problem under explorable uncertainty with predictions that assumes the predicted weights to be completely accurate, i.e., $\overline{w}_v = w_v$ for all $v \in V$, can exploit the characterization of optimal solutions given in Lemma 3.2.9 to compute a query set that is optimal if the predicted weights are indeed correct. Querying this set leads to 1-consistency but may perform arbitrarily bad in case of incorrect predictions, as we show down below.

On the other hand, known 2-competitive algorithms for the adversarial problems without predictions [Kah91; HL21], as for example the witness set algorithm of Section 2.3.3, are not better than 2-consistent. Furthermore, the algorithms for the stochastic setting of Chapter 3 do not guarantee any robustness at all. Thus, we need new algorithms and techniques to achieve improved consistency and robustness tradeoffs. The lower bound of 2 on the adversarial competitive ratio without predictions (cf. Section 2.2.2) rules out any robustness factor less than 2 for our model, so we aim at matching this lower bound in terms of robustness.

To quantify the best possible consistency and robustness tradeoff, we give the following lower bound. This lower bound for example shows that 1-consistent algorithms cannot be better than n-robust and that 2-robust algorithms cannot be better than 1.5-consistent. This means that 1.5-consistency is best possible for algorithms which match the lower bound of 2 in terms of robustness. We remark that this lower bound does not hold for the sorting problem. In fact, we show an improved tradeoff for sorting under explorable uncertainty in Section 4.4.

Theorem 4.2.1. *Let $\beta \geq 2$ be a fixed integer. For hypergraph orientation under explorable uncertainty with predictions, 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 $\alpha = 1 + \frac{1}{\beta'}$ for some integer $\beta' \geq 1$ is β-robust for $\beta < \beta'$. The result holds even for orienting a single hyperedge or a simple (non-hyper) graph.*

Proof. Assume, for the sake of contradiction, that there is a deterministic β-robust algorithm that is α-consistent with $\alpha = 1 + \frac{1}{\beta} - \varepsilon$ for some $\varepsilon > 0$. Consider an instance with vertices $\{0, 1, \ldots, \beta\}$, a single hyperedge that contains all $\beta + 1$ vertices, and intervals and predicted weights as in Figure 4.1. If the predictions are correct, then vertex 0 has minimum weight in the hyperedge. In that case, the optimal query set is $\{1, \ldots, \beta\}$ as it is impossible to verify

$\overline{w}_0 = w_0 \leq w_i = \overline{w}_i$ for all $i \in \{1, \dots, \beta\}$ without querying all vertices in $\{1, \dots, \beta\}$ and querying those vertices suffices to identify 0 as the vertex of minimum precise weight. The algorithm must query the vertices $\{1, \dots, \beta\}$ first, as otherwise it would query $\beta + 1$ vertices in case all predictions are correct, while the optimal query set is of size β; a contradiction to the algorithm being α-consistent for an $\alpha < 1 + \frac{1}{\beta}$.

Suppose without loss of generality that the algorithm queries the vertices $\{1, \dots, \beta\}$ in increasing order. Consider the adversarial choice $w_i = \overline{w}_i$, for $i = 1, \dots, \beta - 1$, and then $w_\beta \in I_0$ and $w_0 \notin I_1 \cup \ldots \cup I_\beta$. This forces the algorithm to query also vertex 0 in order to prove $w_0 \leq w_\beta$, while an optimal solution only queries vertex 0. Thus, any such algorithm has robustness at least $\beta + 1$, a contradiction to the algorithm being β-robust. Figure 4.1 illustrates this adversarial choice.

The second part of the theorem directly follows from the first part and the known general lower bound of 2 on the competitive ratio [Hof+08; Kah91] (see also Section 2.2.2). Assume there is an α-consistent deterministic algorithm with $\alpha = 1 + \frac{1}{\beta'}$ for some integer $\beta' \geq 1$. If $\beta' < 2$, then the statement follows from the lower bound of 2, so assume $\beta' \geq 2$. Consider the instance above with $\beta = \beta' - 1$. Then the algorithm has to query vertices $\{1, \dots, \beta\}$ first to ensure α-consistency, as otherwise it would have a competitive ratio of $\frac{\beta+1}{\beta} > 1 + \frac{1}{\beta'} = \alpha$ in case that all predictions are correct. By the argumentation above, the robustness factor of the algorithm is at least $\beta + 1 = \beta'$.

To prove the result for the (non-hyper) graph orientation problem, the only difference is that we use edges $\{0, i\}$ for $1 \leq i \leq \beta$ instead of a single hyperedge containing all vertices. The rest of the proof remains the same. □

A main goal of this chapter is to design algorithms that match this optimal consistency and robustness tradeoff. To this end, we continue by introducing preliminary results that help us to achieve this goal.

4.2.1 Preliminaries

Recall that we say a vertex $v \in S$ is *leftmost* in hyperedge $S \in E$ if the uncertainty interval I_v of v has minimum lower limit among the intervals of the vertices in S, i.e., $L_v = \min_{u \in S} L_u$. If a hyperedge $S \in E$ contains a leftmost vertex v with a trivial uncertainty interval, then clearly no other vertex in S can have a smaller weight than v and we already know the orientation of S. Thus, we can assume without loss of generality that no hyperedge contains a leftmost vertex with a trivial uncertainty interval, since otherwise we could simply remove the hyperedge. Let v be a leftmost vertex in a hyperedge $S \in E$. Then we can also assume that $I_v \cap I_u \neq \emptyset$ for all $u \in S \setminus \{v\}$, because otherwise the vertex u could be removed from the hyperedge S since we would already know that $w_v < w_u$ and, therefore, that u cannot be of minimum weight in S. For the special case of graphs, this means that we assume $I_v \cap I_u \neq \emptyset$ for each $\{u, v\} \in E$, since otherwise we could simply remove the edge. The following assumption summarizes these observations. Note that we used the same assumptions in Chapter 3.

Assumption 3.2.1. *We assume without loss of generality that all problem instances for hypergraph orientation under explorable uncertainty satisfy the following properties:*

1. *No hyperedge $S \in E$ has a leftmost vertex v with a trivial uncertainty interval I_v.*
2. *If v is leftmost in a hyperedge S, then $I_v \cap I_u \neq \emptyset$ for all $u \in S \setminus \{v\}$.*

Basic Preliminaries The crucial structure and unifying concept in hypergraph orientation and sorting under explorable uncertainty without predictions are *witness sets* [Bru+05], as we have seen in Section 2.3.3. Witness sets are the key to any comparison with an optimal solution. Recall that 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 (cf. Section 2.3.3). We refer to witness sets of size two also as *witness pairs* and restate the following characterization of witness pairs; see Section 2.3.3 for a proof. To that end, recall that a hyperedge $S \in E$ is *not solved* if we do not know the orientation of S yet.

Lemma 2.3.7 (Kahan [Kah91])**.** *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. Let $S \in E$ be a not yet solved hyperedge of H. 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 this end, recall that a vertex is *mandatory* (cf. Section 2.3.2), if it is part of every feasible query set. We restate the following lemma that allows us to fully characterize mandatory vertices; see Section 2.3.3 for a proof.

Lemma 2.3.5. *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. A vertex $v \in V$ with a non-trivial uncertainty interval I_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 a minimum-weight vertex u of S.*

Prediction Mandatory Vertices While the lemma gives us a full characterization of mandatory vertices, it depends on the precise weights of the vertices, which are initially unknown. Identifying mandatory vertices based on the interval structure alone is not always possible, as otherwise there would be a 1-competitive algorithm contradicting the adversarial lower bound of 2.

In the learning-augmented setting however, we have access to additional information in form of the untrusted predictions $\overline{w}_v$ on the precise weights w_v. Using this additional information, we can identify vertices that are mandatory under the assumption that the predictions are correct, i.e., $w_v = \overline{w}_v$ for all $v \in V$. We call such vertices *prediction mandatory*.

Definition 4.2.2. *Given an instance of hypergraph orientation under explorable uncertainty with predictions with hypergraph $H = (V, E)$, uncertainty intervals I_v for all $v \in V$* and *predicted weights $\overline{w}_v \in I_v$ for all $v \in V$. A vertex $v \in V$ is* prediction mandatory *if it is mandatory for the realization of precise weights where all predictions are correct, i.e., $\overline{w}_v = w_v$ for all $v \in V$.*

For correct predictions, querying prediction mandatory vertices can never worsen the competitive ratio as even the optimal solution *must* query such vertices. However, in the lower bound example of Theorem 4.2.1, the vertices $1, \ldots, \beta$ are prediction mandatory and we have seen that querying those vertices leads to the worst possible robustness. One aspect of our algorithms will be to carefully balance the exploitation of prediction mandatory vertices with other methods that allow us to still achieve a good robustness. In order to do so, we have to identify prediction mandatory vertices. We can do this by using Lemma 2.3.5 with respect to the predicted weights instead of the precise weights:

Lemma 4.2.3. *Consider an instance of hypergraph orientation under explorable uncertainty with predictions. A vertex v with a non-trivial uncertainty interval I_v is prediction mandatory if and only if there is a hyperedge $S \in E$ with $v \in S$ such that either (i) v has minimum predicted weight in S and $\overline{w}_u \in I_v$ for some $u \in S \setminus \{v\}$, or (ii) v doesn't have minimum predicted weight in S and $\overline{w}_u \in I_v$ for vertex u with minimum predicted weight in S.*

Lemma 4.2.3 can be shown by using the proof of Lemma 2.3.5 and replacing the precise weights with the predicted weights. Since we have full access to the predicted weights $\overline{w}_v$, we can use this lemma to identify and compute all prediction mandatory vertices, which allows us to use such vertices within our algorithms.

Preprocessed Instances and the Offline Algorithm As also discussed in Chapter 3, the Lemma 2.3.5 does not only enable us to identify mandatory or prediction mandatory vertices given full knowledge of the precise or predicted 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. To identify such vertices, we restate the following corollary.

Corollary 2.3.6. *Consider an instance $H = (V, E)$ for hypergraph orientation under explorable uncertainty. If the interval of a leftmost vertex v in a not yet solved hyperedge S contains the precise weight of another vertex in S, then v is mandatory. In particular, if v is leftmost in (a not yet solved) S and $I_u \subseteq I_v$ for some $u \in S \setminus \{v\}$, then v is mandatory.*

Every algorithm can query vertices that are known to be mandatory according to Corollary 2.3.6 without ever worsening its competitive ratio. As in the previous chapter, we refer to instances that do not admit vertices that are mandatory by the corollary as *preprocessed*:

Definition 3.2.2. *An instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$ is* preprocessed*, if it does not contain vertices that can be identified as mandatory by using Corollary 2.3.6. That is, for every hyperedge $S \in E$, there is no leftmost vertex $v \in S$ with $I_u \subseteq I_v$ for an $u \in S \setminus \{v\}$.*

Our algorithms will heavily exploit the particular structure of preprocessed instances described by the the following lemma. For a proof of the lemma see Chapter 3.

Lemma 3.2.3. *Consider a preprocessed instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. Then, each hyperedge $S \in E$ satisfies the following properties:*

1. *S has a unique leftmost vertex v with a non-trivial uncertainty interval.*
2. *$I_u \cap I_v \neq \emptyset$ and $I_u \setminus I_v \neq \emptyset$ for all $u \in S \setminus \{v\}$.*

Using Lemma 2.3.5 and the structure of preprocessed instances, we 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 2.3.5. After that, the instance is preprocessed and 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 in this configuration is to query a minimum vertex cover in the *vertex cover instance* (see Chapter 3 for an example):

Definition 3.2.6. *Consider an instance of hypergraph orientation under explorable uncertainty with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$. The* vertex cover instance *of H is the graph $\bar{G} = (V, \bar{E})$ with $\{v, u\} \in \bar{E}$ if and only if there is a not yet solved hyperedge $S \in E$ such that $v, u \in S$, v is leftmost in S and $I_v \cap I_u \neq \emptyset$. For the special case of a graph G instead of a hypergraph H, it holds that $\bar{G} = G$.*

See Algorithm 8 for pseudocode of the offline algorithm. For a proof of its optimality, we refer to the proof of Lemma 3.2.9 in Chapter 3.

Algorithm 8: Offline algorithm for hypergraph orientation under explorable uncertainty.

```
  Input: Hypergraph $H = (V, E)$, intervals $I_v$ and precise weights $w_v$ for all $v \in V$.
1 $M \leftarrow$ All vertices that are mandatory by Lemma 2.3.5;
2 Query $M$ ;                          /* First stage of the algorithm */
3 $\bar{G}[V \setminus M] \leftarrow$ Vertex cover instance of $H$ for the instance after querying $M$;
4 $VC \leftarrow$ Minimum cardinality vertex cover of $\bar{G}[V \setminus M]$;
5 Query $VC$ ;                        /* Second stage of the algorithm */
6 return $Q^* = M \cup VC$;
```

A key idea of our algorithms is to emulate the offline algorithm using the predicted information. We can use the algorithm under the assumption that the predicted weights are correct and compute an optimal query set for correct predictions. Since blindly following the offline algorithm might lead to a competitive ratio of n for faulty predictions (cf. Theorem 4.2.1), 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. For a proof we again refer to Chapter 3.

Lemma 3.2.8. *Given a preprocessed instance of hypergraph orientation with hypergraph $H = (V, E)$ and uncertainty intervals I_v for all $v \in V$, let Q be an arbitrary vertex cover of $\bar{G}$. After querying Q, for each hyperedge $S \in E$, we either know the orientation of S or can determine it by exhaustively querying according to Corollary 2.3.6.*

4.2.2 Accuracy of Predictions

Since 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. Natural candidates for such measures include the number of inaccurate predictions $k_\# = |\{v \in V \mid w_v \neq \overline{w}_v\}|$ or an ℓ_1 error metric such as $k_{\ell_1} = \sum_{v \in V} |w_v - \overline{w}_v|$. These measures, however, are not meaningful for the hypergraph orientation problem under explorable uncertainty with predictions. For the number of inaccurate predictions, we can show that even for $k_\# = 1$, the competitive ratio cannot be better than the known bound of 2. Similar, we can show that if $k_{\ell_1} > 0$, then no deterministic algorithm can have a competitive ratio better than 2. Both results hold for instances with any even number of vertices $n \geq 2$ and $|\mathrm{OPT}| \in \Omega(n)$. Thus, even for arbitrarily large instances, the smallest possible (non-zero) error $k_\#$ or k_{ℓ_1} already leads to a competitive ratio of 2. These results prohibit any more fine-grained competitive ratios depending on $k_\#$ or k_{ℓ_1}. We note that the lower bound for $k_\#$ does not hold for interval graphs (the sorting problem).

Theorem 4.2.4. *If $k_\# \geq 1$ or $k_{\ell_1} > 0$, then any deterministic algorithm for the hypergraph orientation problem under uncertainty with predictions has competitive ratio $\rho \geq 2$. This result holds even for instances with $|\mathrm{OPT}| \in \Omega(n)$ and an arbitrarily large even number of vertices n, and even for (non-hyper) graphs.*

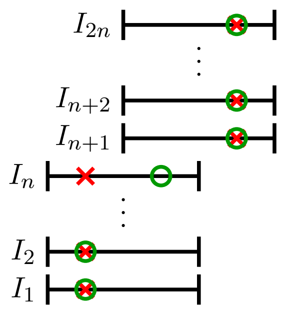


FIGURE 4.2: Uncertainty intervals and predicted and precise weights for the instance used in the proof of Theorem 4.2.4.

Proof. We first show the lower bound for $k_\# \geq 1$. Consider a hypergraph with vertices $\{1, \dots, 2n\}$, hyperedges $S_i = \{i, n+1, n+2, \dots, 2n\}$, for $i = 1, \dots, n$, and intervals and predicted weights as depicted in Figure 4.2.

Assume w.l.o.g. that the algorithm queries the vertices $\{1, \dots, n\}$ in the order $1, 2, \dots, n$ and the vertices $\{n+1, \dots, 2n\}$ in the order $n+1, n+2, \dots, 2n$. Before the algorithm queries vertex n or $2n$, the adversary sets all predictions as correct, so the algorithm will eventually query n or $2n$. If the algorithm queries n before $2n$, then the adversary chooses a weight $w_n \in \bigcap_{1 \leq j \leq n} I_{n+j}$ for n that forces a query to all vertices $n+1, \dots, 2n$. Furthermore, the adversary picks the weights of the vertices $n+1, \dots, 2n$ as equal to their predictions. Thus, we have $k_\# = 1$ and the algorithm queries $2n$ vertices while the optimal solution only queries the vertices $n+1, \dots, 2n$. See Figure 4.2 for an illustration. A symmetric argument holds if the algorithm queries vertex $2n$ before vertex n: In this case, the adversary selects $w_{2n} \in \bigcap_{1 \leq j \leq n} I_j$ and all other predicted weights are correct. Then, we again have $k_\# = 1$ and the algorithm queries $2n$ vertices while the optimal solution only queries the vertices $1, \dots, n$.

For (non-hyper) graphs, we use edges $\{i, j\}$ for all $1 \leq i \leq n, n+1 \leq j \leq 2n$ instead of hyperedges. The rest of the proof remains the same, which concludes the proof for $k_\# \geq 1$.

Next, we show the part of the theorem with regard to $k_{\ell_1} > 0$. We use the same lower bound instance as before but slightly adjust it: We move the predicted weights of the vertices $1, \dots, n$ arbitrarily close to the lower interval borders of the vertices $n+1, \dots, 2n$ such that they are still outsides of those intervals. Symmetrically, we move the predicted weights of the vertices $n+1, \dots, 2n$ arbitrarily close to the upper interval border of the vertices $1, \dots, n$ such that they are still outsides of those intervals. If the algorithm queries vertex n before $2n$, then the adversary picks $w_n \in \bigcap_{1 \leq j \leq n} I_{n+j}$ arbitrarily close to the lower interval borders of the vertices $n+1, \dots, n$ such that w_n is still contained in those intervals. Symmetrically, if the algorithm queries vertex $2n$ before n, then the adversary picks $w_{2n} \in \bigcap_{1 \leq j \leq n} I_j$ arbitrarily close to the upper interval borders of the vertices $1, \dots, n$ such that w_n is still contained in those intervals. All other weights are chosen according to their predictions. In both cases, we have an arbitrarily small $k_{\ell_1} > 0$ and the algorithm queries $2n$ vertices while the optimal solution only queries n vertices. The latter can be shown with the same arguments as in the proof for $k_\# \geq 1$. □

The lower bound example of Theorem 4.2.4 illustrates that more fine-grained competitive ratios depending on $k_\#$ or k_{ℓ_1} are not possible, as even a small error prevents an improvement over the lower bound of 2 for algorithms without access to predictions. The intuitive reason for this is that both measures completely ignore the structure of the uncertainty intervals, which is crucial for the feasibility of query sets. Thus, we need more refined measures that take the interval structure into account.

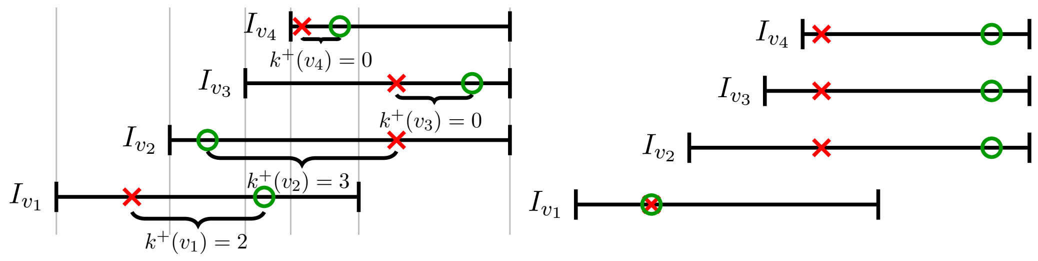


FIGURE 4.3: Example for the error measures k_h and k_M for the hypergraph orientation problem with a single hyperedge $S = \{v_1, v_2, v_3, v_4\}$. Circles illustrate precise weights and crosses illustrate the predicted weights. Shows predictions and precise weights with a total hop distance of $k_h = 5$ and mandatory query distance of $k_M = 1$ (left) and $k_h = 3$ and $k_M = 1$ (right).

As a first refined measure, we consider the *hop distance*. It is very intuitive even though it requires some technical care to make it precise. If we consider only a single predicted weight $\overline{w}_v$ for some $v \in V$, then, in a sense, this value predicts the relation of the precise weight w_v to the intervals of vertices $u \in V \setminus \{v\}$. In particular, w.r.t. a fixed $u \in V \setminus \{v\}$, the value $\overline{w}_v$ predicts whether w_v is left of I_u ($\overline{w}_v \leq L_u$), right of I_u ($\overline{w}_v \geq U_u$), or contained in I_u ($L_u < \overline{w}_v < U_u$). 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 4.3.

Note that $k_\# = 0$ implies $k_h = 0$, so Theorem 5.2.1 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 the feasibility of a query set. Therefore, 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., $k_M = |\mathcal{I}_P \Delta \mathcal{I}_R| = |(\mathcal{I}_P \cup \mathcal{I}_R) \setminus (\mathcal{I}_P \cap \mathcal{I}_R)| = |(\mathcal{I}_P \setminus \mathcal{I}_R) \cup (\mathcal{I}_R \setminus \mathcal{I}_P)|$. Intuitively, the error measure k_M captures differences between the sets of feasible query sets for the predicted weights and the precise weights: If there is a vertex $v \in \mathcal{I}_P \setminus \mathcal{I}_R$, then all feasible query sets for correct predictions contain v while there exist feasible query sets for the precise weights that do not contain v (and vice versa for $v \in \mathcal{I}_R \setminus \mathcal{I}_P$).

Figure 4.3 (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 is mandatory by Lemma 2.3.5 and, therefore, $\mathcal{I}_P = \{v_1\}$. It implies $k_M = |\mathcal{I}_P \Delta \mathcal{I}_R| = 1$.

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. In contrast, we can define different problems on the same set of uncertainty intervals and the error k_h remains the same. For instances of hypergraph orientation, we can relate k_M to k_h.

Theorem 4.2.5. *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., $k_M \leq k_h$.*

Proof. Consider an instance of hypergraph orientation with hypergraph $H = (V, E)$ and uncertainty intervals $\mathcal{I} = \{I_v \mid v \in V\}$, precise weights w and predicted weights $\overline{w}$. Recall that $\mathcal{I}_P$ and $\mathcal{I}_R$ are the sets of prediction mandatory elements and mandatory elements, respectively. Observe that k_M counts the vertices that are in $\mathcal{I}_P \setminus \mathcal{I}_R$ and those that are in $\mathcal{I}_R \setminus \mathcal{I}_P$. We will show the claim that, for every vertex v in those sets, there is a vertex u such that the weight of u passes over L_v or U_v (or both) when going from $\overline{w}_v$ to w_v, i.e., $w_u \leq L_v < \overline{w}_u$, $\overline{w}_u \leq L_v < w_u$, $w_u < U_v \leq \overline{w}_u$ or $\overline{w}_u < U_v \leq w_u$. This means that each vertex $v \in \mathcal{I}_P \Delta \mathcal{I}_R$ is mapped to a unique pair (u, v) such that the weight of u passes over at least one endpoint of I_v, and hence each such pair contributes at least one to the hop distance k_h. This implies $k_M \leq k_h$.

It remains to prove the claim. Consider an $v \in \mathcal{I}_P \setminus \mathcal{I}_R$. (The argumentation for intervals in $\mathcal{I}_R \setminus \mathcal{I}_P$ is symmetric, with the roles of w and $\overline{w}$ exchanged.) As v is not in $\mathcal{I}_R$, replacing all intervals for vertices in $\mathcal{I} \setminus \{v\}$ by their precise weights yields an instance that is solved. This means that in every hyperedge $S \in E$ that contains v, one of the following cases holds:

(a) v is known not to be the minimum of S w.r.t. precise weights w. This means that there is a vertex u in S with $w_u \leq L_v$.

(b) v is known to be the minimum of S w.r.t. precise weights w. This means that all vertices $u \in S \setminus \{v\}$ satisfy $w_u \geq U_v$.

As v is in $\mathcal{I}_P$, replacing all intervals of vertices in $V \setminus \{v\}$ by their predicted weights yields an instance that is not solved. This means that there exists at least one hyperedge $S' \in E$ that contains v and satisfies the following:

(c) All vertices u in $S' \setminus \{v\}$ satisfy $\overline{w}_u > L_v$, and there is at least one such u with $L_v < \overline{w}_u < U_v$.

If S' falls into case (a) above, then by (a) there is a vertex u in $S' \setminus \{v\}$ with $w_u \leq L_v$, and by (c) we have $\overline{w}_u > L_v$. This means that the weight of u passes over L_v. If S' falls into case (b) above, then by (c) there exists an vertex u in $S' \setminus \{v\}$ with $\overline{w}_u < U_v$, and by (b) we have $w_u \geq U_v$. Thus, the weight of u passes over v. This establishes the claim, and hence we have shown that $k_M \leq k_h$ for the hypergraph orientation problem. □

4.3 Hypergraph Orientation

We consider the general hypergraph orientation problem, and design learning-augmented algorithms with respect to the error measures k_h and k_M. For error measure k_h, we give an algorithm that matches the lower bound on the optimal consistency and robustness tradeoff of Theorem 4.2.1 and gracefully degrades between consistency and robustness with a linear error dependency on k_h.

To match the lower bound on the consistency and robustness tradeoff, we, for a given $\gamma \in \mathbb{N}_{\geq 2}$, have to guarantee $(1 + \frac{1}{\gamma})$-consistency and γ-robustness. As we observed before, following the offline algorithm based on the predicted information can lead to an arbitrarily bad robustness while using just the witness set algorithm will not improve upon 2-consistency. The idea of our algorithm is to emulate the offline algorithm using the predicted information and to combine it with the witness set algorithm. To illustrate this idea, assume $\gamma = 2$, so we have to guarantee 1.5-consistency and 2-robustness. To combine both algorithms, we consider *strengthened witness sets*, which are sets $W \subseteq V$ of size three such that every feasible solution queries at least *two* members of W. If we repeatedly query strengthened witness sets, then we achieve a competitive ratio of at most 1.5, matching our target consistency. Clearly, strengthened witness sets cannot always be identified based on the given graph and intervals alone. If we always could identify a strengthened witness set when the instance is not solved yet, then we would have a 1.5-competitive algorithm by repeatedly querying such sets, contradicting the lower bound of 2 on the adversarial competitive ratio (cf. Theorem 2.2.2). Therefore, we have to identify strengthened witness sets based on the predicted information, i.e., we identify sets W of cardinality three such that each feasible solution contains at least two members of the set *if* the predictions of the elements in W are correct. Since we can only identify strengthened witness sets based on the predicted information, we cannot afford to just query the complete set, as we might lose the guarantee on the set if the predictions are faulty, which could violate our target 2-robustness. To that end, we query such sets in a carefully selected order that allows us to detect errors that might cause us to violate the 2-robustness after at most two queries within the set. We select the first two queries within the set in such a way that they form a witness set. So even if there is an error within these queries, we can discard the third query and the two queries we already executed will never violate 2-robustness as they form a witness set. Furthermore, we will show that we can charge all executed queries that violate the consistency bound to a distinct error.

Our algorithm does this repeatedly until we cannot identify strengthened witness sets anymore, not even by using the predictions. After that, the instance has a certain structure that allows us to solve it with an adjusted second phase of the offline algorithm while achieving the optimal consistency and robustness tradeoff of Theorem 4.2.1 with linear error dependency.

If $\gamma > 2$, then the first phase of our algorithm repeatedly identifies a strengthened witness set *and* $\gamma - 2$ prediction mandatory vertices. It then queries the $\gamma - 2$ prediction mandatory vertices and afterwards proceeds to query the strengthened witness sets as described above. We show that this adjustment allows us to achieve the optimal tradeoff with linear dependency on k_h for every integral $\gamma \geq 2$.

With respect to error measure k_M, we first show that a consistency and robustness tradeoff matching Theorem 4.2.1 is not possible with linear error dependency on k_M. Instead, we design an algorithm with a slightly worse consistency and robustness tradeoff but with a linear error dependency on k_M. The tradeoff is best possible for algorithms with linear error dependency on k_M. Since $k_M \leq k_h$, the competitive ratio of this second algorithm can be better than the ratio of the k_h-dependent algorithm for certain instances. We achieve the k_M-dependent bound by giving a simpler algorithm that emulates the offline algorithm and augments it with additional queries. 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 $\overline{w}$.

4.3.1 Learning-augmented Algorithm With Respect To the Hop Distance

We start by designing an algorithm that achieves the optimal consistency and robustness tradeoff (cf. Theorem 4.2.1) with a linear error dependency on the hop distance k_h. The following theorem summarizes our main result with respect to error measure k_h.

Theorem 4.3.1. *There is an algorithm for hypergraph orientation under explorable uncertainty with predictions that, given $\gamma \in \mathbb{N}_{\geq 2}$, achieves a competitive ratio of* $\min\{(1 + \frac{1}{\gamma})(1 + k_h/|\mathrm{OPT}|), \gamma\}$.

As mentioned in the introduction of this section, our algorithm is based on identifying strengthened witness sets using the predicted weights, i.e., sets $W \subseteq V$ with $|W| = 3$ such that every feasible solution contains at least two elements of W if the predictions are accurate. We want to be able to query W in such a way that (i) the first two queries in W are a witness set and (ii) after the first two queries in W we either have detected a prediction error or can guarantee that each feasible solution indeed contains two elements of W (no matter if the predicted weights of vertices outside W are correct or not).

To achieve this, we identify prediction mandatory vertices that are not only mandatory if all predicted weights are correct but become mandatory if a *single* predicted weight is correct. To that end, we use the following definition.

Definition 4.3.2. *We say that a predicted weight $\overline{w}_u$* enforces *another vertex v if u and v have non-trivial uncertainty intervals, $\overline{w}_u \in I_v$, and $u, v \in S$, where S is a hyperedge such that either v is leftmost in S, or u is leftmost in S and v is leftmost in $S \setminus \{u\}$.*

If the predicted weight $\overline{w}_u$ of a vertex u enforces another vertex v and the predicted weight of u is accurate, then after querying u we know for sure that v is indeed mandatory. The following lemma formulates this property.

Lemma 4.3.3. *If $\overline{w}_u$ enforces v, then $\{v, u\}$ is a witness set. Also, if $w_u \in I_v$, then v is mandatory.*

Proof. Since $\overline{w}_u$ enforces v, there must be a hyperedge S with $v, u \in S$ such that $\overline{w}_u \in I_v$ and either v is leftmost in S or u is leftmost in S and v is leftmost in $S \setminus \{u\}$. The first claim of the lemma follows from Lemma 2.3.7 as one of u, v is leftmost in S and $I_v \cap I_u \neq \emptyset$.

Consider the second claim of the lemma. If v has minimum precise weight in S or is leftmost in S, then the second claim follows from Lemma 2.3.5 and Corollary 2.3.6. Otherwise, the fact that $w_u \in I_v$ and that v is leftmost in $S \setminus \{u\}$ implies that I_v contains the minimum precise weight, so the claim follows from Lemma 2.3.5. □

We can now define our Algorithm 9. Within the definition of the algorithm, the *current instance* always refers to the problem instance obtained after executing all previous queries. We say that a vertex is prediction mandatory for the current instance, if it is mandatory if the predicted weight of all *not yet queried* vertices are correct. Note that there can be vertices that are prediction mandatory for the current instance but not prediction mandatory for the initial instance (and vice versa). The current vertex cover instance refers to the vertex cover instance of the current instance.

The algorithm ensures that the current instance always remains preprocessed by exhaustively querying vertices that are mandatory by Corollary 2.3.6 (cf. Lines 1, 7 and 13). The Lines 9 to 12 identify and query strengthened witness sets. They do so by identifying a witness set $\{u, w\}$ such that the predicted weight $\overline{w}_u$ enforces another vertex v. Only if $w_u \in I_v$, the algorithm also queries v. If that is the case, then Lemma 4.3.3 implies that v is mandatory and, therefore, every feasible solution must query at least two members of $\{u, v, w\}$. If such a triple $\{u, v, w\}$ of vertices does not exist but there still is a vertex u such that $\overline{w}_u$ enforces a vertex v, then the algorithm just queries v in Line 12. We will prove that this never violates the target consistency or robustness. If $\gamma > 2$, then the algorithm precedes this step by querying (up-to) $\gamma - 2$ predictions mandatory vertices in Lines 4 to 8. The algorithm does this repeatedly as long as possible and afterwards queries a minimum vertex cover of the current vertex cover instance in Line 15. Exploiting Lemma 3.2.8, the remaining instance can then be solved by exhaustively querying vertices that are mandatory due to Corollary 2.3.6 (cf. Line 16).

We proceed by proving three more important properties of the algorithm that will help us to prove Theorem 4.3.1.

The Lemma 4.3.3 implies that if $\overline{w}_u$ enforces v and the predicted value of u is correct, then v is mandatory. This directly implies that v is prediction mandatory for the current instance: If the predicted weight of all not yet queried vertices are correct, then the predicted weight of u is correct and v is mandatory. This leads to the following corollary.

Corollary 4.3.4. *Consider a point of execution of the algorithm in which a predicted weight $\overline{w}_u$ enforces another vertex v. Then v is prediction mandatory for the current instance.*

Algorithm 9: Learning-augmented algorithm for the hypergraph orientation problem under explorable uncertainty with respect to the hop distance k_h

```
Input: Hypergraph $H = (V, E)$, intervals $I_v$ and predictions $\overline{w}_v$ for all $v \in V$
1  while there is a known mandatory vertex $v$ by Corollary 2.3.6 do query $v$;
2  repeat
3      $Q \leftarrow \emptyset$;
4      $P \leftarrow$ set of prediction mandatory vertices for current instance;
5      while $P \neq \emptyset$ and $|Q| < \gamma - 2$ do
6          pick and query some $u \in P$;    $Q \leftarrow Q \cup \{u\}$;
7          while there is a known mandatory vertex $v$ by Corollary 2.3.6 do query $v$;
8          $P \leftarrow$ set of prediction mandatory vertices for the current instance;
9      if $\exists$ distinct $u, v, w$ s.t. $\overline{w}_u$ enforces $v$ and $\{u, w\}$ is a witness set for the current
        instance by Lemma 2.3.7 then
10         query $u, w$;
11         if $w_u \in I_v$ then query $v$ ;
12     else if $\exists v, u$ such that $\overline{w}_u$ enforces $v$ then query $v$ ;
13     while there is a known mandatory vertex $v$ by Corollary 2.3.6 do query $v$;
14 until the current instance has no prediction mandatory vertices ;
15 Compute and query a minimum vertex cover $Q'$ for the current vertex cover instance;
16 while there is a known mandatory vertex $v$ by Corollary 2.3.6 do query $v$;
```

Next, we show that there is at most one iteration of the repeat-loop that executes less than $\gamma - 2$ queries in Line 6 or no queries in Lines 10–12.

Lemma 4.3.5. *Every iteration of the repeat-loop in Algorithm 9 (cf. Lines 1–14) apart from the final one executes $\gamma - 2$ queries in Line 6 and at least one query in Lines 10–12. Furthermore, if an iteration executes a query in Lines 10–12, then it executes $\gamma - 2$ queries in Line 6.*

Proof. Consider an iteration of the repeat-loop that executes less than $\gamma - 2$ queries in Line 6. Then, the while-loop from Line 5 to Line 8 terminates because the current instance has no prediction mandatory vertices. This means that the algorithm also does not execute any queries in Lines 10–12 as there cannot be a predicted weight $\overline{w}_u$ that enforces another vertex v because v would be prediction mandatory for the current instance by Corollary 4.3.4. Since the algorithm does not execute queries in Lines 10–12, the current instance still does not contain prediction mandatory vertices at the end of the current iteration of the repeat-loop, which implies that the loop terminates.

To conclude the proof, consider an iteration of the repeat-loop that executes $\gamma - 2$ queries in Line 6 but no queries in Lines 10–12. No queries in Lines 10–12 imply that there is no $\overline{w}_u$ that enforces a vertex v. Since the current instance is preprocessed, this means that for each hyperedge S we have that (i) $\overline{w}_u \notin I_v$ for the unique leftmost vertex v in S and all not yet queried vertices $u \in S \setminus \{v\}$ and (ii) $\overline{w}_v \notin I_u$ for the unique leftmost vertex v in S and all not yet queried vertices $u \in S \setminus \{v\}$. If the predictions of the not yet queried vertices are correct, then we can find the orientation of each hyperedge S by either querying the unique leftmost vertex v in S or all not yet queried vertices in $S \setminus \{u\}$. This implies that no vertex is prediction mandatory for the current instance and, therefore, the loop terminates. □

We prove the following lemma that helps us to prove that queries in Line 12 will never violate our target consistency or robustness.

Lemma 4.3.6. *Let u, v be a pair that satisfies the condition in Line 12 of Algorithm 9 leading to a query of v. After querying v, vertex u will either become mandatory by Corollary 2.3.6* *and be queried in the next execution of Line 13 or for each hyperedge S containing u we either know the orientation of S or know that u cannot be of minimum weight in S.*

Proof. Consider the instance before v is queried. Due to the failed test in Line 9, for every hyperedge S containing v, the following facts hold:

1. If u is leftmost in S, then the orientation of S is already known, or $v \in S$ and v is the only vertex in $S \setminus \{u\}$ with an interval that intersects I_u.
2. If u is not leftmost in S but intersects the interval of the leftmost vertex v' in S, then $v = v'$.
3. If u is not leftmost in S and does not intersect the interval of the leftmost vertex in S, then u is certainly not of minimum weight in S.

If condition (1) holds and the orientation of S is not known then, after querying v, either $w_v \notin I_u$ and the orientation of S is determined, or $w_v \in I_u$ and u becomes mandatory by Corollary 2.3.6.

If condition (2) holds, then either $w_v \notin I_u$ and u is certainly not of minimum weight in S, or $w_v \in I_u$ and u becomes mandatory due to Corollary 2.3.6. The result follows trivially if condition (3) holds. □

Intuitively, the lemma means that if vertex u does not become mandatory by Corollary 2.3.6 after querying v, then the algorithm will *never* even consider vertex u anymore as all hyperedges containing u are either resolved or have an orientation that is completely independent of vertex u. If that is the case, then the algorithm queries *exactly* one vertex of the witness set $\{u, v\}$. This can never lead to a violation of the target consistency and robustness as even the optimal solution has to query at least one member of $\{u, v\}$. If on the other hand u becomes mandatory, then either the predicted weight $\overline{w}_u$ is correct and v is also mandatory by Lemma 4.3.3 or the predicted weight of $\overline{w}_u$ is wrong. In the former case even OPT queries u and v, so queries to those vertices certainly do not lead to a violation of the target consistency. In the latter case, $\{u, v\}$ is still a witness set and we will show that we can charge one of the queries against a prediction error caused by vertex u, so queries to $\{u, v\}$ do not violate robustness or error-dependency.

Using these insights, we are finally ready to prove Theorem 4.3.1.

Proof of Theorem 4.3.1. Before we prove the performance bounds, we remark that the algorithm clearly solves the given instance by definition of the final two lines of the algorithm and Lemma 3.2.8. Next, we separately show that the algorithm executes at most $\gamma \cdot |\mathrm{OPT}|$ queries and at most $(1 + \frac{1}{\gamma})(|\mathrm{OPT}| + k_h)$ queries. Let ALG denote the set of queries executed by the algorithm.

Proof of $|\mathrm{ALG}| \leq \gamma \cdot |\mathrm{OPT}|$ (robustness). We start by proving the robustness bound. Vertices queried in Line 11 are mandatory due to Lemma 4.3.3 and, thus, in any feasible solution. Clearly, querying those vertices will never worsen the competitive ratio of ALG. To analyze all further queries executed by ALG, fix an optimal solution OPT.

Consider an iteration of the repeat-loop in which some query is performed in Lines 10–12. Let P' be the set of vertices queried in Lines 6, 10 and 12. If the iteration queries a vertex v in Line 12 that is enforced by the predicted weight $\overline{w}_u$ of a vertex u, then we include u in P' independent of whether the algorithm queries u at some point or not. Note that, by Lemma 4.3.6, such a vertex u is considered in exactly one iteration as it either is queried directly afterwards in Line 13 or will never be considered again be the algorithm (as we argued above). Using this and the fact that we never query vertices multiple times, we can conclude that the sets P' of different iterations are pairwise disjoint. We continue by showing that all such sets P' are also witness sets of size at most γ and, thus, querying them never violates the γ-robustness.

By Lemma 4.3.3, P' contains exactly $\gamma - 2$ vertices queried in Line 6. Furthermore, P' contains either two vertices u and w queried in Line 10 or two vertices u and v as considered in Line 12. Either way, we have $|P'| = \gamma$.

Next, we argue that P' is a witness set. If Line 10 is executed, then note that $\{u, w\} \subseteq P'$ is a witness set. If a query is performed in Line 12, then note that $\{v, u\} \subseteq P'$ is a witness set. In both cases, P' contains a witness set and, therefore, is a witness set itself. We conclude that P' is a witness set of size γ and, thus, querying P' never worsens the competitive ratio below γ.

Let V' be the set of unqueried vertices in Line 4 during the iteration of the repeat-loop consisting of Lines 1–14 in which no query is performed in Lines 10–12. Recall that Lemma 4.3.5 states that there is at most one such iteration and it has to be the last iteration of the loop. If no such iteration exists, then let V' denote the set of unqueried vertices before Line 15.

If the orientation is not yet known at this point, then the instance is not yet solved and we have $|\mathrm{OPT} \cap V'| \geq 1$. Furthermore, $|Q| \leq \gamma - 2$ holds for the set of queries executed in the iteration of the repeat-loop in which no query is performed in Lines 10–12. This implies $|Q| \leq (\gamma - 2) \cdot |\mathrm{OPT} \cap V'|$.

Let Q' denote the set of all vertices queried in Lines 15 and 16. Since the queries of Line 15 are a minimum vertex cover for the current instance and this instance is preprocessed, they are a lower bound on $|(\mathrm{OPT} \cap V') \setminus Q|$ by Lemma 2.3.7. Additionally, all queries of Line 16 are mandatory and thus their number is at most $|(\mathrm{OPT} \cap V') \setminus Q|$. This implies that $|Q'| \leq 2 \cdot |(\mathrm{OPT} \cap V') \setminus Q|$. Combining the bounds for $|Q|$ and $|Q'|$, we get $|Q| + |Q'| \leq \gamma \cdot |\mathrm{OPT} \cap V'|$.

All remaining vertices queried by ALG have been queried in Lines 1, 13 and 7. Thus, they are mandatory, part of any feasible solution, and never violate the γ-robustness. This concludes the proof of the robustness bound.

Proof of $|\mathrm{ALG}| \leq (1+\frac{1}{\gamma})(|\mathrm{OPT}|+k_h)$ (consistency and error-dependency). We continue by proving consistency and error-dependency. Fix an optimal solution OPT. Let $k^-(u)$ be the number of vertices v such that the value of v passes over an endpoint of u, i.e., $w_v \leq L_u < \overline{w}_v$, $\overline{w}_v \leq L_u < w_v$, $\overline{w}_v < U_u \leq w_v$ or $w_v < U_u \leq \overline{w}_v$. From the arguments in the proof of Theorem 4.2.5, it can be seen that, for each vertex u that is prediction mandatory at some point during the execution of the algorithm (not necessarily for the initially given instance) and is *not* in OPT, we have that $k^-(u) \geq 1$. The same holds for not prediction mandatory vertices that turn out to be mandatory.

For a subset $U \subseteq V$, let $k^-(U) = \sum_{u \in U} k^-(u)$. Note that $k_h = k^-(V)$ holds by reordering summations.

In the following, we will show for various disjoint subsets $S \subseteq V$ that $|S \cap \mathrm{ALG}| \leq (1 + \frac{1}{\gamma}) \cdot (|\mathrm{OPT} \cap S| + k^-(S))$. The union of the subsets S will contain ALG, so it is clear that the bound of $(1 + \frac{1}{\gamma}) \cdot (1 + \frac{k_h}{|\mathrm{OPT}|})$ on the competitive ratio of the algorithm follows.

Vertices queried in Line 7 are in any feasible solution, so the set P_0 of these vertices satisfies $|P_0| \leq |\mathrm{OPT} \cap P_0|$.

If there is an execution of the loop consisting of Lines 1–14 that does not perform queries in Lines 10–12, then let P_1 be the set of vertices queried in Line 6. Every vertex $u \in P_1$ is prediction mandatory for the current instance, so if $u \notin \mathrm{OPT}$ then $k^-(u) \geq 1$. Thus, we have that $|P_1| \leq |P_1 \cap \mathrm{OPT}| + k^-(P_1)$.

Let V' be the set of unqueried vertices before the execution of Line 15, and let Q' denote the vertex cover of Line 15. Since Q' is a minimum vertex cover of the current vertex cover instance, we have that $|Q'| \leq |\mathrm{OPT} \cap V'|$. Let M be the set of vertices queried in Line 16. Each vertex $v \in M$ is known mandatory because it contains the precise weight w_u of a vertex $u \in Q$. But since v was not prediction mandatory before querying Q', we have $\overline{w}_u \notin I_v$ and, therefore, $k^-(v) \geq 1$. This implies $|V' \cap \mathrm{ALG}| = |Q' \cup M| \leq |V' \cap \mathrm{OPT}| + k^-(M) \leq |V' \cap \mathrm{OPT}| + k^-(V')$.

Finally, consider an execution of the repeat-loop in which some query is performed in Lines 10–12. Let Q be the set of vertices queried in Line 6, and let W be the set of vertices queried in Lines 10–12. If a query is performed in Line 12 and u is queried in Line 13 directly afterwards, then include u in W as well. Note that $|Q| = \gamma - 2$ holds by Lemma 4.3.5. If $\overline{w}_u$ enforces v in Line 9 or 12, then v is prediction mandatory due to Corollary 4.3.4. Also, note that $k^-(Q) \geq |Q \setminus \mathrm{OPT}|$, since every vertex in Q is prediction mandatory at some point. If some $v \in Q$ is not in OPT, then $k^-(v) \geq 1$ as argued above.

We divide the proof in three cases. For a pair $\{v, u\}$ as in Line 12, note that, due to Lemma 4.3.5 (and as argued before the proof), u is not considered more than once, and is not considered in any of the previous cases.

1. If $|W| = 1$, then some vertex v was queried in Line 12 because $\overline{w}_u$ enforces v, and u is not queried by the algorithm due to Lemma 4.3.6. Then it suffices to note that $\{v, u\}$ is a witness set to see that $|Q \cup W| \leq |\mathrm{OPT} \cap (Q \cup \{u, v\})| + k^-(Q)$.

2. Consider $|W| = 2$. If W is a pair of the form $\{u, w\}$ queried in Line 10, then $k^-(v) \geq 1$ because $\overline{w}_u$ enforces v but $w_u \notin I_v$ (as v was not queried in Line 11). We can conceptually move this contribution in the hop distance to u, making $k^-(v) := k^-(v) - 1$ and $k^-(u) := k^-(u) + 1$. If v is considered another time in Line 9 or in another point of the analysis because it is enforced by some predicted weight, then it has to be the predicted weight of a vertex $u' \neq u$, so we are not counting the contribution to the hop distance more than once. If W is a pair of the form $\{v, u\}$ queried in Line 12 and in Line 13 directly afterwards, then either $W \subseteq \mathrm{OPT}$ or $k^-(v) \geq 1$: It holds that u is mandatory, so if v is not in OPT then it suffices to see that $\overline{w}_u$ enforces v to conclude that $k^-(v) \geq 1$. Either way, the fact that W is a witness set is enough to see that $|Q \cup W| \leq |\mathrm{OPT} \cap (Q \cup W)| + k^-(Q) + k^-(W)$.

3. If $|W| = 3$, then $W = \{u, v, w\}$ as in Line 9, and $|Q \cup W| = \gamma + 1$. As v is queried in Line 11, it is mandatory by Lemma 4.3.3. Since $\{u, w\}$ is a witness set, it holds that v and at least one of $\{u, w\}$ are contained in any feasible solution. This implies that at least $\frac{\gamma}{\gamma+1} \cdot |Q \cup W| - k^-(Q)$ of the vertices in $Q \cup W$ are also in OPT, so $|Q \cup W| \leq (1 + \frac{1}{\gamma})(|\mathrm{OPT} \cap (Q \cup W)| + k^-(Q))$.

The remaining intervals queried in Lines 1 and 13 are in any feasible solution. $\square$

4.3.2 Learning-augmented Algorithm w.r.t. the Mandatory Query Distance

We continue by designing a learning-augmented algorithm with a linear error dependency on the mandatory query distance k_M. Before we do so, we provide the following lower bound stating that we can only hope for a slightly worse consistency and robustness tradeoff if we want a linear error dependency on k_M.

Theorem 4.3.7. *Let $\gamma \in \mathbb{R}_{\geq 2}$ be fixed. If a deterministic algorithm for hypergraph orientation under explorable uncertainty with predictions 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.*

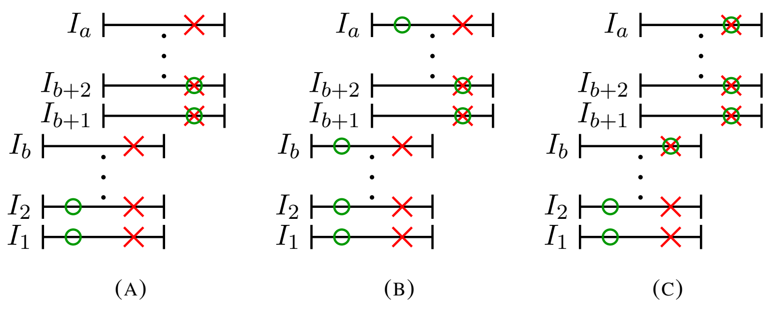


FIGURE 4.4: Intervals, precise weights and predicted weights as used in the lower bound instance in the proof of Theorem 4.3.7.

Proof. We first establish the following auxiliary claim, which is slightly weaker than the statement of the theorem:

Claim 4.3.8. *Let $\gamma' \geq 2$ be a fixed rational number. Every deterministic algorithm for the hypergraph orientation problem has competitive ratio at least $1 + \frac{1}{\gamma'-1}$ for $k_M = 0$ or has competitive ratio at least γ' for arbitrary k_M.*

Let $\gamma' = \frac{a}{b}$, with integers $a \geq 2b > 0$. Consider an instance with vertices $\{1, \dots, a\}$, hyperedges $S_i = \{i, b+1, b+2, \dots, a\}$ for $i = 1, \dots, b$, and intervals and predicted weights as depicted in Figure 4.4a. Suppose without loss of generality that the algorithm queries vertices $\{1, \dots, b\}$ in the order $1, 2, \dots, b$, and the vertices $\{b+1, \dots, a\}$ in the order $b+1, b+2, \dots, a$. Let the predictions be correct for $b+1, \dots, a-1$, and $w_1, \dots, w_{b-1} \notin I_1 \cup \dots \cup I_a$.

If the algorithm queries vertex a before vertex b, then the adversary sets $w_a \in I_b$ and $w_b \notin I_a$. (See Figure 4.4b.) This forces a query in all vertices $\{1, \dots, b\}$ as they become mandatory by Corollary 2.3.6, so the algorithm queries all a vertices, while the optimal solution queries only the b vertices $\{b+1, \dots, a\}$. Thus, the competitive ratio is at least $\frac{a}{b} = \gamma'$ if k_M can be arbitrarily large.

If the algorithm queries b before a, then the adversary sets $w_a = \overline{w}_a$ and $w_b \in I_a$; see Figure 4.4c. This forces the algorithm to query all remaining vertices in $\{b+1, \dots, a\}$ as they become mandatory by Corollary 2.3.6, i.e., a queries in total, while the optimum queries only the $a - b$ vertices in $\{b+1, \dots, a\}$. Note, however, that $k_M = 0$, since the right-side vertices $\{b+1, \dots, a\}$ are mandatory for both predicted and precise weights by Lemmas 4.2.3 and 2.3.5, while $1, \dots, b$ are neither mandatory nor prediction mandatory. Thus, the competitive ratio is at least $\frac{a}{a-b} = 1 + \frac{1}{\gamma'-1}$ for $k_M = 0$. This concludes the proof of Claim 4.3.8.

Now we are ready to prove the theorem. Let $\gamma \geq 2$ be a fixed rational. Assume that there is a deterministic algorithm that is γ-robust and has competitive ratio strictly smaller than $1 + \frac{1}{\gamma-1}$, say $1 + \frac{1}{\gamma+\varepsilon-1}$ with $\varepsilon > 0$, for $k_M = 0$. Let γ' be a rational number with $\gamma < \gamma' < \gamma + \varepsilon$. Then the algorithm has competitive ratio strictly smaller than γ' for arbitrary k_M and competitive ratio strictly smaller than $1 + \frac{1}{\gamma'-1}$ for $k_M = 0$, a contradiction to Claim 4.3.8. This shows the first statement of the theorem.

Let $\gamma \geq 2$ again be a fixed rational. Assume that there is a deterministic algorithm that has competitive ratio $1 + \frac{1}{\gamma-1}$ for $k_M = 0$ and is $(\gamma - \varepsilon)$-robust, where $\varepsilon > 0$. As there is a lower bound of 2 on the robustness of any deterministic algorithm, no such algorithm can exist for $\gamma = 2$. So we only need to consider the case $\gamma > 2$ and $\gamma - \varepsilon \geq 2$. Let γ' be a rational number with $\gamma - \varepsilon < \gamma' < \gamma$. Then the algorithm has competitive ratio strictly smaller than $1 + \frac{1}{\gamma'-1}$ for $k_M = 0$ and competitive ratio strictly smaller than γ' for arbitrary k_M, a contradiction to Claim 4.3.8. This shows the second statement of the theorem. □

This lower bound shows that the k_h-dependent guarantee of Algorithm 9 (cf. Theorem 4.3.1) cannot translate to the error measure k_M. Instead, for any $\gamma \in \mathbb{N}_{\geq 2}$, we aim for a $(1 + \frac{1}{\gamma-1})$-consistent and γ-robust algorithm with linear error dependency on k_M.

To that end, we prove the following tight bound by presenting Algorithm 11 with dependency on k_M. We remark that this algorithm only uses the initial set of prediction mandatory vertices, and otherwise ignores the predicted weights. Since access to this set is sufficient to execute the algorithm, it requires strictly less predicted information than the Algorithm 11, which relies on having access to the actual predicted weights.

Theorem 4.3.9. *There is an algorithm for hypergraph orientation under explorable uncertainty with predictions that, given an integer parameter $\gamma \geq 2$, has a competitive ratio of $\min\{(1 + \frac{1}{\gamma-1}) \cdot (1 + \frac{k_M}{|\mathrm{OPT}|}), \gamma\}$.*

Before we give a formal proof of the theorem, we start by sketching the main ideas. The algorithm emulates the two-stage structure of the offline algorithm (cf. Algorithm 8). 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 11 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 2.3.5). In contrast to the k_h-dependent algorithm, we fix the set P and do *not* recompute it when the instance changes. Then the algorithm tries to find a vertex $p \in P$ that is part of a witness set $\{p, b\}$ for the current instance. 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 $\mathcal{I}_P \setminus \mathcal{I}_R$ and, thus, contributes to the mandatory query distance k_M. 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 of the current vertex cover instance and vertices that become known mandatory by Corollary 2.3.6. For the second phase itself, we can use that a minimum vertex cover of the vertex cover instance (cf. Definition 3.2.6) is a lower bound on the optimal solution for the remaining instance by Lemma 2.3.7. 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.

Formal proof of Theorem 4.3.9. We proceed by turning these arguments into a formal analysis of Algorithm 11 to prove Theorem 4.3.9. To that end, we first show the following auxiliary lemma. Recall that $\mathcal{I}_P$ denotes the set of (initially) prediction mandatory vertices for the instance and $\mathcal{I}_R$ denotes the set of vertices that are mandatory for the precise weights.

Lemma 4.3.10. *Every vertex queried in Line 9 of Algorithm 11 is in $\mathcal{I}_R \setminus \mathcal{I}_P$, i.e., mandatory but not (initially) prediction mandatory.*

Proof. Clearly every such vertex is in $\mathcal{I}_R$ because it is known to be mandatory by Corollary 2.3.6, so it remains to prove that it is not in $\mathcal{I}_P$.

If a vertex $v \in \mathcal{I}_P$ is queried in Line 9, then it cannot be queried within the while-loop that starts at Line 2. Since v is not queried in that loop, the condition for identifying a witness set

Algorithm 11: 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 2.3.5);
2  while $\exists p \in P$ and an unqueried vertex $b$ where $\{p, b\}$ is a witness set for the current
     instance by Lemma 2.3.7 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 $v$ by Corollary 2.3.6 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  while there is a known mandatory vertex $v$ by Corollary 2.3.6 do query $v$;
```

in Line 2 implies that, after executing the loop, v is not leftmost in a not yet solved hyperedge S with $v \in S$ and that the interval of v does not intersect the interval of the leftmost vertex in a not yet solved hyperedge S with $v \in S$.

As this holds for every hyperedge that contains v, the vertex cannot become known mandatory in Line 9 and, therefore, is not queried. This implies that a vertex queried in Line 9 cannot be contained in $\mathcal{I}_P$ and, thus, the lemma. $\square$

Using the auxiliary lemma, we now proceed to prove Theorem 4.3.9.

Proof of Theorem 4.3.9. Before we prove the performance bounds, we remark that the algorithm clearly solves the given instance by definition of the final two lines of the algorithm and Lemma 3.2.8. Next, we separately show that the algorithm executes at most $\gamma \cdot |\mathrm{OPT}|$ queries and at most $(1 + \frac{1}{\gamma - 1}) \cdot (|\mathrm{OPT}| + k_M)$ queries.

Proof of $|\mathrm{ALG}| \leq \gamma \cdot |\mathrm{OPT}|$ (robustness). Given $P' \cup \{b\}$ queried in Line 5, at least one vertex is in any feasible solution since $\{b, p\}$ is a witness set and, thus, $P' \cup \{b\}$ is a witness set of size γ. Therefore, querying $P' \cup \{b\}$ never worsens the robustness below γ.

Line 7 is executed at most once because the size of P never increases. Fix an optimum solution OPT, and let V' be the set of unqueried vertices before Line 7 is executed (or before Line 8 if Line 7 is never executed). Let P be the set of vertices queried in Line 7, and let Q be the queries in Line 8.

If the orientation is already known before querying P and Q, then it must hold $P = Q = \emptyset$ and the lemma clearly holds. If the orientation is not yet known at this point, then $|\mathrm{OPT} \cap V'| \geq 1$, so $|P| \leq \gamma - 2$ implies $|P| \leq (\gamma - 2) \cdot |\mathrm{OPT} \cap V'|$. Also, since Q is a minimum vertex cover of the vertex cover instance, we get $|Q| \leq |\mathrm{OPT} \cap V'|$ by Lemma 2.3.7. Let M be the set of vertices in V' that are queried in Line 9; clearly $M \subseteq \mathrm{OPT} \cap V'$ as all those vertices are mandatory. Thus $|P| + |Q| + |M| \leq \gamma \cdot |\mathrm{OPT} \cap V'|$.

The vertices queried in Line 6 are in any feasible solution, which implies the robustness bound.

Proof of $\mathrm{ALG} \leq (1 + \frac{1}{\gamma - 1}) \cdot (|\mathrm{OPT}| + k_M)$ (consistency and error-dependency). Fix an optimal solution OPT. In the following, we will show for various disjoint subsets $J \subseteq V$ that $|J \cap \mathrm{ALG}| \leq (1 + \frac{1}{\gamma - 1}) \cdot (|\mathrm{OPT} \cap J| + k_J)$, where $k_J \leq |J \cap (\mathcal{I}_P \Delta \mathcal{I}_R)|$. The union of the subsets J will contain ALG, so it is clear that the bound of $(1+\frac{1}{\gamma-1})\cdot(1+\frac{k_M}{|\mathrm{OPT}|})$ on the competitive ratio of the algorithm follows.

Vertices queried in Lines 6 of Algorithm 11 are part of any feasible solution, hence the set P_0 of these vertices satisfies $|P_0| \leq |\mathrm{OPT} \cap P_0|$.

Given $P' \cup \{b\}$ queried in Line 5, at least $\frac{\gamma-1}{\gamma}$ of the vertices in $P' \cup \{b\}$ are prediction mandatory for the initial instance by choice of P'. Among those, let $k' \leq k_M$ be the number of vertices in $\mathcal{I}_P \setminus \mathcal{I}_R$. Then, $|\mathrm{OPT} \cap (P' \cup \{b\})| \geq \frac{\gamma-1}{\gamma} \cdot |P' \cup \{b\}| - k'$, which gives the desired bound, i.e., $|P' \cup \{b\}| \leq (1+\frac{1}{\gamma-1}) \cdot (|\mathrm{OPT} \cap (P' \cup \{b\})| + k')$.

Every vertex queried in Line 7 that is not in OPT is in $\mathcal{I}_P \setminus \mathcal{I}_R$. Hence, if there are k'' such vertices, then the set P of vertices queried in Line 7 satisfies $|P| \leq |\mathrm{OPT} \cap P| + k'' < (1+\frac{1}{\gamma-1}) \cdot (|\mathrm{OPT} \cap P| + k'')$.

Let V' be the set of unqueried vertices before Line 8 of Algorithm 11 is executed, and let Q be the queries in Line 8. Then $|Q| \leq |\mathrm{OPT} \cap V'|$ because Q is a minimum vertex cover of the vertex cover instance, which is a lower bound on OPT by Lemma 2.3.7. Let M be the set of vertices that are queried in Line 9. It holds that $|\mathrm{ALG} \cap V'| = |Q \cup M| \leq |\mathrm{OPT} \cap V'| + |M|$, so the claimed bound follows from Lemma 4.3.10. □

4.3.3 Non-Integral Parameter Gamma via Randomization

The parameter γ in Theorems 4.3.1 and 4.3.9 is restricted to integral values since the corresponding algorithms use it to determine sizes of query sets. Nevertheless, a generalization to arbitrary $\gamma \in \mathbb{R}_+$ is possible at a small loss in the guarantee. We give the following upper bound on the achievable tradeoff between consistency and robustness with linear error-dependency on k_M.

Theorem 4.3.11. *For any real number $\gamma \geq 2$, there is a randomized algorithm for the hypergraph orientation problem under explorable uncertainty with predictions that achieves a competitive ratio of* $\min\{(1+\frac{1}{\gamma-1}+\xi)\cdot(1+\frac{k_M}{|\mathrm{OPT}|}),\gamma\}$, *for* $\xi \leq \frac{\gamma-\lfloor\gamma\rfloor}{(\gamma-1)^2}$.

Proof. For $\gamma \in \mathbb{Z}$, we run Algorithm 11 and achieve the performance guarantee from Theorem 4.3.9. Assume $\gamma \notin \mathbb{Z}$, and let $\{\gamma\} := \gamma - \lfloor\gamma\rfloor = \gamma - \lceil\gamma\rceil + 1$ denote its fractional part. We run the following randomized variant of Algorithm 11. We randomly chose γ' as $\lceil\gamma\rceil$ with probability $\{\gamma\}$ and as $\lfloor\gamma\rfloor$ with probability $1-\{\gamma\}$, and then we run the algorithm with γ' instead of γ. We show that the guarantee from Theorem 4.3.9 holds in expectation with an additive term less than $\{\gamma\}$, more precisely, we show the competitive ratio

$$\min\left\{\left(1+\frac{1}{\gamma-1}+\xi\right)\cdot\left(1+\frac{k_M}{|\mathrm{OPT}|}\right),\gamma\right\},\ \text{for}\ \xi = \frac{\{\gamma\}(1-\{\gamma\})}{(\gamma-1)\lfloor\gamma\rfloor(\lfloor\gamma\rfloor-1)} \leq \frac{\{\gamma\}}{(\gamma-1)^2}.$$

Following the arguments in the proof of Theorem 4.3.9 on the robustness, the ratio of the algorithm's number of queries $|\mathrm{ALG}|$ and $|\mathrm{OPT}|$ is bounded by γ'. In expectation the robustness is

$$\begin{aligned}\mathbb{E}\left[\gamma'\right] &= (1-\{\gamma\})\cdot\lfloor\gamma\rfloor + \{\gamma\}\cdot\lceil\gamma\rceil \\ &= (1-\{\gamma\})\cdot(\gamma-\{\gamma\}) + \{\gamma\}\cdot(\gamma-\{\gamma\}+1) \\ &= \gamma.\end{aligned}$$

The error-dependent bound on the competitive ratio is in expectation (with $|\mathrm{OPT}|$ and k_M not being random variables)

$$\mathbb{E}\left[\left(1+\frac{1}{\gamma'-1}\right)\cdot\left(1+\frac{k_M}{|\mathrm{OPT}|}\right)\right] = \left(1+\mathbb{E}\left[\frac{1}{\gamma'-1}\right]\right)\cdot\left(1+\frac{k_M}{|\mathrm{OPT}|}\right).$$

Applying simple algebraic transformations, we obtain

$$\begin{aligned}\mathbb{E}\left[\frac{1}{\gamma'-1}\right] &= \frac{1-\{\gamma\}}{\lfloor\gamma\rfloor-1}+\frac{\{\gamma\}}{\lceil\gamma\rceil-1} = \frac{1-\{\gamma\}}{\gamma-\{\gamma\}-1}+\frac{\{\gamma\}}{\gamma-\{\gamma\}}\\ &= \frac{(1-\{\gamma\})(\gamma-\{\gamma\})+\{\gamma\}(\gamma-\{\gamma\}-1)}{(\gamma-\{\gamma\}-1)(\gamma-\{\gamma\})}\\ &= \frac{\gamma-2\{\gamma\}}{(\gamma-\{\gamma\}-1)(\gamma-\{\gamma\})} = \frac{1}{\gamma-1}-\frac{1}{\gamma-1}+\frac{\gamma-2\{\gamma\}}{(\gamma-\{\gamma\}-1)(\gamma-\{\gamma\})}\\ &= \frac{1}{\gamma-1}+\frac{\{\gamma\}(1-\{\gamma\})}{(\gamma-1)(\gamma-\{\gamma\}-1)(\gamma-\{\gamma\})} = \frac{1}{\gamma-1}+\frac{\{\gamma\}(1-\{\gamma\})}{(\gamma-1)\lfloor\gamma\rfloor(\lfloor\gamma\rfloor-1)}.\end{aligned}$$

Hence, the error-dependent bound on the competitive ratio is in expectation

$$\left(1+\frac{1}{\gamma-1}+\xi\right)\cdot\left(1+\frac{k_M}{|\mathrm{OPT}|}\right) \text{ with } \xi=\frac{\{\gamma\}(1-\{\gamma\})}{(\gamma-1)\lfloor\gamma\rfloor(\lfloor\gamma\rfloor-1)}\leq\frac{\{\gamma\}}{(\gamma-1)^2}\leq 1,$$

which concludes the proof. □

We can repeat essentially the same proof to obtain the analogous result for Algorithm 9 and linear error-dependency on k_h.

Theorem 4.3.12. *For any real number $\gamma \geq 2$, there is a randomized algorithm for the hypergraph orientation problem under explorable uncertainty with predictions that achieves a competitive ratio of* $\min\{(1+\frac{1}{\gamma}+\xi)\cdot(1+\frac{k_h}{|\mathrm{OPT}|}),\gamma\}$, *for* $\xi\leq\frac{1}{48}<0.021$.

Proof. For $\gamma\in\mathbb{Z}$, we run Algorithm 9 and achieve the performance guarantee from Theorem 4.3.1. Assume $\gamma\notin\mathbb{Z}$. As before, let $\{\gamma\}:=\gamma-\lfloor\gamma\rfloor=\gamma-\lceil\gamma\rceil+1$ denote its fractional part. We run the following randomized variant of Algorithm 9. We randomly chose γ' as $\lceil\gamma\rceil$ with probability $\{\gamma\}$ and as $\lfloor\gamma\rfloor$ with probability $1-\{\gamma\}$, and then we run the algorithm with γ' instead of γ. We show that the guarantee from Theorem 4.3.1 holds in expectation with an additive term less than $\{\gamma\}$, more precisely, we show the competitive ratio

$$\min\left\{\left(1+\frac{1}{\gamma}+\xi\right)\cdot\left(1+\frac{k_h}{|\mathrm{OPT}|}\right),\gamma\right\}, \text{ for } \xi=\frac{\{\gamma\}(1-\{\gamma\})}{(\gamma-1)\lfloor\gamma\rfloor(\lfloor\gamma\rfloor-1)}\leq\frac{\{\gamma\}}{(\gamma-1)^2}.$$

Exactly as in the proof of Theorem 4.3.11, we get $\mathbb{E}[\gamma']=\gamma$. The error-dependent bound on the competitive ratio is in expectation (with OPT and k_h not being random variables)

$$\mathbb{E}\left[\left(1+\frac{1}{\gamma'}\right)\cdot\left(1+\frac{k_h}{|\mathrm{OPT}|}\right)\right]=\left(1+\mathbb{E}\left[\frac{1}{\gamma'}\right]\right)\cdot\left(1+\frac{k_M}{|\mathrm{OPT}|}\right).$$

Applying simple algebraic transformations, we obtain

$$\begin{aligned}\mathbb{E}\left[\frac{1}{\gamma'}\right] &= \{\gamma\}\frac{1}{\lceil\gamma\rceil}+(1-\{\gamma\})\cdot\frac{1}{\lfloor\gamma\rfloor}\\ &= \{\gamma\}\frac{1}{\lceil\gamma\rceil}+(1-\{\gamma\})\cdot\frac{1}{\lfloor\gamma\rfloor}-\frac{1}{\gamma}+\frac{1}{\gamma}\\ &= \frac{1}{\gamma}+\frac{1}{\gamma\lceil\gamma\rceil\lfloor\gamma\rfloor}\cdot\Big(\{\gamma\}\gamma\lfloor\gamma\rfloor+(1-\{\gamma\})\cdot\lceil\gamma\rceil\gamma-\lceil\gamma\rceil\lfloor\gamma\rfloor\Big).\end{aligned}$$

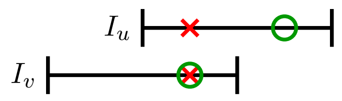


FIGURE 4.5: Uncertainty intervals as well as the predicted and precise weights as used in the proof of Theorem 4.4.1.

As we consider fractional $\gamma > 2$, it holds $\lceil\gamma\rceil = \lfloor\gamma\rfloor + 1$. Using this, we rewrite the term in brackets as

$$\begin{aligned}
&\{\gamma\}\gamma\lfloor\gamma\rfloor + (1 - \{\gamma\}) \cdot (\lfloor\gamma\rfloor + 1)\gamma - (\lfloor\gamma\rfloor + 1)\lfloor\gamma\rfloor \\
&= \{\gamma\}\gamma\lfloor\gamma\rfloor + (\lfloor\gamma\rfloor + 1)\gamma - \{\gamma\} \cdot (\lfloor\gamma\rfloor + 1)\gamma - (\lfloor\gamma\rfloor + 1)\lfloor\gamma\rfloor \\
&= (\lfloor\gamma\rfloor + 1)(\gamma - \lfloor\gamma\rfloor) - \{\gamma\} \cdot \gamma \\
&= \{\gamma\}(\lfloor\gamma\rfloor + 1 - \gamma) \\
&= \{\gamma\}(1 - \{\gamma\}),
\end{aligned}$$

where the third and fourth equalities come from the fact that $\gamma - \lfloor\gamma\rfloor = \{\gamma\}$. Note that for $\{\gamma\} \geq 0$, the expression $\{\gamma\}(1 - \{\gamma\})$ is at most $1/4$, where it reaches its maximum for $\{\gamma\} = 1/2$. Further, notice that for fractional $\gamma > 2$ it holds that $\gamma\lceil\gamma\rceil\lfloor\gamma\rfloor \geq 12$. We conclude that

$$\mathbb{E}\left[\frac{1}{\gamma'}\right] \leq \frac{1}{\gamma} + \frac{\{\gamma\}(1 - \{\gamma\})}{\gamma\lceil\gamma\rceil\lfloor\gamma\rfloor} \leq \frac{1}{\gamma} + \frac{1}{48} < \frac{1}{\gamma} + 0.021$$

which proves the theorem. □

4.4 Sorting under Explorable Uncertainty

In this section, 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 intervals 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.

As the main result of this section, we give a learning-augmented algorithm for sorting under explorable with predictions that is 1-consistent and 2-robust with a linear error-dependency for any $k \in \{k_\#, k_M, k_h\}$. Clearly, no algorithm can be better than 1-consistent, and no deterministic algorithm can be better than 2-robust by Theorem 2.2.2. Before we give our algorithmic results, we show that a sublinear error dependency is not possibly by giving a simple lower bound example.

Theorem 4.4.1. *Any deterministic algorithm for sorting or hypergraph orientation under explorable uncertainty with predictions (even for pairwise disjoint hyperedges) has a competitive ratio $\rho \geq \min\{1 + \frac{k}{|\mathrm{OPT}|}, 2\}$, for any error measure $k \in \{k_\#, k_M, k_h\}$.*

Proof. Consider the input instance of the hypergraph orientation problem consisting of a single edge $\{v, u\}$ with intervals and predicted weights as shown in Figure 4.5.

If the algorithm starts querying I_v, then the adversary sets $w_v = \overline{w}_v$ and the algorithm is forced to query u. Then $w_u \in I_u \setminus I_v$, so the optimum queries only u. It is easy to see that

$k_\# = k_M = k_h = 1$. A symmetric argument holds if the algorithm starts querying u. In that case, $w_u = \overline{w}_u$ which forces as query to v with $w_v \in I_v \setminus I_u$. Taking multiple copies of this instance gives the result for any $k \leq |\mathrm{OPT}|$. □

4.4.1 A Learning-augmented Algorithm for Sorting

As a main result of this section, we show the following upper bound that matches Theorem 4.4.1.

Theorem 4.4.2. *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 orienting interval graphs is the simple characterization of mandatory vertices: any vertex with an interval that contains the precise weight of another vertex is mandatory [HL21]. This observation is a direct consequence of Lemma 2.3.5 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 2.3.7 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 12) 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 after the first stage queries. Algorithm 12 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 vertices (cf. Line 4) and all vertices 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 Section 4.4.2.

Lemma 4.4.3. *For an instance of the sorting problem, let* $\mathcal{I}_P$ *be the set of prediction mandatory vertices and* M *be the set of known mandatory vertices after querying* $\mathcal{I}_P$ *(by exhaustively applying Corollary 2.3.6). 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 [HL21]. 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.

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

Algorithm 12: Learning-augmented algorithm for sorting under explorable uncertainty with 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 there is a known mandatory vertex $v$ by Corollary 2.3.6 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  while there is a known mandatory vertex $v$ by Corollary 2.3.6 do query $v$;
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 4.4.3);
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 4.4.3) then query $\{x_1, x_3, \ldots, x_{p-1}\}$;
13         else query $\{x_2, x_4, \ldots, x_p\}$;
14     while there is a known mandatory vertex $v$ by Corollary 2.3.6 do query $v$;
```

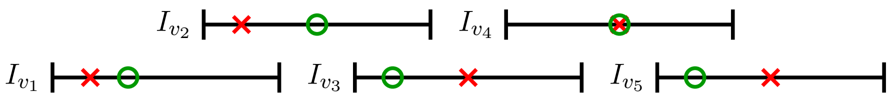


FIGURE 4.6: Example showing that it is necessary to query a specific vertex cover in the second phase to ensure 2-robustness. Circles illustrate precise weights and crosses illustrate the predicted weights.

witness sets of the critical isolated vertices into account when deciding which vertex cover to query. To see this, consider the example of Figure 4.6: 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) vertices left. As the only vertex queried in the first phase, $\{v_1\}$ must be part of every clique partition. Since v_1 is not mandatory, it would qualify as a critical isolated vertex, and v_2 is the only possible distinct partner of v_1 with an intersecting interval that Lemma 4.4.3 could assign to v_1. After querying v_1, the remaining vertex cover instance is the path v_2, v_3, v_4, v_5. One possible minimum vertex cover of this instance is $\{v_3, v_5\}$, but querying this vertex cover renders v_2 and v_4 mandatory by Corollary 2.3.6. 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 illustrates that the selection of the minimum vertex cover in the second phase is important to ensure 2-robustness.

The next lemma shows that instances occurring in the second phase indeed have a structure similar to the example by exploiting that such instances have no prediction mandatory vertices.

Lemma 4.4.4. *Each connected component of an interval graph without prediction mandatory vertices is either a path or a single vertex.*

Proof. First, observe that there are no intervals I_v, I_u with $I_u \subseteq I_v$ as this would imply $\overline{w}_u \in I_v$, which contradicts the assumption as v would be prediction mandatory. Thus, the graph is a proper interval graph. We claim that the graph contains no triangles; for proper interval graphs, this implies that each connected component is a path, because the 4-star $K_{1,3}$ is a forbidden induced subgraph [Weg67]. Suppose there is a triangle abc, and assume that $L_a \leq L_b \leq L_c$; it holds that $U_a \leq U_b \leq U_c$ because no interval is contained in another. Since I_a and I_c intersect, we have that $U_a \geq L_c$, so $I_b \subseteq I_a \cup I_c$ and it must hold that $\overline{w}_b \in I_a$ or $\overline{w}_b \in I_c$, a contradiction to the instance being prediction mandatory free. □

Further, we observe that if the intervals of critical isolated vertices intersect intervals of vertices on such a path component, they must also 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 4.4.3 is an endpoint of such a path component, as we show in Section 4.4.2.

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 vertices (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. This case would for example ensure that we pick the "right" vertex cover for the example instance of Figure 4.6. Lemma 3.2.8 guarantees that the algorithm indeed queries a feasible query set. The following lemma shows that this strategy indeed ensures 2-robustness by using Lemma 4.4.3.

Lemma 4.4.5. *Algorithm 12 is* 2*-robust for sorting under explorable uncertainty with predictions.*

Proof. Fix an optimal 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 2.3.5. 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 $2 \cdot |W \cap \mathrm{OPT}|$ vertices 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 4.4.3, 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 we noted above and will show in Section 4.4.2, 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 $\mathcal{W}$. Then, $\mathcal{W}$ contains all critical isolated vertices of $\mathcal{C}$ and all vertices 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 4.4.3

and this partner is an endpoint of a path component, W contains at most one critical isolated vertex per distinct endpoint of P and, thus, 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 the interval 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 because $w_{x_1} \in I_{x_2}$, 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 because $w_{x_1} \notin I_{x_2}$, 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 4.4.6. *Algorithm 12 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 4.2.5 then 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 2.3.5. 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. □

4.4.2 Computing the Clique Partition

We continue by proving Lemma 4.4.3 and restate it here for the sake of readability:

Lemma 4.4.3. *For an instance of the sorting problem, let $\mathcal{I}_P$ be the set of prediction mandatory vertices and M be the set of known mandatory vertices after querying $\mathcal{I}_P$ (by exhaustively applying Corollary 2.3.6). 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 describe how to compute a clique partition $\mathcal{C}$ of $\mathcal{I}_P \cup M$ that satisfies the lemma. To that end, consider the set $\mathcal{I}_P \setminus M$. The elements of M are allowed to be part of a clique of size one, so we ignore them for now. Each $v \in \mathcal{I}_P \setminus M$ is prediction mandatory because it contains the predicted weight of some other vertex u (cf. Lemma 4.2.3 and recall that the input instance is an interval graph). For each such v we pick an arbitrary $\pi(v) \in V$ with $\overline{w}_{\pi(v)} \in I_v$ as the *parent* of v. Next, we build a forest of arborescences (rooted out-trees) that contains all vertices of $\mathcal{I}_P \setminus M$ (and some additional vertices) and define it by its arcs $\mathcal{E}$. Afterwards, we use $\mathcal{E}$ to define the clique partition of the lemma. We construct $\mathcal{E}$ by just iteratively considering all elements $v \in \mathcal{I}_P \setminus M$ in an arbitrary order and add $(\pi(v), v)$ to $\mathcal{E}$ if that does not create a cycle. Each so constructed arborescence contains at most one vertex that is not in $\mathcal{I}_P \cup M$, and this vertex has to be the root of the arborescence.

Algorithm 13: Algorithm to create a clique partition from a forest of arborescences.

```
Input: Forest of arborescences $\mathcal{E}$, set of prediction mandatory vertices $\mathcal{I}_P$, set of
       known mandatory vertices $M$.
1 $C_v \leftarrow \emptyset$ for all $v \in V$; $\mathcal{S} \leftarrow \mathcal{I}_P \cup M$;
2 while $\mathcal{S} \neq \emptyset$ do
3     let $v$ be a deepest vertex in the forest of arborescences $(\mathcal{I}, \mathcal{E})$ among those in $\mathcal{S}$;
4     if $(\pi(v), v) \in \mathcal{E}$ then
5         $C_{\pi(v)} \leftarrow \{v' \in \mathcal{S} : (\pi(v), v') \in \mathcal{E}\}$;
6         if $\pi(v) \in \mathcal{S}$ then $C_{\pi(v)} \leftarrow C_{\pi(v)} \cup \{\pi(v)\}$;
7         $\mathcal{S} \leftarrow \mathcal{S} \setminus C_{\pi(v)}$;
8     else $C_v \leftarrow \{v\}$;    $\mathcal{S} \leftarrow \mathcal{S} \setminus C_v$;
9 return $\mathcal{C} = \{C_v \mid v \in V \wedge C_v \neq \emptyset\}$ ;
```

Based on $\mathcal{E}$, we create a first clique partition $\mathcal{C}$ using Algorithm 13. Since all vertices in set C_v, as created by the algorithm, contain the predicted weight $\overline{w}_v$, the created sets are clearly cliques. Each of the partial clique partitions (created for a single arborescence in the forest of arborescences) may contain at most a single clique of size one. To satisfy the lemma, each such clique $\{v\}$ must either satisfy $v \in M$ or needs a distinct $u \notin \mathcal{I}_P \cup M$ with $I_v \cap I_u \neq \emptyset$.

If the root of the arborescence is in M, then the only clique $\{v\}$ of size one created for the arborescence contains the root of the arborescence. Since the root is in M, the cliques for the arborescence satisfy the lemma. If the root of the arborescence is not in $\mathcal{I}_P \cup M$, then the vertex v in the clique of size one might not be the root of the arborescence but a direct child of the root. In that case, the parent $\pi(v)$ of v (the vertex in the clique of size 1) is not in $\mathcal{I}_P \cup M$ and we use $\pi(v)$ as the distinct partner $u \notin \mathcal{I}_P \cup M$ with $I_v \cap I_u \neq \emptyset$ of v. We remark that there must be such a $\pi(v)$ that is the endpoint of a path component in the subgraph induced by $V \setminus (\mathcal{I}_P \cup M)$ as otherwise there must be a vertex u on the path with $\overline{w}_v \in I_u$; a contradiction to the vertices on the path not being part of $\mathcal{I}_P$. We pick this endpoint of a path component as the partner of v. In case we run into this situation for multiple cliques $\{v\}$ of different arborescences but with the same $\pi(v)$, we can just merge all those cliques into a single one. This results in a clique as the intervals of all vertices in those smaller cliques contain the predicted weight $\overline{w}_{\pi(v)}$.

The problematic case is when the root of the arborescence is part of $\mathcal{I}_P \setminus M$. Note that this can only be the case if the parent $\pi(r)$ of the root r is also part of the arborescence, as otherwise the corresponding edge $(\pi(r), r)$ would have been added and r would not be the root. So the parent $\pi(r)$ of the root r is also part of the arborescence but the edge $(\pi(r), r)$ was not added because it would have created a cycle. We handle this situation by showing the following auxiliary lemma. It states that in this case we can revise the clique partition of that arborescence in such a way that all cliques in that clique partition have size at least 2. The auxiliary lemma then concludes the proof of Lemma 4.4.3.

Lemma 4.4.7. *Consider an out-tree (arborescence) T on a set of prediction mandatory vertices, where an edge (u, v) represents that $\overline{w}_u \in I_v$. Let the root be r. Let vertex m with $\overline{w}_m \in I_r$ be a descendant of the root somewhere in T. Then the vertices in T can be partitioned into cliques (sets of pairwise overlapping intervals) in such a way that all cliques have size at least 2.*

Proof. We refer to the clique partition method of Lines 2–8 in Algorithm 13 as algorithm CP. This method will partition the nodes of an arborescence into cliques, each consisting either of a subset of the children of a node, or of a subset of the children of a node plus the parent

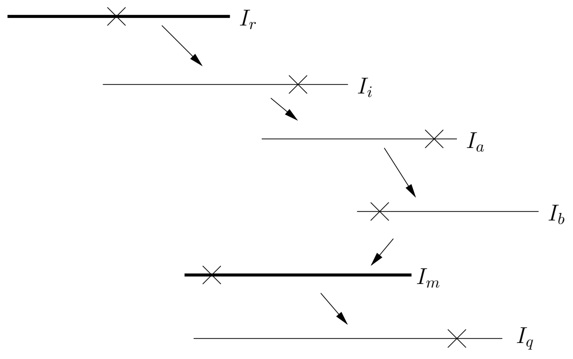


FIGURE 4.7: Illustration of path from I_r to I_m's child I_q in T

of those children. In the case considered in this lemma, all cliques will have size at least 2, except that the clique containing the root of the tree may have size 1.

We first modify T as follows: If there is a node v in T that is not a child of the root r but contains $\overline{w}_r$, then we make r the parent of v (i.e., we remove the subtree rooted at v and re-attach it below the root). After this transformation, all vertices whose intervals contain $\overline{w}_r$ are children of r.

Apply CP to each subtree of T rooted at a child of r. For each of the resulting partitions, we call the clique containing the root of the subtree the *root clique* of that subtree. There are several possible outcomes that can be handled directly:

- At least one of the clique partitions has a root clique of size 1. In that case we combine all these root cliques of size 1 with r to form a clique of size at least 2, and we are done: This new clique together with all remaining cliques from the clique partitions of the subtrees forms the desired clique partition.
- All of the clique partitions have root cliques of size at least 2, and at least one of them has a root clique of size at least 3. Let s be the root node of a subtree whose root clique has size at least 3. We remove s from its clique and form a new clique from s and r, and we are done.
- All of the clique partitions have root cliques of size exactly 2, and at least one of the children v of r has $\overline{w}_v \in I_r$. Then we add r to the root clique that contains v. We can do this because all intervals in that root clique contain $\overline{w}_v$.

Now assume that none of these cases applies, so we have the following situation: All of the clique partitions have root cliques of size exactly 2, and every child v of r has its predicted weight outside I_r, i.e., $\overline{w}_v \notin I_r$. In particular, m, the vertex that makes r prediction mandatory ($\overline{w}_m \in I_r$), cannot be a child of r.

Let T' be the subtree of T that is rooted at a child of r and that contains m. Let the root of T' be v.

Observe that I_v is the only interval in T' that contains $\overline{w}_r$, because all vertices with intervals containing $\overline{w}_r$ are children of r in T. Assume w.l.o.g. that $\overline{w}_v$ lies to the right of I_r. Then all intervals of vertices in T', except for I_v, lie to the right of $\overline{w}_r$. See Figure 4.7 for an illustration of a possible configuration of the intervals on the path from r to m (and a child q of m) in T.

Now re-attach the subtree T_m rooted at m as a child of r (ignoring the fact that $\overline{w}_r$ is not inside I_m), and let $T_v = T' \setminus T_m$ denote the result of removing T_m from T'. Re-apply CP to the two separate subtrees T_m and T_v. The possible outcomes are:

- The root clique of at least one of the two subtrees has size 1. We can form a clique by combining r with those (one or two) root cliques of size 1. As both I_v and I_m intersect

I_r from the right, the resulting set is indeed a clique. Together with all other cliques from the clique partitions of T_m and T_v, and those of the other subtrees of r in T, we obtain the desired clique partition.

- The root cliques of both subtrees have size at least 2. We add r to the root clique of T_m. That root clique contains only vertices with intervals containing $\overline{w}_m$, and I_r also contains $\overline{w}_m$, so we do indeed get a clique if we add r to that root clique. This new clique, together with all other cliques from the clique partitions of T_m and T_v, and those of the other subtrees of r in T, forms the desired clique partition.

This concludes the proof of the lemma. □

4.4.3 Guarantee depending on the Number of Wrong Predictions

We continue by proving that Algorithm 12 executes at most $|\mathrm{OPT}| + k_\#$ queries. This then concludes the proof of Theorem 4.4.2.

Recall that, in order to compute the clique partition of Lemma 4.4.7, we, for each $v \in \mathcal{I}_P \setminus M$ fix an arbitrary $\pi(v) \in V$ with $\overline{w}_{\pi(v)} \in I_v$ as the *parent* of v. These parents will be used in the following proofs.

To prove the $k_\#$-dependent guarantee, we need the following auxiliary lemma.

Lemma 4.4.8. *Fix the state of set $\mathcal{S}$ as in Line 3 of Algorithm 12. For each vertex u, let $S_u = \{v \in \mathcal{S} : \pi(v) = u\}$. For any path P considered in Line 9 and any vertex $u \in P$ that is not an endpoint of P, it holds that $S_u = \emptyset$.*

Proof. Suppose by contradiction that some $u \in P$ that is not an endpoint of P has $S_u \neq \emptyset$. Let a and b be its neighbors in P, and let $v \in S_u$. We have that $\overline{w}_u \notin I_a \cup I_b$, otherwise either a or b would be prediction mandatory and, therefore, have been queried in Line 4. Thus $(I_v \cap I_u) \setminus (I_a \cup I_b) \neq \emptyset$, because $\overline{w}_u \in I_v$ but $\overline{w}_u \notin I_a \cup I_b$. It is not the case that $I_v \subseteq I_u$ or $I_u \subseteq I_v$: If $I_v \subseteq I_u$, then u would have been queried in Line 2 before P is considered; if $I_u \subseteq I_v$, then v would have been queried in Line 2 and $v \notin \mathcal{S}$. Therefore it must be that $I_v \subseteq I_a \cup I_u \cup I_b$, otherwise $I_a \subseteq I_v$ or $I_b \subseteq I_v$ (again a contradiction for $v \in \mathcal{S}$) since $(I_v \cap I_u) \setminus (I_a \cup I_b) \neq \emptyset$ and a, u, b forms a simple path in the interval graph. However, if $I_v \subseteq I_a \cup I_u \cup I_b$, then v would have forced a query to a, b or u in Line 5 as one of the corresponding intervals must contain w_v and, thus, becomes mandatory by Corollary 2.3.6. This is a contradiction to a,b and u being part of path P. □

Theorem 4.4.9. *Algorithm 12 performs at most $|\mathrm{OPT}| + k_\#$ queries.*

Proof. Fix an optimum solution OPT. We partition the vertices in V into sets with the following properties. One of the sets $\tilde{S}$ contains vertices that are not queried by the algorithm. We have a collection $\mathcal{S}'$ of vertex sets in which each set has at most one vertex not in OPT, i.e., $|S' \setminus \mathrm{OPT}| \leq 1$ for all $S' \in \mathcal{S}'$. Also, if it has one vertex not in OPT, then we assign a distinct prediction error to that set, in such a way that each error is assigned to at most one set. The vertex corresponding to the prediction error does not need to be in the same set. Let V' be the set of vertices with a prediction error ($w_v \neq \overline{w}_v$ for all $v \in V'$) assigned to some set in $\mathcal{S}'$. Finally, we have a collection $\mathcal{W}$ of vertex sets such that for every $W \in \mathcal{W}$ it holds that $|\mathrm{ALG} \cap W| \leq |W \cap \mathrm{OPT}| + k_\#(W \setminus V')$, where $k_\#(X)$ is the number of vertices in X with incorrect predictions. If we have such a partition, then it is clear that we spend at most $\mathrm{OPT} + k_\#$ queries.

We begin by adding a set that contains all vertices queried in Lines 2 and 5 ($M_1 \cup M_2$ in the Pseudocode) to $\mathcal{S}'$; all such vertices are clearly in OPT, and we do not need to assign a prediction error.

Fix the state of $\mathcal{S}$ as in Line 3. To deal with the vertices queried in Line 4, we add to $\mathcal{S}'$ the set $S_u = \{v \in \mathcal{S} : \pi(v) = u\}$ for all $u \in V$. Note that each such set is a clique, because the corresponding intervals of all vertices in S_u contain $\overline{w}_u$ and we are considering an interval graph. Therefore, using Lemma 2.3.7, at most one vertex in S_u is not in OPT, and if that occurs, then $\overline{w}_u \neq w_u$, and we assign this prediction error to S_u.

Let $P = x_1 x_2 \cdots x_p$ with $p \geq 2$ be a path considered in Line 9, and let P' be the set of intervals in P that are queried in the following execution of Lines 10, 12 or 13. It clearly holds that $|P'| = \lfloor |P|/2 \rfloor \leq |P \cap \mathrm{OPT}|$ as P is a path and each edge defines a witness set by Lemma 2.3.7. It also holds that at most $k_\#(P')$ intervals in P are queried in Line 14: Each vertex $u \in P'$ can force a query to at most one vertex v in Line 14 as weight w_u can only be contained in the interval of at most one neighbor v of u in the path, and in that case the predicted weight of u is incorrect because $w_u \in I_v$ but $\overline{w}_u \notin I_v$, or v would have been queried in Line 4. We will create a set $W \in \mathcal{W}$ and possibly modify $\mathcal{S}'$, in such a way that $P \subseteq W$ and $P' \cap V' = \emptyset$, so it is enough to show that

$$|\mathrm{ALG} \cap W| \leq |W \cap \mathrm{OPT}| + k_\#(P'). \tag{4.4.1}$$

We initially take W as the vertices in P. By Lemma 4.4.8, it holds that $S_u = \emptyset$ for any $u \in P$ with $u \notin \{x_1, x_p\}$. If $S_{x_1} \subseteq \mathrm{OPT}$, then we did not assign a prediction error to S_{x_1}. Otherwise, let v be the only vertex in $S_{x_1} \setminus \mathrm{OPT}$. The predicted weight of I_{x_1} is incorrect because $\overline{w}_{x_1} \in I_v$, and it must hold that $x_1 \in \mathrm{OPT}$, or OPT would not be able to decide the order between x_1 and v (as $I_v \cap I_{x_1} \neq \emptyset$ which implies that $\{v, x_1\}$ is an edge and, therefore, a witness pair). If $x_1 \notin P'$, then we will not use its error in the bound of $|\mathrm{ALG} \cap W|$ to prove Equation (4.4.1). Otherwise, we add v to W and remove it from S_{x_1}, and now we do not need to assign a prediction error to S_{x_1} anymore as v was the sole member of S_{x_1} not in OPT. We do a similar procedure for x_p, and since at most one of x_1, x_p is in P' by definition of Lines 10, 12 and 13, we only have two cases to analyze: (1) $W = P$, or (2) $W = P \cup \{v\}$ with $\pi(v) \in \{x_1, x_p\}$.

1. $W = P$. Clearly $|\mathrm{ALG} \cap W| \leq |P'| + k_\#(P') \leq |W \cap \mathrm{OPT}| + k_\#(P')$ as we already argued that each member of P' (the vertices queried in Lines 10, 12 or 13) can lead to at most one additional query in Line 14 and only if it has an incorrect predicted weight.

2. $W = P \cup \{v\}$, with $\pi(v) \in \{x_1, x_p\}$. Suppose w.l.o.g. that $\pi(v) = x_1$. Remember that $x_1 \in P'$, that $x_1 \in \mathrm{OPT}$, that $v \notin \mathrm{OPT}$ and that the predicted weight of x_1 is incorrect. Since $x_1 \in P'$, it holds that $|P|$ is even and $x_2 \notin P'$. We have two cases.

   (a) x_2 is not queried in Line 14. Then x_1 does not force a query in Line 14, so

   $$\begin{aligned} |\mathrm{ALG} \cap W| &\leq |P' \cup \{v\}| + k_\#(P' \setminus \{x_1\}) \\ &= |P'| + 1 + k_\#(P' \setminus \{x_1\}) \\ &\leq |P \cap \mathrm{OPT}| + k_\#(P') \\ &\leq |W \cap \mathrm{OPT}| + k_\#(P'). \end{aligned}$$

   (b) x_2 is queried in Line 14. Then $x_1, x_2 \in \mathrm{OPT}$, and $|\mathrm{OPT} \cap (P \setminus \{x_1, x_2\})| \geq |P' \setminus \{x_1\}|$ because $|P|$ is even. Therefore,

   $$\begin{aligned} |\mathrm{ALG} \cap W| &\leq |P' \cup \{x_2, v\}| + k_\#(P' \setminus \{x_1\}) \\ &\leq |P \cap \mathrm{OPT}| + 1 + k_\#(P' \setminus \{x_1\}) \\ &\leq |W \cap \mathrm{OPT}| + k_\#(P'). \end{aligned}$$

Finally, we add the remaining intervals that are not queried by the algorithm to $\tilde{S}$. □

4.5 Learnability of Predictions

In this section, we argue about the learnability of our predictions with regard to the different error measures for a given instance of hypergraph orientation $H = (V, E)$ under explorable uncertainty with the set of uncertainty intervals $\mathcal{I} = \{I_v \mid v \in V\}$ and $n := |V|$ vertices. To this end, we prove the predictions to be *probably approximately correct (PAC) learnable* [Val84]. The formal definition for this type of learnability results is part of the theorem statements in the following two subsections.

We assume that the realization w of precise weights for $\mathcal{I}$ is i.i.d. drawn from an unknown distribution D, and that we can i.i.d. sample realizations from D to obtain a training set. Let $\mathcal{H}$ denote the set of all possible prediction vectors $\overline{w}$, with $\overline{w}_v \in I_v$ for each $I_v \in \mathcal{I}$. Let $k_h(w, \overline{w})$ denote the hop distance of the prediction $\overline{w}$ for the realization with the precise weights w. Since w is drawn from D, the value $k_h(w, \overline{w})$ is a random variable. Analogously, we consider $k_M(w, \overline{w})$ with regard to the mandatory query distance. Our goal is to learn predictions $\overline{w}$ that (approximately) minimize the expected error $\mathbb{E}_{w\sim D}[k_h(w, \overline{w})]$ respectively $\mathbb{E}_{w\sim D}[k_M(w, \overline{w})]$. In the following, we argue separately about the learnability with respect to k_h and k_M.

4.5.1 Learning with Respect to the Hop Distance

We prove the following theorem that states learnability w.r.t. k_h.

Theorem 4.5.1. *Given a fixed instance of the hypergraph orientation problem under explorable uncertainty with graph $G = (V, E)$ and intervals I_v for all $v \in V$. For any $\varepsilon, \delta \in (0, 1)$, there exists a learning algorithm that, using a training set of size m, returns predictions $\overline{w} \in \mathcal{H}$, such that $\mathbb{E}_{w\sim D}[k_h(w, \overline{w})] \leq \mathbb{E}_{w\sim D}[k_h(w, \overline{w}^*)] + \varepsilon$ holds with probability at least $(1 - \delta)$, where $\overline{w}^* = \arg\min_{\overline{w}' \in \mathcal{H}} \mathbb{E}_{w\sim D}[k_h(w, \overline{w}')]$. The sample complexity is $m \in \mathcal{O}\left(\frac{(\log(n)-\log(\delta/n))\cdot(2n)^2}{(\varepsilon/n)^2}\right)$ and the running time is polynomial in m and n.*

We refer to $k^+(v)$ (cf. Section 4.2.2) as *the hop distance of a vertex* $v \in V$. Since each I_v is an open interval, there are infinitely many predictions $\overline{w}$, and, thus, the set $\mathcal{H}$ is also infinite. In order to reduce the size of $\mathcal{H}$, we discretize each I_v by fixing a finite number of potentially predicted values $\overline{w}_v$ of I_v. We define the set $\mathcal{H}_v$ of predicted values for I_v as follows. Let $\{B_1, \ldots, B_l\}$ be the set of lower and upper limits of intervals in $\mathcal{I} \setminus I_v$ that are contained in I_v. Assume that $B_1, \ldots, B_l$ are indexed by increasing value. Let $B_0 = L_v$ and $B_{l+1} = U_v$ and, for each $j \in \{0, \ldots, l\}$, let h_j be an arbitrary value of (B_j, B_{j+1}). We define $\mathcal{H}_i = \{B_1, \ldots, B_l, h_0, \ldots, h_l\}$. Since two values $\overline{w}_v, \overline{w}_v' \in (B_j, B_{j+1})$ always lead to the same hop distance for vertex v, there will always be an element of $\mathcal{H}_v$ that minimizes the expected hop distance for v. As $k_h(w, \overline{w})$ is just the sum of the hop distances over all $v \in V$, and the hop distances of two vertices v and v' with $v \neq v'$ are independent, restricting $\mathcal{H}$ to the set $\mathcal{H}_1 \times \mathcal{H}_2 \times \ldots \times \mathcal{H}_{|E|}$ (assuming the vertices are enumerated from 1 to $|V|$) does not affect the accuracy of our predictions. Each $\mathcal{H}_v$ contains at most $\mathcal{O}(|V|)$ values, and, thus, the discretization reduces the size of $\mathcal{H}$ to at most $\mathcal{O}(|V|^{|V|})$. In particular, $\mathcal{H}$ is now finite.

To efficiently learn predictions that satisfy Theorem 4.5.1, we again exploit that the hop distances of two vertices v and v' with $v \neq v'$ are independent. This is, because the hop distance of v only depends on the predicted weight $\overline{w}_v$ and the precise weight w_v, but is independent of all $\overline{w}_{v'}$ and $w_{v'}$ with $v \neq v'$. Let $k_v^+(w_v, \overline{w}_v)$ denote the hop distance $k^+(v)$ of vertex v for the predicted weight $\overline{w}_v$ and the precise weight w_v, and, for each $v \in V$, let $\overline{w}_v^*$ denote the predicted weight that minimizes $\mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v)]$. Since the hop distances of the single vertices are independent, the vector $\overline{w}^*$ then minimizes the expected hop distance of the complete instance. Thus, if we can approximate the individual $\overline{w}_v^*$, then we can show Theorem 4.5.1.

Lemma 4.5.2. *Given a fixed instance of the hypergraph orientation problem under explorable uncertainty with graph $G = (V, E)$ and intervals I_v for all $v \in V$. For any $\varepsilon, \delta \in (0, 1)$, and any $v \in V$, there exists a learning algorithm that, using a training set of size $m \in \mathcal{O}\left(\frac{(\log(|V|)-\log(\delta))\cdot|V|^2}{\varepsilon^2}\right)$, returns a predicted weight $\overline{w}_v \in \mathcal{H}_v$ in time polynomial in $|V|$ and m, such that $\mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v)] \leq \mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v^*)] + \varepsilon$ holds with probability at least $(1-\delta)$, where $\overline{w}_v^* = \arg\min_{\overline{w}_v \in \mathcal{H}} \mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v)]$.*

Proof. We show that the basic *empirical risk minimization (ERM)* algorithm already satisfies the lemma. ERM first i.i.d. samples a training set $S = \{w^1, \dots, w^m\}$ of m precise weight vectors from D. Then, it returns the $\overline{w}_v \in \mathcal{H}_v$ that minimizes the *empirical error* $h_S(\overline{w}_v) = \frac{1}{m}\sum_{j=1}^m k_v^+(w_v^j, \overline{w}_v)$.

Recall that, as a consequence of the discretization, $\mathcal{H}_v$ contains at most $\mathcal{O}(|V|)$ values. Since $\mathcal{H}_v$ is finite, and the error function k_v^+ is bounded by the interval $[0, |V|]$, it satisfies the *uniform convergence property*; cf. [SB14]. (This follows also from the fact that $\mathcal{H}_v$ is finite and, thus, has finite VC-dimension; cf. [Vap92].) This implies that, for

$$m = \left\lceil \frac{2\log(2|\mathcal{H}_i|/\delta)|V|^2}{\varepsilon^2} \right\rceil \in \mathcal{O}\left(\frac{(\log(|V|)-\log(\delta))\cdot|V|^2}{\varepsilon^2}\right),$$

it holds $\mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v)] \leq \mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v^*)] + \varepsilon$ with probability at least $(1-\delta)$, where $\overline{w}_v$ is the predicted weight learned by ERM (cf. [SB14; Vap99]). As $|\mathcal{H}_v| \in \mathcal{O}(|V|)$, ERM also satisfies the running time requirements of the lemma. □

Proof of Theorem 4.5.1. Let $\varepsilon' = \frac{\varepsilon}{|V|}$ and $\delta' = \frac{\delta}{|V|}$. Furthermore, let $\mathcal{H}_{\max} = \arg\max_{\mathcal{H}_v} |\mathcal{H}_v|$. To learn predictions that satisfy the theorem, we first sample a training set $S = \{w^1, \dots, w^m\}$ with $m = \left\lceil \frac{2\log(2|\mathcal{H}_{\max}|/\delta')|V|^2}{\varepsilon'^2} \right\rceil$. Next, we apply Lemma 4.5.2 to each $\mathcal{H}_v$ to learn a predicted weight $\overline{w}_v$ that satisfies the guarantees of the lemma for ε', δ'. In each application of the lemma, we use the *same* training set S that was previously sampled.

For each $\overline{w}_v$ learned by applying the lemma, the probability that the guarantee of the lemma is *not* satisfied is less than δ'. By the union bound this implies that the probability that at least one $\overline{w}_v$ with $v \in V$ does not satisfy the guarantee is upper bounded by $\sum_{v\in V} \delta' \leq |V| \cdot \delta' = \delta$. Thus, with probability at least $(1-\delta)$, all $\overline{w}_v$ satisfy $\mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v)] \leq \mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v^*)] + \varepsilon'$. Since by linearity of expectations

$$\mathbb{E}_{w\sim D}[k_h(w, \overline{w}^*)] = \sum_{v\in V} \mathbb{E}_{w\sim D}[k_v^+(w_v, w_v^*)],$$

we can conclude that the following inequality, where $\overline{w}$ is the vector of the learned predicted values, holds with probability at least $(1-\delta)$, which implies the theorem:

$$\begin{aligned}
\mathbb{E}_{w\sim D}[k_h(w, \overline{w})] &= \sum_{v\in V} \mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v)] \\
&\leq \sum_{v\in V} \mathbb{E}_{v\sim D}[k_v^+(w_v, \overline{w}_v^*)] + \varepsilon' \\
&\leq \left(\sum_{v\in V} \mathbb{E}_{w\sim D}[k_v^+(w_v, \overline{w}_v^*)]\right) + |V| \cdot \varepsilon' \\
&\leq \mathbb{E}_{w\sim D}[k_h(w, \overline{w}^*)] + \varepsilon.
\end{aligned}$$

□

4.5.2 Learning with Respect to the Mandatory Query Distance

Next, we argue about the learnability w.r.t. k_M. Since each I_v is an open interval, there are infinitely many predictions $\overline{w}$, and, thus, the set $\mathcal{H}$ is also infinite. In order to reduce the size of $\mathcal{H}$, we discretize each I_v by fixing a finite number of potentially predicted weights $\overline{w}_v$ of I_v using the same technique as in the previous section for k_h.

The following lemma shows that we do not lose any precision by using the discretized $\mathcal{H}$. Here, for any vector $\overline{w}$ of predicted weights, $\mathcal{I}_{\overline{w}}$ denotes the set of prediction mandatory vertices. In particular, $\mathcal{H}$ is now finite.

Lemma 4.5.3. *For a given instance of the hypergraph orientation problem with intervals $\mathcal{I}$, let $\overline{w}$ be a vector of predicted weights that is not contained in the discretized $\mathcal{H}$. Then, there is a $\overline{w}' \in \mathcal{H}$ such that $\mathcal{I}_{\overline{w}} = \mathcal{I}_{\overline{w}'}$.*

The lemma implies that, for each $\overline{w}$, there is an $\overline{w}' \in \mathcal{H}$ that has the same error w.r.t. k_M as $\overline{w}$. Thus, there always exists an element $\overline{w}$ of the discretized $\mathcal{H}$ such that $\overline{w}$ minimizes the expected error over all possible vectors of predicted weights.

Proof of Lemma 4.5.3. Given the vector of predicted weights $\overline{w}$, we construct a vector $\overline{w}' \in \mathcal{H}$ that satisfies the lemma.

For each $\overline{w}_v$, we construct $\overline{w}'_v$ as follows: If $\overline{w}_v = B_j$ for some $j \in \{1, \dots, l\}$, then we set $\overline{w}'_v = B_j$, where B_j is defined as in the definition of the discretized $\mathcal{H}_v$. Otherwise it must hold $\overline{w}_v \in (B_j, B_{j+1})$ for some $j \in \{1, \dots, l\}$, and we set $\overline{w}'_v = h_j$, where h_j is defined as in the discretization. Then, for each $u \in V \setminus \{v\}$, it holds $\overline{w}_v \in I_u$ if and only if $\overline{w}'_v \in I_u$.

We show that each $v \in \mathcal{I}_{\overline{w}}$ is also contained in $\mathcal{I}_{\overline{w}'}$. Since v is mandatory assuming precise weights $\overline{w}$, Lemma 2.3.5 implies that there is a hyperedge S such that either (i) $\overline{w}_v$ is the minimum weight of S and $\overline{w}_u \in I_v$ for some $u \in S \setminus \{v\}$ or (ii) $\overline{w}_v$ is not the minimum weight of S but contains the weight $\overline{w}_u$ of the vertex u with minimum weight $\overline{w}_u$ in S.

Assume v satisfies case (i) for the predicted weights $\overline{w}$. By construction of $\overline{w}'$ it then also holds $\overline{w}'_u \in I_v$. Thus, if v has minimum weight in S for the weights $\overline{w}'$, then $v \in \mathcal{I}_{\overline{w}'}$ by Lemma 2.3.5. Otherwise, some $v' \in S \setminus \{v\}$ must have the minimum weight $\overline{w}'_{v'}$ in S for the weights $\overline{w}'$. Since v has minimum weight for the weights $\overline{w}$, it must hold $\overline{w}'_{v'} < \overline{w}'_v$ but $\overline{w}_{v'} \geq \overline{w}_v$. By construction of $\overline{w}'$, this can only be the case if $\overline{w}'_{v'}, \overline{w}_{v'} \in I_v$. This implies that v satisfies case (ii) for the weights $\overline{w}'$ and, therefore $v \in \mathcal{I}_{\overline{w}'}$ by Lemma 2.3.5.

Assume v satisfies case (ii) for the predicted weights $\overline{w}$. By construction of $\overline{w}'$ it then also holds $\overline{w}'_u \in I_v$. Thus, if u has minimum weight in S for the weights $\overline{w}'$, then $v \in \mathcal{I}_{\overline{w}'}$. Otherwise, some $u' \in S \setminus \{u\}$ must have minimum weight in S for the weights $\overline{w}'$. If $u' = v$, then v satisfies case (i) for the weights $\overline{w}'$ and, therefore, $v \in \mathcal{I}_{\overline{w}'}$ by Lemma 2.3.5. If $u' \neq u$, then it must hold $\overline{w}'_{u'} < \overline{w}'_u$ but $\overline{w}_{u'} \geq \overline{w}_u$ as u has minimum weight in S for weights $\overline{w}$ but u' has minimum weight in S for weights $\overline{w}'$. By construction and since $\overline{w}_u \in I_v$, this can only be the case if $\overline{w}'_{u'}, \overline{w}_{u'} \in I_v$. This implies that v satisfies case (ii) for the weights $\overline{w}'$ and, therefore $v \in \mathcal{I}_{\overline{w}'}$ by Lemma 2.3.5.

Symmetrically, we can show that each $v \in \mathcal{I}_{\overline{w}'}$ is also contained in $\mathcal{I}_{\overline{w}}$, which implies $\mathcal{I}_{\overline{w}} = \mathcal{I}_{\overline{w}'}$. □

Recall that k_M is defined as $k_M = |\mathcal{I}_P \Delta \mathcal{I}_R|$, where $\mathcal{I}_P$ is the set of predictions mandatory elements and $\mathcal{I}_R$ is the set of mandatory elements. In contrast to learning predictions $\overline{w}$ w.r.t. k_h, a vertex $v \in \mathcal{I}$ being part of $\mathcal{I}_P \Delta \mathcal{I}_R$ depends not only on $\overline{w}_v$ and w_v, but on the predicted and precise weights of vertices $V \setminus \{v\}$. Thus, the events of v and u with $v \neq u$ being part of $\mathcal{I}_P \Delta \mathcal{I}_R$ are not necessarily independent. Therefore, we cannot separately learn the predicted weights $\overline{w}_v$ for each $v \in V$, which is a major difference to the proof for k_h in the previous section.

Since the discretized $\mathcal{H}$ is still finite and k_M is bounded by $[0, n]$, we still can apply *empirical risk minimization (ERM)* to achieve guarantees similar to the ones of Theorem 4.5.1. However, because we cannot learn the $\overline{w}_v$ separately, we would have to find the element of $\mathcal{H}$ that minimizes the empirical error. ERM first i.i.d. samples a training set $S = \{w^1, \dots, w^m\}$ of m precise weights vectors from D. Then, it returns the predicted weights $\overline{w} \in \mathcal{H}$ that minimizes the *empirical error* $k_S(\overline{w}) = \frac{1}{m}\sum_{j=1}^{m} k_M(w^j, \overline{w})$. As $\mathcal{H}$ is of exponential size, a straightforward implementation of ERM requires exponential running time. We use this straightforward implementation to prove the following theorem.

Theorem 4.5.4. *Given a fixed instance of the hypergraph orientation problem under explorable uncertainty with graph $G = (V, E)$ and intervals I_v for all $v \in V$. For any $\varepsilon, \delta \in (0, 1)$, there exists a learning algorithm that, using a training set of size m, returns predictions $\overline{w} \in \mathcal{H}$, such that $\mathbb{E}_{w \sim D}[k_M(w, \overline{w})] \leq \mathbb{E}_{w \sim D}[k_M(w, \overline{w}^*)] + \varepsilon$ holds with probability at least $(1 - \delta)$, where $\overline{w}^* = \arg\min_{\overline{w}' \in \mathcal{H}} \mathbb{E}_{w \sim D}[k_M(w, \overline{w}')]$. The sample complexity is $m \in \mathcal{O}\left(\frac{(n \cdot \log(n) - \log(\delta)) \cdot n^2}{\varepsilon^2}\right)$ and the running time is exponential in n.*

Proof. Since $|\mathcal{H}| \in \mathcal{O}(n^n)$ and k_M is bounded by $[0, n]$, ERM achieves the guarantee of the theorem with a sample complexity of $m \in \mathcal{O}\left(\frac{(n \cdot \log(n) - \log(\delta)) \cdot (n)^2}{(\varepsilon)^2}\right)$; for details we refer to [SB14; Vap92]. The prediction $\overline{w} \in \mathcal{H}$ that minimizes the empirical error $k_S(\overline{w}) = \frac{1}{m}\sum_{j=1}^{m} k_M(w^j, \overline{w})$, where $S = \{w^1, \dots, w^m\}$ is the training set, can be computed by iterating through all elements of $S \times H$. Thus, the running time of ERM is polynomial in m but exponential in n. □

To circumvent the exponential running time, we present an alternative approach. In contrast to k_h, for a fixed realization, the value k_M only depends on $\mathcal{I}_P$. Instead of showing the learnability of the predicted weights, we prove that the set $\mathcal{I}_P$ that leads to the smallest expected error can be (approximately) learned. To be more specific, let $\mathcal{P}$ be the power set of V, let $\mathcal{I}_w$ denote the set of mandatory vertices for the realization with precise weights w, and let $k_M(\mathcal{I}_w, P)$ with $P \in \mathcal{P}$ denote the mandatory query distance under the assumption that $\mathcal{I}_P = P$ and $\mathcal{I}_R = \mathcal{I}_w$. Since w is drawn from D, the value $\mathbb{E}_{w \sim D}[k_M(\mathcal{I}_w, P)]$ is a random variable. We show the following theorem.

Theorem 4.5.5. *Given a fixed instance of the hypergraph orientation problem under explorable uncertainty with graph $G = (V, E)$ and intervals I_v for all $v \in V$. For any $\varepsilon, \delta \in (0, 1)$, there exists a learning algorithm that, using a training set of size $m \in \mathcal{O}\left(\frac{(n - \log(\delta)) \cdot n^2}{\varepsilon^2}\right)$, returns a predicted set of mandatory vertices $P \in \mathcal{P}$ in time polynomial in n and m, such that $\mathbb{E}_{w \sim D}[k_M(\mathcal{I}_w, P)] \leq \mathbb{E}_{w \sim D}[k_M(\mathcal{I}_w, P^*)] + \varepsilon$ holds with probability at least $(1 - \delta)$, where $P^* = \arg\min_{P' \in \mathcal{P}} \mathbb{E}_{w \sim D}[k_M(\mathcal{I}_w, P')]$.*

Note that this theorem only allows us to learn a set of prediction mandatory vertices P that (approximately) minimizes the expected mandatory query distance. It does not, however, allow us to learn the predicted weights $\overline{w}$ that lead to the set of prediction mandatory vertices P. In particular, it can be the case, that no such predicted weights exist. Thus, there may not be a realization with precise weights w and $k_M(\mathcal{I}_w, P) = 0$. On the other hand, learning P already allows us to execute Algorithm 11 for the hypergraph orientation problem. Applying Algorithm 12 for the sorting problem would require knowing the corresponding predicted weights $\overline{w}$.

Proof of Theorem 4.5.5. We again show that the basic *empirical risk minimization (ERM)* algorithm already satisfies the lemma. ERM first i.i.d. samples a training set $S = \{w^1, \dots, w^m\}$ of m precise weight vectors from D. Then, it returns the $P \in \mathcal{P}$ that minimizes the *empirical error* $k_S(P) = \frac{1}{m}\sum_{j=1}^{m} k_M(\mathcal{I}_{w^j}, P)$. Since $\mathcal{P}$ is of exponential size, i.e., $|\mathcal{P}| \in \mathcal{O}(2^n)$, we cannot afford to naively iterate through $\mathcal{P}$ in the second stage of ERM, but have to be more careful.

By definition, $\mathcal{P}$ contains $\mathcal{O}(2^n)$ elements and, thus, is finite. Since $\mathcal{P}$ is finite, and the error function k_M is bounded by the interval $[0, n]$, it satisfies the *uniform convergence property* (cf. [SB14]). This implies that, for

$$m = \left\lceil \frac{2\log(2|\mathcal{P}|/\delta)n^2}{\varepsilon^2} \right\rceil \in \mathcal{O}\left(\frac{(n - \log(\delta)) \cdot n^2}{\varepsilon^2}\right),$$

it holds $\mathbb{E}_{w\sim D}[k_M(\mathcal{I}_w, P)] \leq \mathbb{E}_{w\sim D}[k_M(\mathcal{I}_w, P^*)] + \varepsilon$ with probability at least $(1-\delta)$, where P is the set $P \in \mathcal{P}$ learned by ERM (cf. [SB14; Vap99]).

It remains to show that we can compute the set $P \in \mathcal{P}$ that minimizes the empirical error $k_S(P) = \frac{1}{m}\sum_{j=1}^m k_M(\mathcal{I}_{w^j}, P)$ in time polynomial in n and m. For each v, let $p_v = |\{\mathcal{I}_{w^j} \mid 1 \leq j \leq m \wedge v \in \mathcal{I}_{w^j}\}|$ and let $q_v = m - p_v$. For an arbitrary $P \in \mathcal{P}$, we can rewrite $k_S(P)$ as follows:

$$\begin{aligned} k_S(P) &= \frac{1}{m}\sum_{j=1}^m k_M(\mathcal{I}_{w^j}, P) = \frac{1}{m}\sum_{j=1}^m |\mathcal{I}_{w^j} \Delta P| \\ &= \frac{1}{m}\sum_{j=1}^m |P \setminus \mathcal{I}_{w^j}| + |\mathcal{I}_{w^j} \setminus P| \\ &= \frac{1}{m}\left(\sum_{v\in P} q_v + \sum_{v \notin P} p_v\right). \end{aligned}$$

A set $P \in \mathcal{P}$ minimizes the term $k_S(P) = \frac{1}{m}(\sum_{v\in P} q_v + \sum_{v\notin P} p_v)$, if and only if, $q_v \leq p_v$ holds for each $v \in P$. Thus, we can compute the $P \in \mathcal{P}$ that minimizes $k_S(P)$ as follows:

1. Compute q_v and p_v for each $I_v \in \mathcal{I}$.
2. Return $P = \{v \in V \mid q_v \leq p_v\}$.

Since this algorithm can be executed in time polynomial in n and m, the theorem follows. □

4.6 Concluding Remarks

In this chapter, we showed how to exploit access to untrusted predictions to circumvent known lower bounds for hypergraph orientation and sorting under explorable uncertainty and sparked the discussion on error measures by presenting two new metrics, the hop distance and the mandatory query distance, tailored to problems under explorable uncertainty. We proved relations between the different errors and fully characterized the best possible consistency and robustness tradeoffs for the respective errors.

For most of our algorithmic results, we assumed access to predictions on all precise weights. However, for the k_M-depend algorithm for hypergraph orientation, we noticed that access to only the set of prediction mandatory vertices suffices to achieve the best possible tradeoff w.r.t. the mandatory query distance. This raises the question whether there exist further prediction models that require less information but still allow good consistency and robustness results. As a next research step, we suggest investigating such prediction models.

Furthermore, it would be interesting to investigate what tradeoff bounds are possible for the sorting problem if we only have access to the set of prediction mandatory vertices instead of the predicted weights.

Chapter 5

Learning-Augmented Algorithms for Minimum Spanning Tree with Uncertainty

In this chapter, we study how to utilize (possibly erroneous) predictions to solve the minimum spanning tree problem under explorable uncertainty, a fundamental combinatorial optimization problem that has been central also to the research area of explorable uncertainty. We are given a (multi)graph with uncertain edge weights that can be revealed via queries. Our aim is to minimize the number of queries necessary to obtain sufficient information for identifying a minimum spanning tree. For all integral $\gamma \geq 2$, we present algorithms that are γ-robust and $(1 + \frac{1}{\gamma})$-consistent, meaning that they use at most $\gamma|\mathrm{OPT}|$ queries if the predictions are arbitrarily wrong and at most $(1 + \frac{1}{\gamma})|\mathrm{OPT}|$ queries if the predictions are correct, where $|\mathrm{OPT}|$ is the optimal number of queries for the given instance. We show that this tradeoff is best possible. Furthermore, we argue that the *hop distance* (see also Chapter 4) is a useful measure for the amount of prediction error and design algorithms with performance guarantees that degrade smoothly with the hop distance. Our results demonstrate that access to untrusted predictions can help to circumvent the known lower bound of two (see also Section 2.2.2), without any degradation of the worst-case ratio. In the process, we provide new structural insights for the minimum spanning tree problem under explorable uncertainty that might be useful in the context of query-based algorithms regardless of predictions.

Bibliographic remark: This chapter is mainly based on joint work with T. Erlebach, M. de Lima and N. Megow [Erl+22]. Some results are based on a different joint work with the same group of authors [Erl+23; Erl+20]. Therefore, some parts correspond to or are identical with [Erl+23; Erl+22; Erl+20].

Contents

5.1 Introduction

We continue our study of learning-augmented algorithms in the area of optimization under explorable uncertainty and focus on the fundamental *minimum spanning tree (MST) problem*. Recall that in this problem we are given a (multi)graph $G = (V, E)$ with uncertain *precise edge weights* $w_e \in \mathbb{R}_+$ for the edges $e \in E$. The edge weights are initially unknown and, for each edge e, we instead are given an uncertainty interval I_e that contains the unknown precise edge weight w_e and is either open or trivial, i.e., $I_e = (L_e, U_e)$ with $w_e \in I_e$ or $I_e = \{w_e\}$. An edge e is called trivial if the corresponding uncertainty interval I_e is trivial. We call L_e and U_e the *lower* and *upper limit* of I_e. If $I_e = \{w_e\}$, then $U_e = L_e = w_e$. A *query* of edge e has cost c_e, reveals the precise weight w_e and, thus, reduces the corresponding uncertainty interval to $I_e = \{w_e\}$.

Our task is to determine a minimum spanning tree with respect to the initially uncertain precise weights w_e. A *spanning tree* of G is a subgraph of G that contains no cycles and connects all the vertices of G, and a *minimum spanning tree (MST)* is a spanning tree of G with minimum total edge weight. We characterize a spanning tree by its edge set and say that $T \subseteq E$ is a spanning tree if the subgraph $G' = (V, T)$ is a spanning tree. For the minimum spanning tree problem, a query set is called *feasible* if it reveals sufficient information to identify an MST (not necessarily the precise weight of the MST). Our goal is to find a feasible query set of minimum cost.

To more formally define feasible and optimal query sets, we say that a query set $Q \subseteq E$ is *feasible* if there exists a set of edges $T \subseteq E$ such that T is an MST for the precise weights w_e of all $e \in Q$ and *every possible* combination of edge weights in I_e for the *unqueried* edges $e \in E \setminus Q$. That is, querying a feasible query set Q must give us sufficient information to identify a spanning tree T that is an MST for the precise weights no matter what the precise weights of the unqueried edges $E \setminus Q$ actually are. We refer to Section 2.1.3 for an example instance and feasible query set. A feasible query set Q is *optimal* if it has minimum cost $c(Q) = \sum_{e \in Q} c_e$ among all feasible query sets. In this chapter, we consider unit query costs, i.e., $c_e = 1$ for all $e \in E$, so the cost of a query set Q is equal to its cardinality $|Q|$.

As in all previous chapters, we study *adaptive strategies* that make queries sequentially and utilize precise weights revealed by previous queries to decide upon the next query. As there exist input instances that are impossible to solve without querying all edges, we evaluate such adaptive algorithms in an *instance-dependent* manner: For each input, we compare the number of queries made by an algorithm with the best possible number of queries *for that input*, using *competitive analysis* (see also Section 2.2 for a formal definition). For the sake of readability, we briefly restate the definition. Recall that, for a given problem instance, OPT denotes an arbitrary optimal query set. An algorithm is *ρ-competitive* if it, for any problem instance, has query cost at most $\rho \cdot c(\mathrm{OPT})$. For uniform query costs as considered in this chapter, an algorithm is ρ-competitive if it, for any problem instance, executes at most $\rho \cdot |\mathrm{OPT}|$ queries. The *competitive ratio* of an algorithm is the minimum ρ for which the algorithm is ρ-competitive.

While MST under explorable uncertainty is not a classical online problem where the input is revealed passively over time, the query results are uncertain and, to a large degree, dictate whether decisions to query certain edges were good or not. For analyzing an algorithm, it is natural to assume that the query results are determined by an adversary. This gives the problem a clear online flavor and prohibits the existence of 1-competitive algorithms even if we have unlimited running time and space [Hof+08]. We note that competitive algorithms in general do not have any running time requirements, but all our algorithms for the MST problem run in polynomial time.

The MST problem is among the most widely studied problems in the research area of explorable uncertainty and has been a cornerstone in the development of algorithmic approaches and lower bound techniques [Hof+08; EH14; EH15; MMS17; FMM20; MS19; MÇ22]. The best known deterministic algorithm for MST with uncertainty is 2-competitive, and no deterministic algorithm can be better [Hof+08] (see Section 2.2.2 for the lower bound instance). For randomized algorithms, there is a lower bound of 1.5 on the competitive ratio and an upper bound of 1.707 [MMS17]. If the input graph is a cactus graph, then there is a best possible 1.5-competitive randomized algorithm for arbitrary query costs [MÇ22]. Further work considers the non-adaptive problem, which has a very different flavor [MS19]. The offline problem, i.e., the problem of computing an optimal feasible query set while knowing the query results in advance, can be solved to optimality in polynomial time [EH14]. This is in contrast to the hypergraph orientation problem for which we showed NP-hardness of the offline problem in Chapter 3. Erlebach et al. [Hof+08] also considered the MST problem in the euclidean plane with uncertain vertex positions and gave a 4-competitive witness set algorithm.

In this chapter, we once more consider the *learning-augmented setting* (see also Section 2.2.3) and assume that an algorithm has, for each edge e, access to a prediction $\overline{w}_e \in I_e$ for the unknown precise edge weight w_e. Since these predictions might be wrong, an algorithm still has to execute queries in order to guarantee that it finds an MST w.r.t. the initially unknown precise edge weights. However, the prediction can be used to select the query strategy and, if they are of high accuracy, help to reduce the query costs.

These predictions could for example be obtained by using machine learning (ML) methods. Given the tremendous progress in artificial intelligence and ML in recent decades, it seems reasonable to expect that the predictions are of good accuracy, but there is no guarantee and the predictions might be completely wrong. This lack of provable performance guarantees for ML often causes concerns regarding how confident one can be that an ML algorithm will perform sufficiently well in all circumstances. We address the very natural question whether the availability of such predictions can be exploited by query algorithms for the MST problem under explorable uncertainty. Ideally, an algorithm should perform very well if predictions are accurate, but even if they are arbitrarily wrong, the algorithm should not perform worse than an algorithm without access to predictions.

Building on the algorithmic techniques and further results introduced in the previous chapter for the hypergraph orientation problem, we design learning-augmented algorithms for the minimum spanning tree problem under explorable uncertainty that improve upon the adversarial lower bound if the predictions are of high accuracy while at the same time achieving provable performance guarantees even for arbitrarily bad predictions. While our algorithms for the MST problem will reuse some ideas and techniques as used in the previous chapter, they require substantial additional work and structural results specific to the minimum spanning tree problem. During the course of this chapter, we will discuss the similarities and differences between both problems.

To analyze the performance of our algorithms, we again adopt the notions of *consistency* and *robustness* as introduced in [LV21; PSK18] (see also Section 2.2.3 for a formal definition). Recall that 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. Furthermore, we are interested in a smooth transition between the case with correct predictions and the case with arbitrarily wrong predictions. We aim for performance guarantees that degrade gracefully with increasing prediction error. To this end, we will consider the same error measures as introduced in the previous chapter.

Given predicted weights for the uncertainty intervals, it is tempting to simply run an optimal algorithm under the assumption that the predictions are correct. This corresponds to the offline problem using the predicted weights as precise weights and can be solved in polynomial time using the algorithm by Erlebach et al. [EH14]. While this is optimal with respect to consistency, it might give arbitrarily bad solutions in the case when the predictions are faulty. Instead of blindly trusting the predictions, we need more sophisticated strategies to be robust against prediction errors. This chapter is dedicated to designing such strategies, which requires new lower bounds on an optimal solution, new structural insights, and new algorithmic techniques.

5.1.1 Our Results

We give algorithms for the MST problem with uncertainty that are parameterized by a hyperparameter γ reflecting the user's confidence in the accuracy of the predictor. For any integral $\gamma \geq 2$, we present a $(1 + \frac{1}{\gamma})$-consistent and γ-robust algorithm, and show that this is the best possible tradeoff between consistency and robustness. In particular, for $\gamma = 2$, we obtain a 2-robust and 1.5-consistent algorithm. It is worth noting that this algorithm achieves the improved competitive ratio of 1.5 for accurate predictions while maintaining the best possible worst-case ratio of 2.

Our main result is a second and different algorithm with a more fine-grained performance analysis which obtains a guarantee that improves with the accuracy of the predictions. Similar to the hypergraph orientation problem of the previous chapter, very natural, simple error measures such as the number of inaccurate predictions or the ℓ_1-norm of the difference between predicted and precise weights turn out to prohibit any reasonable error-dependency. Therefore, we consider again the *hop distance* k_h (see also Chapter 4), which takes structural insights about the uncertainty intervals into account. Our main result is a learning-augmented algorithm with a competitive ratio with a linear error-dependency $\min\{(1 + \frac{1}{\gamma}) + \frac{5 \cdot k_h}{|\mathrm{OPT}|}, \gamma + 1\}$, for any integral $\gamma \geq 2$. This result nicely illustrates the usefulness of the hop distance as a problem independent error measure for problems under explorable uncertainty with predictions.

We remark that PAC-learnability of the predictions and the removal of the integrality condition in the upper bounds on parameter γ via randomization can be shown analogously to the previous chapter.

5.1.2 Outline

We start the chapter in Section 5.2 by giving a lower bound on the best possible tradeoff between consistency and robustness. Furthermore, we adjust the error measures of the previous chapter for the MST problem and discuss their limits. To conclude the first section, we review existing results for the MST problem under explorable uncertainty that will later on be used in the design of our learning-augmented algorithms. Afterwards, in Section 5.3, we give an

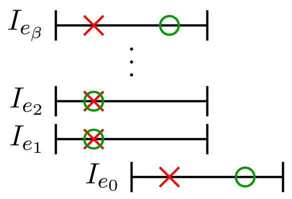


FIGURE 5.1: Uncertainty intervals for the lower bound example of Theorem 5.2.1. Red crosses indicate predicted weights and green circles show precise weights.

overview of the basic algorithmic framework that our learning-augmented algorithms will implement.

In Section 5.4 we prove several new structural results that will be the foundation of our algorithms and their analysis. These structural results might be of independent interest also for the MST problem under explorable uncertainty without predictions. In Section 5.5 we give a first algorithm that transforms arbitrary instances of the MST problem with predictions into instances with a "nice" structure. The algorithm does so while achieving a guarantee in range of the optimal consistency and robustness tradeoff.

Afterwards, we design algorithms for instances with such a nice structure. In Sections 5.6 and 5.7 we separately give and analyze such algorithms with and without error-dependent guarantees. In combination with the algorithm that transforms arbitrary instances into nice instances, these algorithms achieve our algorithmic main results.

5.2 Preliminaries

We start the chapter by reviewing existing results on the MST problem under explorable uncertainty and also give new preliminary results that lay the foundation for our learning-augmented algorithms.

First, we give a lower bound on the best possible tradeoff between consistency and robustness for the MST problem under explorable uncertainty with predictions. Afterwards, we revisit the error measures as introduced in the previous chapter and adjust them for the MST problem.

Finally, we summarize (and slightly extend) existing structural results for the MST problem under explorable uncertainty. Later in this chapter, we will further extend these structural results and exploit them in our learning-augmented algorithms.

5.2.1 Lower Bound on the Consistency and Robustness Tradeoff

We give the following lower bound on the optimal tradeoff between consistency and robustness for the MST problem under explorable uncertainty with predictions. The proof of the theorem uses the same idea as the corresponding lower bound for hypergraph orientation in the previous chapter but slightly adjusts it to the MST problem.

Theorem 5.2.1. *Let $\beta \geq 2$ be a fixed integer. For the MST problem under explorable uncertainty with predictions, 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 $\alpha = 1 + \frac{1}{\beta'}$ for some integer $\beta' \geq 1$ is β-robust for $\beta < \beta'$.*

Proof. Assume, for the sake of contradiction, that there is a deterministic β-robust algorithm that is α-consistent with $\alpha = 1 + \frac{1}{\beta} - \varepsilon$ for some $\varepsilon > 0$. Consider the instance that consists of a single cycle with the edges $e_0, e_1, \ldots, e_\beta$ and uncertainty intervals as indicated in Figure 5.1.

To solve this given instance, an algorithm has to query edges until it identifies an edge of maximum precise weight in the cycle. Then, the set of all edges except this one clearly forms an MST with respect to the precise weights. Observe that, if the predicted weights of the figure are indeed correct, the optimal query set is $\{e_1, \dots, e_\beta\}$ as we cannot prove that the weights of these edges are smaller than the predicted weight of e_0 without querying them. For the precise weights as indicated in the figure on the other hand, it suffices to query e_0 in order to prove that it has maximum precise weight.

To be α-consistent, the algorithm must query the edges $\{e_1, \dots, e_\beta\}$ first, as otherwise it would query $\beta + 1$ edges in case all predictions are correct, while there is an optimal query set of size β, which would imply a consistency of $1 + \frac{1}{\beta} > \alpha$.

Suppose w.l.o.g. that the algorithm queries the edges $\{e_1, \dots, e_\beta\}$ in order of increasing indices. Consider the adversarial choice $w_{e_i} = \overline{w}_{e_i}$, for $i = 1, \dots, \beta - 1$, and then $w_{e_\beta} \in I_{e_0}$ and $w_{e_0} \notin I_{e_1} \cup \dots \cup I_{e_\beta}$. This forces the algorithm to query also e_0, while an optimal solution only queries e_0. Thus, any such algorithm has robustness at least $\beta + 1$, a contradiction.

The second part of the theorem directly follows from the first part and the known general lower bound of 2 on the competitive ratio [Hof+08; Kah91]; see also Section 2.2.2. Assume there is an α-consistent deterministic algorithm with $\alpha = 1 + \frac{1}{\beta'}$ for some integer $\beta' \geq 1$. If $\beta' < 2$, then the statement follows from the lower bound of 2, so assume $\beta' \geq 2$. Consider the instance above with $\beta = \beta' - 1$. Then the algorithm has to query vertices $\{1, \dots, \beta\}$ first to ensure α-consistency, as otherwise it would have a competitive ratio of $\frac{\beta+1}{\beta} > 1 + \frac{1}{\beta'} = \alpha$ in case that all predictions are correct. By the argumentation above, the robustness factor of the algorithm is at least $\beta + 1 = \beta'$.

□

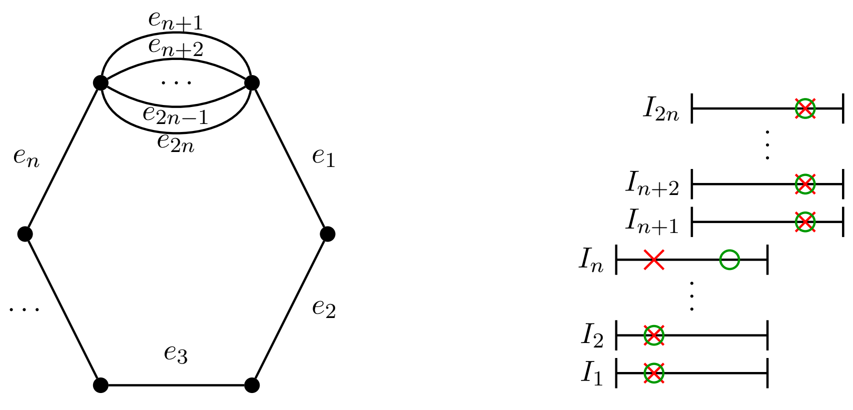
This tradeoff lower bound again proves that fully trusting the predictions leads to an arbitrarily bad robustness and motivates the usage of more involved algorithmic techniques. A main goal of this chapter is to design algorithms that match this tradeoff lower bound.

5.2.2 Error Metrics

Since consistency and robustness only capture the extremes in terms of prediction quality, we aim for a more fine-grained performance analysis giving guarantees that depend on the quality of the predictions. To achieve this, we need error measures that capture the quality of the predictions. A very natural, simple error measure is the number of inaccurate predictions $k_\# = |\{e \in E \mid w_e \neq \overline{w}_e\}|$. However, as we have seen for the hypergraph orientation problem in Chapter 4, we can show that even for $k_\# = 1$ the competitive ratio cannot be better than the known bound of 2, even if $|\text{OPT}| \in \Omega(|E|)$ for any even number of edges $|E|$.

Lemma 5.2.2. *Even if $k_\# = 1$, any deterministic algorithm for the MST problem under uncertainty with predictions has competitive ratio $\rho \geq 2$. This result holds even for instances with $|\text{OPT}| \in \Omega(|E|)$ and an arbitrarily large even number of edges $|E|$.*

Proof. Consider an input graph that consists of a path P with n edges $e_1, e_2, \dots, e_n$ and n parallel edges $e_{n+1}, e_{n+2}, \dots, e_{2n}$ between the endpoints of the path. See Figure 5.2a for an illustration of this graph. Each edge e_i is given with an uncertainty interval I_i and a predicted weight $\overline{w}_{e_i}$. The structure of the intervals and their predicted weights is shown in Figure 5.2b. If all the predictions are correct, the MST consists of the path P, and this can be verified either by querying the n edges of P, or by querying the n parallel edges between the endpoints of P. Assume w.l.o.g. that an algorithm queries the path edges in the order $e_1, e_2, \dots, e_n$ and the parallel edges in the order $e_{n+1}, e_{n+2}, \dots, e_{2n}$. Before the algorithm queries e_n or e_{2n}, the adversary sets all predictions as correct, so the algorithm will eventually have to query e_n or e_{2n}. If the algorithm queries e_n before e_{2n}, then the adversary chooses a



(A) Input graph. (B) Interval structure.

FIGURE 5.2: Instance for a lower bound based on the number of inaccurate predictions. (a) Input graph for the MST problem under explorable uncertainty. (b) Uncertainty intervals of the $2n$ edges. Red crosses indicate predicted weights, and green circles show precise weights.

weight $w_{e_n} \in I_{e_{n+1}} \cap \ldots \cap I_{e_{2n}}$ for e_n (illustrated in Figure 5.2b). This forces the algorithm to also query $e_{n+1}, \ldots, e_{2n}$ and the adversary picks predicted weights for the remaining parallel edges as correct, so the optimal solution only queries $e_{n+1}, \ldots, e_{2n}$. A symmetric argument holds if the algorithm queries e_{2n} before e_n. In either case, the MST is P and the optimal query set consists of n queries, while the algorithm is forced to make $2n$ queries. Furthermore, only a single prediction is incorrect, so $k_\# = 1$. □

The reason for the weakness of the measure $k_\#$ is that it completely ignores the interleaving structure of intervals. Similarly, an ℓ_1 error metric such as $\sum_{e \in E} |w_e - \overline{w}_e|$ would not be meaningful because only the *order* of the weights and the interval endpoints matters for our problem.

To address this weakness, we return to the error measure *hop distance* as introduced in the previous chapter for hypergraph orientation and define it for the MST problem.

The definition is quite intuitive even though it requires some technical care to make it precise. If we consider only a single predicted weight $\overline{w}_e$ for some $e \in E$, then, in a sense, this weight predicts the relation of the precise weight w_e to the intervals of edges $e' \in E \setminus \{e\}$. In particular, w.r.t a fixed $e' \in E \setminus \{e\}$, the weight $\overline{w}_e$ predicts whether w_e is left of $I_{e'}$ ($\overline{w}_e \leq L_{e'}$), right of $I_{e'}$ ($\overline{w}_e \geq U_{e'}$), or contained in $I_{e'}$ ($L_{e'} < \overline{w}_e < U_{e'}$). Interpreting the prediction $\overline{w}_e$ in this way, the prediction is "wrong" (w.r.t. a fixed $e' \in E \setminus \{e\}$) if the predicted relation of the precise weight w_e to interval $I_{e'}$ is not actually true, e.g., w_e is predicted to be left of $I_{e'}$ ($\overline{w}_e \leq L_{e'}$) but the actual w_e is either contained in or right of $I_{e'}$ ($w_e > L_{e'}$). Formally, we define the function $k_{e'}(e)$ that indicates whether the predicted relation of w_e to $I_{e'}$ is true ($k_{e'}(e) = 0$) or not ($k_{e'}(e) = 1$). More precisely, $k_{e'}(e) = 1$ if $\overline{w}_e \leq L_{e'} < w_e$, $w_e \leq L_{e'} < \overline{w}_e$, $w_e < U_{e'} \leq \overline{w}_e$ or $\overline{w}_e < U_{e'} \leq w_e$, and $k_{e'}(e) = 0$ otherwise. With the prediction error $k^+(e)$ for a single $e \in E$, we want to capture the number of relations between w_e and intervals $I_{e'}$ with $e' \in E \setminus \{e\}$ that are not accurately predicted. Thus, we define $k^+(e) = \sum_{e' \in E \setminus \{e\}} k_{e'}(e)$. For a set of edges $E' \subseteq E$, we define $k^+(E') = \sum_{e \in E'} k^+(e)$. Consequently, with the error for the complete instance we want to capture the total number of wrongly predicted relations and, therefore, define it by $k_h = k^+(E)$. See Figure 5.3 for an example.

Symmetrically, we can define $k^-(e) = \sum_{e' \in E \setminus \{e\}} k_e(e')$ and $k^-(E') = \sum_{e \in E'} k^-(e)$ for subsets $E' \subseteq E$. Then $k^+(E) = k_h = k^-(E)$ follows by reordering the summations.

We give the following lower bound on the competitive ratio as a function of k_h.

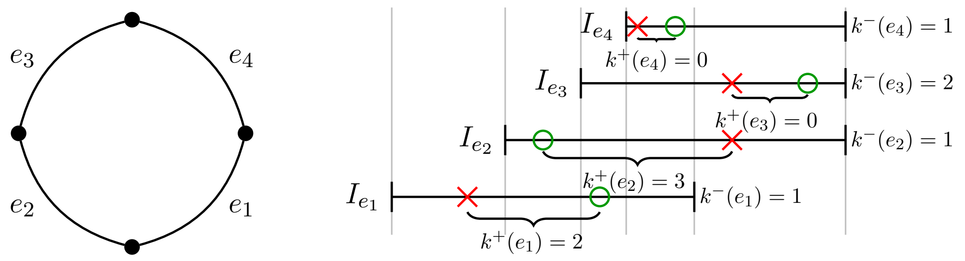


FIGURE 5.3: Example of a single cycle (left) with uncertain edge weights from intersecting intervals $I_{e_1}, I_{e_2}, I_{e_3}, I_{e_4}$ (right). Green circles illustrate precise weights and red crosses illustrate the predicted weights. The predictions have a hop distance of $k_h = \sum_{i=1}^{4} k^-(e_i) = \sum_{i=1}^{4} k^+(e_i) = 5$.

Theorem 5.2.3. *Any deterministic algorithm for MST under explorable uncertainty with predictions has a competitive ratio $\rho \geq \min\{1 + \frac{k_h}{|\mathrm{OPT}|}, 2\}$.*

Proof. Consider the instance of Figure 5.4 with the three edges e_1, e_2 and e_3 of a triangle. Clearly, edge e_3 is part of any MST independent of the precise weights, so it remains to determine which of e_1 and e_2 has larger edge weight. If the algorithm ALG starts querying e_1, then the adversary picks $w_{e_1} \in I_{e_2}$ and the algorithm is forced to query e_2. Then $w_{e_2} \in I_{e_2} \setminus I_{e_1}$, so the optimum queries only e_2. It is easy to see that $k_h = 1$ and, thus, $\frac{|\mathrm{ALG}|}{|\mathrm{OPT}|} = 2 = \min\{1 + \frac{k_h}{|\mathrm{OPT}|}, 2\}$. A symmetric argument holds if the algorithm starts by querying e_2. In that case, $w_{e_2} \in I_{e_1}$ and the algorithm is forced to query e_1. Then $w_{e_1} \in I_{e_1} \setminus I_{e_2}$, so the optimum queries only e_1. Again, $k_h = 1$ and, thus, $\frac{|\mathrm{ALG}|}{|\mathrm{OPT}|} = 2 = \min\{1 + \frac{k_h}{|\mathrm{OPT}|}, 2\}$. Taking multiple copies of this instance (connected by a tree structure), gives the same results for larger values of k_h and $|\mathrm{OPT}|$. □

In Chapter 4, we also considered the error measure *mandatory query distance* k_M for the hypergraph orientation problem under explorable uncertainty with predictions. While this measure can be defined analogously for the MST problem, we will not consider it in this chapter and it remains an open question whether there is a learning-augmented algorithm with a non-trivial error-dependency on k_M.

5.2.3 Witness Sets and Mandatory Edges

In order to design our learning-augmented algorithms for the MST problem under explorable uncertainty, we first review existing results that we will later extend to the problem variant with predictions.

All known algorithms for the problem *without* access to predictions [Hof+08; EH14; MMS17] heavily exploit the concept of witness sets [Bru+05] (see also Section 2.3). Recall that a subset $W \subseteq E$ is a *witness set* if $W \cap Q \neq \emptyset$ for all feasible query sets Q, i.e., every algorithm (including the optimal query set) has to query at least one member of a witness set. Witness sets are the key to the analysis of all known algorithms for the MST problem

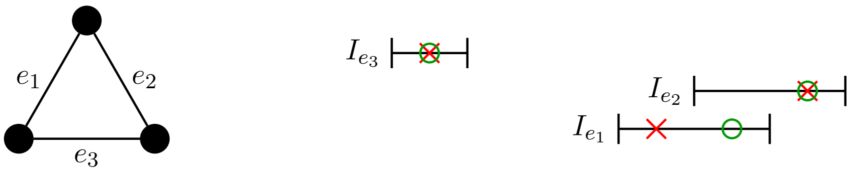


FIGURE 5.4: Lower bound example for the proof of Theorem 5.2.3. Circles illustrate precise weights and crosses illustrate the predicted weights.

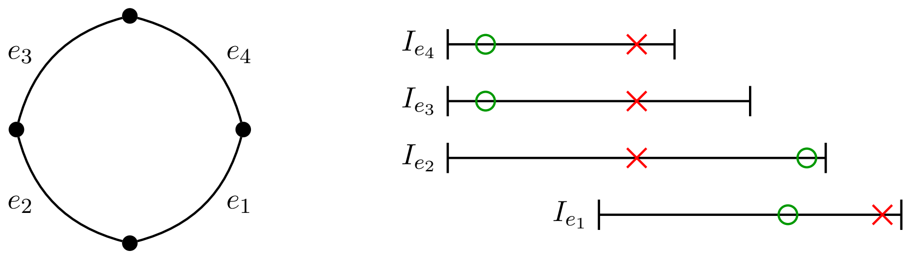


FIGURE 5.5: Example of a single cycle (left) with uncertain edge weights from intersecting intervals $I_{e_1}, I_{e_2}, I_{e_3}, I_{e_4}$ (right). Circles illustrate precise weights and crosses illustrate the predicted weights. For the example, $\{e_1, e_2\}$ is the set of all mandatory edges and $\{e_1\}$ is the set of all edges that would be mandatory if the predictions were correct.

under explorable uncertainty as they allow for a comparison of an algorithm's query set to an optimal solution.

An important special case are witness sets of cardinality one, i.e., edges that are part of every feasible query set. We call such edges *mandatory*. The identification of mandatory edges is especially important, as every algorithm can query mandatory edges without ever worsening its competitive ratio.

For an example of mandatory edges, consider Figure 5.5. In the example, we see the uncertainty intervals and precise weights of four edges that form a simple cycle. We can observe that both e_1 and e_2 are mandatory for this example. To see this, assume that e_1 is not mandatory. Then, there must be a feasible query set Q with $e_1 \notin Q$ for the instance, which implies that $Q = \{e_2, e_3, e_4\}$ must be feasible. But even after querying Q to reveal the precise weights of e_2, e_3 and e_4, it still depends on the still unknown precise weight of e_1 whether there exists an MST T with $e_1 \in T$ (only if $w_{e_1} \leq w_{e_2}$) and/or $e_1 \notin T$ (only if $w_{e_2} \leq w_{e_1}$). Even after querying Q there is no spanning tree T that is an MST for each possible edge weight in I_{e_1} of the unqueried edge e_1 and, thus, Q is not feasible. This implies that e_1 is mandatory, and we can argue analogously that e_2 is mandatory as well.

Lower and Upper Limit Trees In order to identify witness sets and mandatory edges, we consider the *lower and upper limit trees* of a given instance of the MST problem under explorable uncertainty. This structural concept has been introduced by Megow et al. in [MMS17]:

Definition 5.2.4 (Lower and upper limit trees)**.** *Given an instance of the MST problem under explorable uncertainty with graph $G = (V, E)$ and uncertainty intervals I_e for all $e \in E$, a* lower limit tree *of the instance is an MST T_L with respect to the edge weights*

$$w_e^L = \begin{cases} L_e + \epsilon & \text{if } I_e \text{ is non-trivial} \\ w_e & \text{if } I_e \text{ is trivial} \end{cases}$$

for all $e \in E$ and an infinitesimally small $\epsilon > 0$. Similarly, an upper limit tree *of the instance is an MST T_U with respect to the edge weights*

$$w_e^U = \begin{cases} U_e - \epsilon & \text{if } I_e \text{ is non-trivial} \\ w_e & \text{if } I_e \text{ is trivial.} \end{cases}$$

Note that a query to an edge e with a non-trivial uncertainty interval $I_e = (L_e, U_e)$ changes the interval I_e to $I_e = \{w_e\}$ and, therefore, also changes the upper and lower limit of I_e to w_e. By changing the upper and lower limits of an edge, the upper and lower limit trees of the instance can also change. This means that the trees change during the application of a query algorithm. We use the term *current instance* to refer to the problem instance *after* all previous

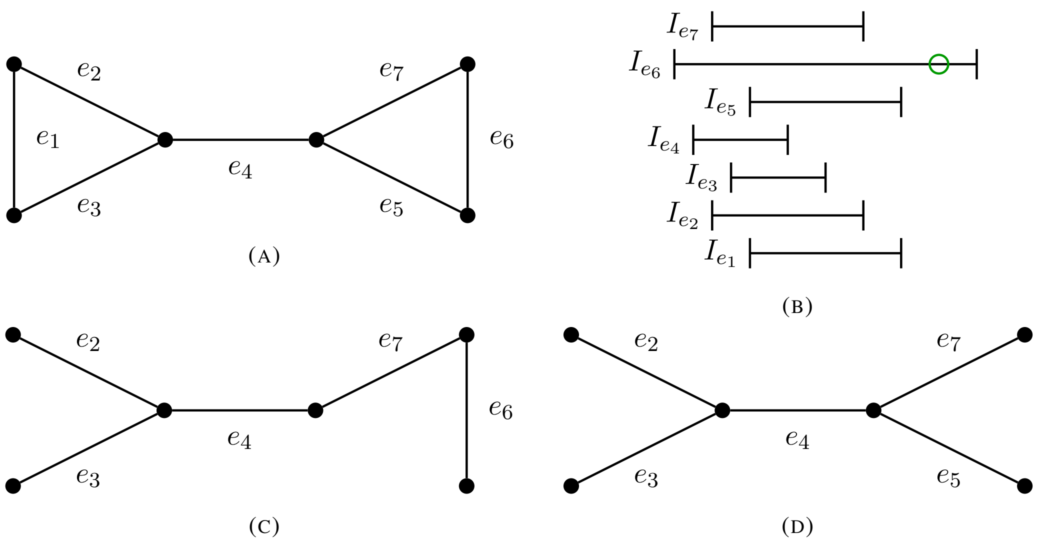


FIGURE 5.6: Part (A) and (B) of the figure show an input instance of the MST problem under explorable uncertainty. Part (C) shows the lower limit tree and part (D) shows the upper limit tree of the instance. After querying edge e_6 to reduce the uncertainty interval of e_6 to the precise weight indicated by the green circle, the upper and lower limit tree both have the form shown in part (D).

queries. With the *current upper and lower limit tree* we refer to the upper and lower limit tree of the current instance. See Figure 5.6 for an example.

Mandatory Edges and Preprocessing For a given instance of the MST problem under explorable uncertainty, fix an upper limit tree T_U and a lower limit tree T_L. A very useful property of T_U and T_L is that all edges in $T_L \setminus T_U$ are mandatory, as was shown in [MMS17].

Lemma 5.2.5 (Megow et al. [MMS17])**.** *Let T_L and T_U be lower and upper limit trees of a given instance of the MST problem under explorable uncertainty with graph $G = (V, E)$ and uncertainty intervals I_e for $e \in E$. Each edge $e \in T_L \setminus T_U$ with a non-trivial uncertainty interval I_e is mandatory.*

Proof. Let T_L and T_U be lower and upper limit trees of a given instance of the MST problem under explorable uncertainty with graph $G = (V, E)$ and uncertainty intervals I_e for $e \in E$.

Assume $T_L \neq T_U$ and let e be an arbitrary edge in $T_L \setminus T_U$ with a non-trivial uncertainty interval. We argue that e is mandatory.

Let X_e denote the set of edges in the cut between the two connected components of the subgraph induced by $T_L \setminus \{e\}$. Since T_L is a lower limit tree, e must be an edge with a minimum lower limit in X_e as otherwise T_L would not be an MST for the edge weights w^L (cf. Definition 5.2.4). Furthermore, for the same reason, there cannot be a trivial edge $e' \in X_e$ with $L_{e'} = w_{e'} = L_e$. This means that if w_e is close enough to L_e, then e has the unique minimum precise weight among the edges in X_e and, thus, has to be part of *every* MST for the instance independent of what the precise weights of the other edges actually are.

Let C_e denote the unique cycle in $T_U \cup \{e\}$. Since T_U is an MST for the edge weights w^U (cf. Definition 5.2.4) and $e \notin T_U$, it must hold that e is an edge of maximum upper limit in C_e. Furthermore, there cannot be a trivial edge $e' \in C_e$ with $U_{e'} = w_{e'} = U_e$. This means that if w_e is close enough to U_e, then e has the unique maximum precise weight among the edges in C_e and, thus, cannot be part of *any* MST for the instance independent of what the precise weights of the other edges actually are.

Without querying e, it is not possible to distinguish between the case where e has to be part of every MST and the case where e cannot be part of any MST. Thus, e is mandatory. □

A direct consequence of this lemma is that an algorithm for the MST problem under explorable uncertainty can always query all non-trivial edges in $T_L \setminus T_U$ without ever worsening its competitive ratio until the lower and upper limit trees are equal for the current instance. We can extend this argument and show that an algorithm can always ensure that T_L and T_U are unique and satisfy $T_L = T_U$ by only querying mandatory edges and contracting/deleting edges with trivial uncertainty intervals.

Lemma 5.2.6. *Given an instance of the MST problem under explorable uncertainty with graph $G = (V, E)$ and uncertainty intervals I_e for $e \in E$. By querying only mandatory edges we can obtain an instance with $T_L = T_U$ such that T_L and T_U are the unique lower limit tree and upper limit tree, respectively.*

Proof. Let T_L be a lower limit tree for a given instance and let T_U be an upper limit tree. According to Lemma 5.2.5, all elements of $T_L \setminus T_U$ are mandatory and we can repeatedly query them for (the adapting) T_L and T_U until $T_L = T_U$. We refer to this process as the *first preprocessing step*.

Consider an $f \in E \setminus T_U$ and the cycle C in $T_U \cup \{f\}$. If f is trivial, then the precise weight w_f is maximal in C and we may delete f without loss of generality. Assume otherwise. If the upper limit of I_f is uniquely maximal in C, then f is not part of any upper limit tree. If there is an $l \in C \setminus \{f\}$ with $U_f = U_l$, then $T'_U = T_U \setminus \{l\} \cup \{f\}$ is also an upper limit tree. Since $T_L \setminus T'_U = \{l\}$, we may execute the first preprocessing step for T_L and T'_U. We repeatedly do this until each $f \in E \setminus T_U$ has the uniquely maximal upper limit in the cycle C in $T_U \cup \{f\}$. Then, T_U is unique.

To achieve uniqueness for T_L, consider some $l \in T_L$ and the cut X of G between the two connected components of $T_L \setminus \{l\}$. If l is trivial, then the precise weight w_l is minimal in X and we may contract l without loss of generality. Assume otherwise. If L_l is uniquely minimal in X, then l is part of every lower limit tree. If there is an $f \in X \setminus \{l\}$ with $L_l = L_f$, then $T'_L = T_L \setminus \{l\} \cup \{f\}$ is also a lower limit tree. Since $T'_L \setminus T_U = \{f\}$ follows from $T_L = T_U$, we may execute the first preprocessing step for T'_L and T_U. We repeatedly do this until L_l for each $l \in T_L$ is uniquely minimal in the cut X of G between the two connected components of $T_L \setminus \{l\}$. Then, T_L is unique. □

The Lemma 5.2.6 can be interpreted as a preprocessing step that allows an algorithm to ensure that the instance has unique upper and lower limit trees T_U and T_L with $T_L = T_U$ without worsening its competitive ratio. Thus, we refer to such instances as *preprocessed*. Going back to the example of Figure 5.6, the initially given instance is not preprocessed as $T_L \neq T_U$. After querying edge e_6, the instance of the example becomes preprocessed.

Definition 5.2.7. *We call an instance of the MST problem under explorable uncertainty* preprocessed, *if the instance has a unique lower limit tree T_L, a unique upper limit tree T_U and satisfies $T_L = T_U$.*

As we remarked before, querying edges can change the lower and upper limit trees of a problem instance. In particular, the lower limit tree for the problem instance *after* querying a feasible query set Q must be an MST with respect to the precise weights. Given an instance with unique $T_L = T_U$, we can show that each $e \in T_L$ that is not part of the MST for the precise weights is mandatory. Similarly, each $e \notin T_L$ that is part of the MST for the precise weights is mandatory as well. A stronger version of this observation is as follows.

Lemma 5.2.8. *Let $G = (V, E)$ be an instance with unique lower and upper limit trees $T_L = T_U$ and let G' be an instance with unique lower and upper limit trees $T'_L = T'_U$ obtained from G by querying set $Q \subseteq E$, then $e \in T_L \Delta T'_L = (T_L \setminus T'_L) \cup (T'_L \setminus T_L)$ implies $e \in Q$.*

Proof. Let $e \in T_L \setminus T'_L$, then $e \in T_L$ and T_L being unique imply that e has the unique minimal lower limit in the cut X_e of G between the two connected components of $T_L \setminus \{e\}$. Thus, e is part of every lower limit tree for G. For e not to be part of T'_L, it cannot have the unique minimal lower limit in the cut X_e of G' anymore. Since querying elements in $X_e \setminus \{e\}$ only increases their lower limits, this can only happen if $e \in Q$.

Let $e \in T'_L \setminus T_L$. Then $T_L = T_U$ and $T'_L = T'_U$ imply $e \in T'_U \setminus T_U$. Since $e \notin T_U$ and T_U is unique, it follows that e has the unique largest upper limit in the cycle C_e of $T_U \cup \{e\}$. Thus, e is not part of any upper limit tree for G. For e to be part of T'_U, it cannot have the unique largest upper limit in the cycle C_e of G' anymore. Since querying elements in $C_e \setminus \{e\}$ only decreases their upper limits, this can only happen if $e \in Q$. $\square$

Identifying Witness Sets and the Witness Set Algorithm The lower and upper limit trees do not only allow us to identify mandatory edges by using the previous two lemmas, but can also be used to identify larger witness sets.

To that end, consider an arbitrary preprocessed instance of the MST problem under explorable uncertainty with the unique lower limit tree T_L. Based on T_L, we can give criteria to identify witness sets and define the witness set algorithm for the MST problem. Let $f_1, \dots, f_l$ denote the edges in $E \setminus T_L$ ordered by non-decreasing lower limits L_{f_i}. Then, C_i with $i \in \{1, \dots, l\}$ denotes the unique cycle in $T_L \cup \{f_i\}$. By definition and since the instance is preprocessed, edge f_i has the unique largest upper *and* lower limit on the cycle C_i. Define $G_i = (V, E_i)$ with $E_i = T_L \cup \{f_1, \dots, f_i\}$ and $G_0 = (V, T_L)$.

Before we consider the criteria to identify witness sets as given in [MMS17], we restate their auxiliary lemma stating that a feasible query set for the complete problem instance must also be feasible for the subproblem instances defined by the subgraphs G_i with $i \in \{1, \dots, l\}$. For a proof of the lemma, we refer to [MMS17].

Lemma 5.2.9 (Megow et al. [MMS17, Lemma 4.1])**.** *Let $i \in \{1, \dots, l\}$. Given a feasible query set Q for the graph $G = (V, E)$ with uncertainty intervals I_e for $e \in E$, the set $Q_i := Q \cap E_i$ is a feasible query set for $G_i = (V, E_i)$ with the same uncertainty intervals I_e for the edges $e \in E_i$.*

The next lemma helps us to iteratively identify minimum spanning trees for subgraphs G_i with increasing i, while only querying witness sets of size at most two until we have found an MST for the complete instance. In particular, if we have already identified an MST T_{i-1} for instance G_{i-1}, the lemma gives us criteria to identify a witness set of size at most two on the unique cycle C in $T_{i-1} \cup \{f_i\}$. For a proof of the lemma, we again refer to [MMS17].

Lemma 5.2.10 (Megow et al. [MMS17, Lemma 4.2 and ff.])**.** *For some realization of edge weights and some $i \in \{1, \dots, \ell - 1\}$, consider the problem instance* after *querying some feasible query set Q_i for graph G_i. Let T_i be the determined MST for graph G_i, let C be the cycle closed by adding f_{i+1} to T_i and let $g \in C \setminus \{f_{i+1}\}$ be an edge with $I_g \cap I_{f_{i+1}} \neq \emptyset$. Then, f_{i+1} has the largest upper limit on cycle C and any feasible query set for G_{i+1} contains f_{i+1} or g. Moreover, if $I_g \subseteq I_{f_{i+1}}$, then f_{i+1} is mandatory.*

Exploiting these two lemmas, the *witness set algorithm* (cf. Section 2.3 for a general, abstract formulation) for the MST problem achieves a competitive ratio of 2 without access to predictions [Hof+08; MMS17]. Given an MST T_{i-1} for subgraph G_{i-1}, the Algorithm 14 finds an MST of instance G_i by considering the unique cycle C in $T_{i-1} \cup \{f_i\}$ and repeatedly querying witness sets $W \subseteq C$ of size at most two until it identifies an edge h of maximum precise weight in C. Then, $T_i = (T_{i-1} \cup \{f_i\}) \setminus \{h\}$ is the MST for instance G_i.

Theorem 5.2.11 (Erlebach et al. [Hof+08], Megow et al. [MMS17])**.** *The witness set algorithm is 2-competitive for the minimum spanning tree problem under explorable uncertainty.*

Algorithm 14: Witness set algorithm for minimum spanning tree under explorable uncertainty.

```
Input: Preprocessed instance of the minimum spanning tree problem under
       explorable uncertainty with graph $G = (V, E)$ and uncertainty intervals $I_e$ for
       all $e \in E$.
Output: A feasible query set $Q$ for the given problem instance.
1  $Q \leftarrow \emptyset$;
2  $T \leftarrow$ lower limit tree $T_L$ of the current instance;
3  $f_1, \dots, f_l \leftarrow$ edges in $E \setminus T_L$ ordered by non-decreasing lower limit $L_{f_i}$;
4  for $i$ from 1 to $l$ do
5      $C \leftarrow$ unique cycle in $T \cup \{f_i\}$;
6      repeat
7          $W \subseteq C \setminus Q \leftarrow$ Witness set of size at most two computed via Lemma 5.2.10;
8          Query $W$;
9          $Q \leftarrow Q \cup W$;
10     until We identified an edge $h$ of maximum precise weight in $C$;
11     $h \leftarrow$ edge of maximum precise weight in $C$;
12     $T \leftarrow (T \cup \{f_i\}) \setminus \{h\}$ ;
13 return $Q$;
```

Proof. To prove the theorem, we can first observe that the current tree T at the beginning of iteration i of the for-loop is an MST for instance $G_{i-1} = T_L \cup \{f_1, \dots, f_{i-1}\}$. We can proof this via induction over i. For $i = 1$, we have that $T = T_L$ is the only spanning tree for instance $G_{i-1} = G_0$ and, thus, also an MST. For larger i, we have $T = (T' \cup \{f_{i-1}\}) \setminus \{h\}$, where T' by induction hypothesis is an MST for instance G_{i-2} and h is an edge of maximum precise weight in the cycle C of $T' \cup \{f_{i-1}\}$ (cf. Line 12). This directly implies that T is an MST for instance G_{i-1}.

The fact that the current tree T at the beginning of each iteration i of the for-loop is an MST for instance G_{i-1} allows us to apply Lemma 5.2.10 on the cycle C in $T \cup \{f_i\}$. So if the algorithm executes queries in Line 8, then it queries a witness set of size at most two. Since all those witness sets are pairwise disjoint, this implies a competitive ratio of at most two.

It remains to argue that the algorithm computes a feasible query set. We already argued that the current T is always an MST for the subinstance G_{i-1}. The argumentation also holds for a hypothetical iteration $l + 1$ and, thus, implies that the final T is an MST for the complete instance. However, this implication relies on the repeat-loop being able to identify an edge h of maximum precise weight in the current cycle C. To finish the proof, we have to argue that whenever we do not know an edge of maximum precise weight in C yet, Lemma 5.2.10 gives us a witness set $W \subseteq C$ to query in Line 8.

To this end, assume that we do not know an edge of maximum precise weight in C yet but also cannot apply Lemma 5.2.10 to identify a witness set $W \subseteq C$. Let f_i denote the edge of maximum upper limit in C. If f_i has a trivial uncertainty interval, either because it was already queried or had a trivial interval initially, then f_i clearly has maximum precise weight in C. Thus, assume otherwise. As we cannot identify a witness set by using Lemma 5.2.10, it must hold that $I_g \cap I_{f_i} = \emptyset$ for all $g \in C \setminus \{f_i\}$. But then, f_i clearly has maximum precise weight in C, a contradiction. □

We remark that, while the witness set algorithm is stated for uniform query costs, it can be extended to arbitrary query costs by using the local ratio technique of Section 2.3.

While the witness set algorithm achieves the best possible adversarial competitive ratio of two, using it as a black box will never improve over 2-consistency in the learning-augmented setting. Since the witness set algorithm does not allow us to obtain an improved consistency and the offline algorithm of [EH14] that blindly trusts the predictions is n-robust by the tradeoff lower bound of Theorem 5.2.1, we need new algorithmic techniques in order to match the best possible consistency and robustness tradeoff. In the following section, we give an overview of these techniques before implementing them in the remainder of this chapter.

5.3 Overview of Techniques

We give an overview of the algorithmic techniques that we use in our learning-augmented algorithms later in this chapter. In order to match the tradeoff lower bound, we aim for $(1 + \frac{1}{\gamma})$-consistent and γ-robust algorithms for each integral $\gamma \geq 2$.

5.3.1 Basic Algorithmic Framework

Our algorithm follows the same basic framework as the learning-augmented algorithms of the previous chapter for hypergraph orientation and sorting under explorable uncertainty. The implementation of the framework however will be very specific to the MST problem under explorable uncertainty.

As for the learning-augmented algorithms of the previous chapter, the basic framework proceeds in two phases: The first phase runs as long as there are prediction mandatory edges, i.e., edges that must be contained in every feasible query set under the assumption that the predictions are correct. Such edges can for example be identified using the offline algorithm by Erlebach et al. [EH14] and we later give further criteria to identify such edges. In this phase, we exploit the existence of those edges and their properties to execute queries with strong local guarantees, i.e., each feasible query set contains a large portion of our queries if the predictions are correct. The basic idea of this phase is analogous to the first phase of the algorithm for hypergraph orientation with a hop distance dependent guarantee. However, the identification of query sets with strong local guarantees is far more involved for the minimum spanning tree problem and a main technical contribution of this chapter.

For the second phase, we observe and exploit that the absence of prediction mandatory queries implies that the predicted optimal solution is a minimum vertex cover in a bipartite auxiliary graph. The challenge here is that the auxiliary graph can change with each wrong prediction. To obtain an error-dependent guarantee we need to adaptively query a dynamically changing minimum vertex cover. This is in contrast to the algorithms for hypergraph orientation of the previous chapter. Those algorithms could afford to non-adaptively query a minimum vertex cover of an auxiliary graph and still guarantee at least 2-robustness. For the minimum spanning tree problem, we show that non-adaptively querying the complete vertex cover can lead to an arbitrarily bad robustness. Handling this additional need for adaptivity in the second framework phase is a main technical contribution of this chapter.

5.3.2 Algorithmic Ideas

During the first phase, we generalize the classical witness set analysis. As in the previous chapter, we extend this concept by considering *strengthened* witness sets of three elements such that any feasible query set must contain at least two of them. Since we cannot always find strengthened witness sets based on structural properties alone (otherwise, there would be a 1.5-competitive algorithm for the problem without predictions), we identify such sets under the assumption that the predictions are correct. Even after identifying such edges, the algorithm needs to query them in a careful order: if the predictions are wrong, we lose the guarantee on the elements, and querying all of them might violate the robustness. In order to identify strengthened witness sets, we provide new, more global criteria to identify additional witness sets (of size two) beyond the ideas used by the witness set algorithm. During the first phase, we repeatedly query $\gamma - 2$ prediction mandatory edges together with a strengthened witness set, which ensures $(1 + \frac{1}{\gamma})$-consistency. We query the elements in a carefully selected order while adjusting for errors to ensure γ-robustness.

For the second phase, we observe that the predicted optimal solution of the remaining instance is a minimum vertex cover VC in a bipartite auxiliary graph representing the structure of *potential* witness pairs (edges of the input graph correspond to vertices of the auxiliary graph). For instances with this property, we aim for 1-consistency and 2-robustness; the best possible tradeoff for such instances. If the predictions are correct, each edge of the auxiliary graph is a witness pair. However, if a prediction error is observed when a vertex of VC is queried, the auxiliary graph changes. This means that some edges of the original auxiliary graph are not actually witness pairs. Indeed, we show that the size of a minimum vertex cover can increase and decrease and does not constitute a lower bound on $|\mathrm{OPT}|$. This is in contrast to the hypergraph orientation problem of the previous two chapters.

If we only aim for consistency and robustness, we can circumvent this problem by selecting a distinct matching partner $h(e) \notin VC$ for each $e \in VC$ applying *Kőnig-Egerváry*'s Theorem (duality of maximum matchings and minimum vertex covers in bipartite graphs, see e.g. [BLW86]). By querying the elements of VC in a carefully chosen order until a prediction error is observed for the first time, we can guarantee that $\{e, h(e)\}$ is a witness set for each $e \in VC$ that is already queried. In the case of an error, this allows us to extend the previously queried elements to disjoint witness pairs and guarantee 2-robustness. Then, we can switch to an arbitrary (prediction-oblivious) 2-competitive algorithm for the remaining queries.

If we additionally aim for an error-sensitive guarantee, however, handling the dynamic changes to the auxiliary graph, its minimum vertex cover VC and matching h requires substantial additional work. In particular, querying the partner $h(e)$ of each already queried $e \in VC$ in case of an error might be too expensive for the error-dependent guarantee. However, if we do not query these partners, the changed instance still depends on them, and if we charge against such a partner multiple times, we can lose the robustness. Our solution is based on an elaborate charging/counting scheme and involves:

1. keeping track of matching partners of already queried elements of VC;
2. updating the matching and VC using an augmenting path method to bound the number of elements that are charged against multiple times in relation to the prediction error;
3. and querying the partners of previously queried edges (and their new matching partners) as soon as they become endpoints of a newly matched edge, in order to prevent dependencies between the (only partially queried) witness sets of previously queried edges.

The error-sensitive algorithm achieves a competitive ratio of $1 + \frac{1}{\gamma} + \frac{5k_h}{|\mathrm{OPT}|}$, at the price of a slightly increased robustness of $\gamma + 1$ instead of γ.

5.4 Prediction Mandatory Edges and New Structural Results

In order to identify strengthened witness sets for the first framework phase, we need new criteria to identify witness sets and prediction mandatory edges. Recall that an edge $e \in E$ is *prediction mandatory* if every feasible query set contains e under the assumption that $\overline{w}_e = w_e$ for all $e \in E$. Note that in the context of adaptive algorithms, it might happen that an edge is not prediction mandatory for the initial instance but might become prediction mandatory

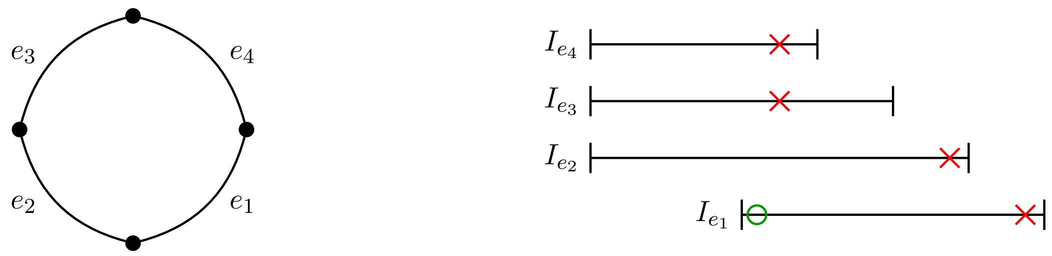


FIGURE 5.7: Example instance consisting of a simple cycle with uncertainty intervals and prediction weights as indicated by the red crosses. For the initial instance, no edge is prediction mandatory. After querying e_1 to reveal the precise weight as indicated by the green circle, e_2 becomes prediction mandatory.

given the precise weights revealed by already executed queries. To be more precise, assume that an algorithm already queried a subset Q of the edges. Then, there might be an edge that is not mandatory under the assumption that $\overline{w}_e = w_e$ for all $e \in E$, but is mandatory given the precise weights w_e of all $e \in Q$ under the assumption that the predicted weights of the not yet queried edges are correct, i.e., $\overline{w}_e = w_e$ for all $e \in E \setminus Q$. Thus, there is a difference between being prediction mandatory for the initially given instance and being prediction mandatory for the current instance after querying a subset $Q \subseteq E$.

See Figure 5.7 for an example consisting of a single cycle with four edges $e_1, \dots, e_4$. In the example, no edge from $\{e_2, e_3, e_4\}$ is initially prediction mandatory as querying just e_1 would prove e_1 to be maximal on cycle if the predicted weight $\overline{w}_1$ was correct. However, after querying e_1 and revealing the precise weight of e_1, the edge e_2 becomes predictions mandatory: Given the precise weight of e_1 and assuming that the predicted weights of e_2, e_3 and e_4 are correct, it is not possible to prove that the weight of e_2 is larger than $\overline{w}_{e_3}$ without querying e_2. Thus, e_2 became prediction mandatory .

If not stated otherwise, we use the term prediction mandatory in reference to being prediction mandatory for the current instance after executing all previous queries.

In this section, we derive criteria to identify prediction mandatory edges and connect the existence of edges that are prediction mandatory but not mandatory for the precise weights to the hop distance error. The latter part will be important when analyzing algorithms with error dependencies on the hop distance.

Furthermore, we give a characterization of instances without prediction mandatory edges. We will exploit this characterization in the second phase of our basic framework.

5.4.1 New Criteria to Identify Witness Sets and (Prediction) Mandatory Edges

We introduce new structural properties to identify witness sets. Existing algorithms for MST under uncertainty [Hof+08; MMS17] essentially follow the algorithms of Kruskal or Prim, and only identify witness sets in the cycle or cut that is currently under consideration. As an example for this, see the witness set algorithm (cf. Algorithm 14).

Let T_L be the lower limit tree of a preprocessed instance of the minimum spanning tree problem under explorable uncertainty with graph $G = (V, E)$. Furthermore, let $f_1, \dots, f_l$ denote the edges in $E \setminus T_L$ indexed by non-decreasing lower limit L_{f_i}. Lemma 5.2.10 shows how to identify a witness set on the cycle closed by f_{i+1} after an MST for graph $G_i = (V, T_L \cup \{f_1, \dots, f_i\})$ has already been determined. Known algorithms for MST under uncertainty build on this by iteratively resolving cycles closed by adding the edges $f_1, \dots, f_l$ one after the other (and analogously for cut-based algorithms). We design algorithms that query edges with a less local strategy; this requires to identify witness sets involving edges f_{i+1} without first verifying an MST for G_i. The following two lemmas provide new structural insights that are fundamental for our algorithms.

Lemma 5.4.1. *Given a preprocessed instance for the MST problem under explorable uncertainty with graph $G = (V, E)$, lower limit tree T_L, and the edges $\{f_1, \dots, f_l\} \subseteq E \setminus T_L$ indexed by non-decreasing lower limit. For all $i \in \{1, \dots, l\}$, let C_i denote the cycle in $T_L \cup \{f_i\}$. If f_i has a non-trivial uncertainty interval and there exists an $l_i \in C_i \setminus \{f_i\}$ with $I_{l_i} \cap I_{f_i} \neq \emptyset$ such that $l_i \notin C_j$ for all $j < i$, then $\{l_i, f_i\}$ is a witness set. Furthermore, if $w_{l_i} \in I_{f_i}$, then f_i is mandatory.*

Proof. Consider the set of edges X_i in the cut of G defined by the two connected components of $T_L \setminus \{l_i\}$. By assumption of the lemma, $l_i, f_i \in X_i$. However, $f_j \notin X_i$ for all $j < i$, as otherwise $l_i \in C_j$ would hold for an $f_j \in X_i$ with $j < i$, which contradicts the assumption.

Let $E_i = T_L \cup \{f_1, \dots, f_i\}$. Then, we can observe $X_i \cap E_i = \{f_i, l_i\}$. Let Q be any feasible query set. By Lemma 5.2.9, $Q_{i-1} = E_{i-1} \cap Q$ verifies an MST T_{i-1} for G_{i-1}. Consider the unique cycle C in $T_{i-1} \cup \{f_i\}$. By Lemma 5.2.10, f_i has the largest upper limit in C after querying $Q_{i-1} \setminus \{l_i\}$. Since $f_i \in X_i$, $f_i \in C$ and C is a cycle, there must be another edge in $X_i \setminus \{f_i\}$ that is part of C. We already observed $X_i \cap E_i = \{l_i, f_i\}$, so we have $l_i \in C$. Lemma 5.2.10 implies that $\{f_i, l_i\}$ is a witness set. If $w_{l_i} \in I_{f_i}$, then f_i must be queried to identify the maximal edge in C, so it follows that f_i mandatory. □

Although phrased based on cycles, the Lemma 5.4.1 actually gives us a criteria to identify witness sets and mandatory edges based on cuts (cf. the cut X_i in the proof). We continue with a similar lemma based on cycles. To prove this second lemma, we require the following additional auxiliary lemma.

Lemma 5.4.2. *Given a preprocessed instance for the MST problem under explorable uncertainty with graph $G = (V, E)$ and lower limit tree T_L. Let Q be a feasible query set that verifies an MST T^*. Consider any path $P \subseteq T_L$ between two endpoints a and b, and let $e \in P$ be the edge with the largest upper limit in P (before querying Q). If $e \notin Q$, then the unique path $\hat{P} \subseteq T^*$ from a to b is such that $e \in \hat{P}$ and e has the largest upper limit in $\hat{P}$ after Q has been queried.*

Proof. For each $i \in \{0, \dots, l\}$ let T_i^* be the MST for G_i as verified by $Q_i = Q \cap E_i$ and let $\hat{C}_i$ be the unique cycle in $T_{i-1}^* \cup \{f_i\}$. Then $T_i^* = T_{i-1}^* \cup \{f_i\} \setminus \{h_i\}$ holds where h_i is an edge of maximum precise weight in $\hat{C}_i$.

Let e be the edge of the lemma. Assume $e \notin Q$. We claim that there cannot be any $\hat{C}_i$ with $e \in \hat{C}_i$ such that Q_i verifies that an edge $e' \in \hat{C}_i$ with $U_{e'} \leq U_e$ is maximal in $\hat{C}_i$. Assume otherwise. If $e' \neq e$, Q_i would need to verify that $w_e \leq w_{e'}$ holds. Since $U_{e'} \leq U_e$, this can only be done by querying e, which is a contradiction to $e \notin Q$. If $e' = e$, then $\hat{C}_i$ still contains edge f_i and we have $e \neq f_i$ by assumption of the lemma. Since $T_L = T_U$, f_i has a larger lower limit than e. To verify that e is maximal in $\hat{C}_i$, Q_i needs to prove $w_e \geq w_{f_i} > L_{f_i}$. This can only be done by querying e, which is a contradiction to $e \notin Q$.

To finish the proof of the lemma, we show via induction on $i \in \{0, \dots, l\}$ that each T_i^* contains a path P_i^* from a to b with $e \in P_i^*$ such that e has the largest upper limit in P_i^* after Q_i has been queried.

Base case $i = 0$: Since $G_0 = (E, T_L)$ is a spanning tree, $T_0^* = T_L$ follows. Therefore $P_0^* = P$ is part of T_L and by assumption $e \in P$ has the largest upper limit in P_0^*.

Inductive step: By induction hypothesis, there is a path P_i^* from a to b in T_i^* with $e \in P_i^*$ such that e has the largest upper limit in P_i^* after querying Q_i. Consider cycle $\hat{C}_{i+1}$. If an edge $e' \in \hat{C}_{i+1} \setminus P_i^*$ is maximal in $\hat{C}_{i+1}$, then $T_{i+1}^* = T_i^* \cup \{f_{i+1}\} \setminus \{e'\}$ contains path P_i^*. Since e by assumption is not queried, e still has the largest upper limit on $P_i^* = P_{i+1}^*$ after querying Q_{i+1} and the statement follows.

Assume some $e' \in P_i^* \cap \hat{C}_{i+1}$ is maximal in $\hat{C}_{i+1}$, then $U_{e'} \leq U_e$ follows by induction hypothesis since e has the largest upper limit in P_i^*. We already observed that $\hat{C}_{i+1}$ then cannot contain e, as otherwise e would have been queried to verify that e' has maximum precise weight in $\hat{C}_{i+1}$. Consider $P' = \hat{C}_{i+1} \setminus P_i^*$. Since $e' \in P_i^*$ is maximal in $\hat{C}_{i+1}$, we can observe that $P' \subseteq T_{i+1}^*$ holds. It follows that path $P_{i+1}^* = P' \cup (P_i^* \setminus \hat{C}_{i+1})$ with $e \in P_{i+1}^*$ is part of T_{i+1}^*. As e is not queried, it still has a larger upper limit than all edges in P_i^*. Additionally, we can observe that after querying Q_{i+1} no $u \in P'$ can have an upper limit $U_u > U_{e'}$. If such an u would exist, querying Q_{i+1} would not verify that e' is maximal on $\hat{C}_{i+1}$, which contradicts the assumption. Using $U_e \geq U_{e'}$, we can conclude that e has the largest upper limit on P_{i+1}^* and the statement follows. □

Using this auxiliary lemma, we are ready to prove the next criterion to identify witness sets and (prediction) mandatory edges.

Lemma 5.4.3. *Given a preprocessed instance for the MST problem under explorable uncertainty with graph $G = (V, E)$, lower limit tree T_L, and the edges $\{f_1, \dots, f_l\} \subseteq E \setminus T_L$ indexed by non-decreasing lower limit. For all $i \in \{1, \dots, l\}$, let C_i denote the cycle in $T_L \cup \{f_i\}$. Consider cycle C_i with $i \in \{1, \dots, l\}$. If an edge $l_i \in C_i \setminus \{f_i\}$ has a non-trivial uncertainty interval such that $I_{l_i} \cap I_{f_i} \neq \emptyset$ and l_i has the largest upper limit in $C_i \setminus \{f_i\}$, then $\{f_i, l_i\}$ is a witness set. Furthermore, if $w_{f_i} \in I_{l_i}$, then l_i is mandatory.*

Proof. To prove the lemma, we have to show that each feasible query set contains at least one element of $\{f_i, l_i\}$. Let Q be an arbitrary feasible query set. By Lemma 5.2.9, $Q_{i-1} := Q \cap E_{i-1}$ is a feasible query set for $G_{i-1} = (V, E_{i-1})$ with $E_{i-1} = T_L \cup \{f_1, \dots, f_{i-1}\}$ and verifies some MST T_{i-1} for G_{i-1}. We show that $l_i \notin Q_{i-1}$ implies either $l_i \in Q$ or $f_i \in Q$.

Assume $l_i \notin Q_{i-1}$ and let C be the unique cycle in $T_{i-1} \cup \{f_i\}$. Since $T_L = T_U$, edge f_i has the largest upper limit in C after querying Q_{i-1}. While we only assume $T_L = T_U$ for the initially given instance, Lemma 5.2.10 implies that f_i still has the largest upper limit in C after querying Q_{i-1}.

If we show that $l_i \notin Q_{i-1}$ implies $l_i \in C$, we can apply Lemma 5.2.10 to derive that $\{f_i, l_i\}$ is a witness set for graph G_i, and thus either $f_i \in Q_i \subseteq Q$ or $l_i \in Q_i \subseteq Q$. For the remainder of the proof we show that $l_i \notin Q_{i-1}$ implies $l_i \in C$. Let a and b be the endpoints of f_i, then the path $P = C_i \setminus \{f_i\}$ from a to b is part of T_L and l_i has the largest upper limit in P by assumption. Using $l_i \notin Q_{i-1}$ we can apply the auxiliary Lemma 5.4.2 to conclude that there must be a path $\hat{P}$ from a to b in T_{i-1} such that l_i has the largest upper limit on $\hat{P}$ after querying Q_{i-1}. Therefore, $C = \hat{P} \cup \{f_i\}$ and it follows $l_i \in C$.

If $w_{f_i} \in I_{l_i}$ and $l_i \notin Q_{i-1}$, then l_i must be queried to identify the maximal edge on C, so l_i is mandatory.

□

5.4.2 Prediction Mandatory Free Instances

We say an instance of the MST problem under explorable uncertainty with predictions is *prediction mandatory free* if it contains no prediction mandatory edges. A key part of our algorithms (the entire first framework phase) is to transform instances into prediction mandatory free instances while maintaining a competitive ratio that allows us to achieve the optimal consistency and robustness tradeoff overall. In order to do so, we rely on the following characterization of prediction mandatory free instances (cf. Figure 5.8 for an illustration).

Lemma 5.4.4. *Given a preprocessed instance of the MST problem under explorable uncertainty with graph $G = (V, E)$, lower limit tree T_L, uncertainty intervals I_e and predicted weights $\overline{w}_e \in I_e$ for all $e \in E$. Let $f_1, \dots, f_l$ denote the edges in $E \setminus T_L$ index by non-decreasing lower limit and let C_i denote the unique cycle in $T_L \cup \{f_i\}$. The instance is prediction mandatory free if and only if $\overline{w}_{f_i} \geq U_e$ and $\overline{w}_e \leq L_{f_i}$ holds for each $e \in C_i \setminus \{f_i\}$ and each cycle C_i with $i \in \{1, \dots, l\}$.*

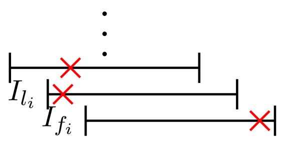


FIGURE 5.8: Intervals in a prediction mandatory free cycle, where f_i is the only edge on the cycle outside of the lower limit tree T_L. Predictions are indicated as red crosses.

Proof. For the first direction, assume $\overline{w}_{f_i} \geq U_e$ and $\overline{w}_e \leq L_{f_i}$ holds for each $e \in C_i \setminus \{f_i\}$ and each cycle C_i with $i \in \{1, \ldots, l\}$. Then each $f_i \in E \setminus T_L$ is predicted to be maximal on C_i and each $e \in T_L$ is predicted to be minimal in X_e, the cut between the components of $T_L \setminus \{e\}$. Assuming the predictions are correct, we observe that each vertex cover of the bipartite graph $\bar{G}$ is a feasible query set [EH14], where $\bar{G} = (\bar{V}, \bar{E})$ with $\bar{V} = \{e \in E \mid I_e \text{ is non-trivial}\}$ and $\bar{E} = \{\{f_i, e\} \subseteq \bar{V} \mid i \in \{1, \ldots, l\}, e \in C_i \setminus \{f_i\} \text{ and } I_e \cap I_{f_i} \neq \emptyset\}$. Since both $Q_1 := T_L$ and $Q_2 := E \setminus T_L$ are vertex covers for $\bar{G}$, Q_1 and Q_2 are both feasible query sets under the assumption that the predictions are correct. This implies that no element is part of every feasible solution because $Q_1 \cap Q_2 = \emptyset$. We can conclude that no element is prediction mandatory and the instance is prediction mandatory free.

Next, we show that instance G being prediction mandatory free implies that $\overline{w}_{f_i} \geq U_e$ and $\overline{w}_e \leq L_{f_i}$ holds for each $e \in C_i \setminus \{f_i\}$ and each cycle C_i with $i \in \{1, \ldots, l\}$; via contraposition. Assume there is a cycle C_i such that $\overline{w}_{f_i} \in I_e$ or $\overline{w}_e \in I_{f_i}$ for some $e \in C_i \setminus \{e\}$. Let C_i be such a cycle with the smallest index.

If $\overline{w}_{f_i} \in I_e$ for some $e \in C_i \setminus \{e\}$, then also $\overline{w}_{f_i} \in I_{l_i}$ for the edge l_i with the largest upper limit in $C_i \setminus \{f_i\}$. This implication holds because the instance is preprocessed and we have $T_L = T_U$. Under the assumption that the predictions are correct, Lemma 5.4.3 implies that l_i is mandatory and, thus, prediction mandatory. This means that G is not prediction mandatory free.

Assume $\overline{w}_e \in I_{f_i}$. We can observe that $e \notin C_j$ for each $j < i$. As the instance is preprocessed and the edges in $E \setminus T_L$ are ordered by non-decreasing lower limit, $\overline{w}_e \in I_{f_i}$ and $j < i$ would otherwise imply $\overline{w}_e \in I_{f_j}$. Since we assumed that C_i is the first cycle with this property, $e \in C_j$ leads to a contradiction. Under the assumption that the predictions are true, Lemma 5.4.1 implies that f_i is mandatory and, thus, prediction mandatory. It follows that G is not prediction mandatory free. □

Using this characterization of prediction mandatory free instances, we can observe that, once an instance is prediction mandatory free, it remains so even if we query further elements, as long as we maintain unique $T_L = T_U$, i.e., keep the instance preprocessed. The following lemma formalizes this observation.

Lemma 5.4.5. *Let G be a preprocessed and prediction mandatory free instance of the MST problem under explorable uncertainty with predictions and let G' be an instance with unique $T'_L = T'_U$ that is obtained from G by querying a set of edges Q, where T'_L and T'_U are the lower and upper limit trees of G'. Then, G' is prediction mandatory free.*

Proof. Let $G = (V, E)$ and $G' = (V', E')$ as well as $T_L = T_U$ and $T'_L = T'_U$ be as described in the lemma. We show that G being prediction mandatory free implies that G' is prediction mandatory free via proof by contradiction. To that end, assume that G' is not prediction mandatory free.

Let T'_L be the lower limit tree of G', let $f'_1, \ldots, f'_{l'}$ be the (non-trivial) edges in $E' \setminus T'_L$ ordered by non-decreasing lower limit, and let C'_i be the unique cycle in $T'_L \cup \{f'_i\}$. By assumption, $T'_L = T'_U$ holds and $T'_L = T'_U$ is unique. We can w.l.o.g. ignore trivial edges in $E' \setminus T'_L$ as those are maximal in a cycle and can be deleted and we can w.l.o.g. ignore trivial edges in T'_L as those can be contracted. Since G' is not prediction mandatory free, there must be some C'_i such that either $\overline{w}_e \in I_{f'_i}$ or $\overline{w}_{f'_i} \in I_e$ for some non-trivial $e \in C'_i \setminus \{f'_i\}$ (cf. Lemma 5.4.4).

Assume $e \notin T_L$. As e is part of $T'_L = T'_U$, Lemma 5.2.8 implies that e must have been queried and therefore is trivial, which is a contradiction. Thus, assume $e \in T_L$. We distinguish between the two cases $\overline{w}_e \in I_{f'_i}$ and $\overline{w}_{f'_i} \in I_e$ and separately show that both cases lead to a contradiction.

First, consider the case $\overline{w}_e \in I_{f'_i}$. As G is prediction mandatory free, the assumption implies $e \notin C_{f'_i}$ and $f_{i'} \notin X_e$, where $C_{f'_i}$ is the cycle in $T_L \cup \{f'_i\}$ and X_e is the cut between the two components of $T_L \setminus \{e\}$ in G. Note that $f'_i \notin T_L$ must hold by Lemma 5.2.8 because f'_i is not in T'_L and is non-trivial in both G and G'. Thus, $T_L \cup \{f'_i\}$ indeed contains a cycle $C_{f'_i}$.

Since C'_i is a cycle containing e, it must contain a second edge f from the cut X_e. This edge f closes a cycle C_f in $T_L \cup \{f\}$ with $e \in C_f$. As $f \in C'_i \setminus \{f'_i\} \subseteq T'_L$ but $f \notin T_L$, Lemma 5.2.8 implies that f was non-trivial in instance G but became trivial in G', i.e., was queried. By Lemma 5.4.4, the instance G being prediction mandatory free implies $\overline{w}_e \notin I_f$ where I_f denotes the uncertainty interval of f before querying it. The fact that $\overline{w}_e \in I_{f'_i}$ but $\overline{w}_e \notin I_f$ implies $L_{f'_i} < L_f$. So even before being queried, f had a larger lower limit than f'_i. By querying f, its lower limit can only increase. Thus, f has a larger lower limit in C'_i than f'_i. This is a contradiction to T'_L being the lower limit tree of the preprocessed instance G'.

Next, consider the case with $\overline{w}_{f'_i} \in I_e$. Remember that f'_i is non-trivial and $f'_i \notin T'_L = T'_U$. According to Lemma 5.2.8, this implies $f'_i \notin T_L = T_U$. Let $C_{f'_i}$ be the cycle in $T_L \cup \{f'_i\}$. Since G is prediction mandatory free, we have $\overline{w}_{f'_i} \notin I_{e'}$ for each $e' \in C_{f'_i} \setminus \{f'_i\}$, which implies $U_e > U_{e'}$. This means that the largest upper limit on the path between the two endpoints of f'_i in $T'_U = T'_L$ is strictly larger than the largest upper limit on the path between the two endpoints of f'_i in $T_U = T_L$. We argue that this cannot happen , which leads to contradiction to $e \in T_L$ and $\overline{w}_{f'_i} \in I_e$.

Let P be the path between the endpoints a and b of f'_i in T_U and let P' be the path between a and b in T'_U. Define U_P to be the largest upper limit on P. Observe that the upper limit of each edge can only decrease from T_U to T'_U since querying edges only decreases their upper limits. So each $e' \in P' \cap P$ cannot have a larger upper limit than U_P. It remains to argue that the upper limit of each $e' \in P' \setminus P$ cannot be larger than U_P. Consider the set $\mathcal{S}$ of maximal subpaths $S \subseteq P'$ such that only the endpoint vertices of S are also on path P. Each $e' \in P' \setminus P$ is part of such a subpath S. Let S be an arbitrary element of $\mathcal{S}$ that only contains edges of $P' \setminus P$, then there is a cycle $C \subseteq S \cup P$ with $S \subseteq C$. Assume $e' \in S$ has a strictly larger upper limit than U_P, then an element of S has the unique largest upper limit on C. It follows that subpath S cannot be part of any upper limit tree in the instance G'. This in turn means that path P' cannot be part of any upper limit tree of G', a contradiction to the assumption that P' is a path in T'_U. □

5.4.3 Relation Between Prediction Mandatory Edges and The Hop Distance

We establish a relation between the set $E_M \subseteq E$ of mandatory edges, the set $E_P \subseteq E$ of prediction mandatory edges, and the hop distance k_h.

Lemma 5.4.6. *Consider an instance of the MST problem under explorable uncertainty with graph $G = (V, E)$, uncertainty intervals I_e and with predicted weights $\overline{w}_e \in I_e$ for all $e \in E$. Let $E_M \subseteq E$ denote the set of mandatory edges and let $E_P \subseteq E$ denote the set of prediction mandatory edges. Each $e \in E_M \Delta E_P$ satisfies $k^-(e) \geq 1$. Consequently, $k_h \geq |E_M \Delta E_P|$.*

Note that the following proof is essentially an adjusted variant of the proof of Theorem 4.2.5 for the hypergraph orientation problem.

Proof. Consider an instance $G = (V, E)$ with uncertainty intervals $I_e = (L_e, U_e)$, precise weights w_e and predicted weights $\overline{w}_e$ for all $e \in E$. Let E_P and E_M be the mandatory queries with respect to the predicted and precise weights, respectively. We claim that, for every interval I_e of an edge $e \in E_P \Delta E_M$, there is an interval I_g of an edge g that lies on a cycle with e such that at least one of the following inequalities holds $w_g \leq L_e < \overline{w}_g$, $w_g < U_e \leq \overline{w}_g$, $\overline{w}_g \leq L_e < w_g$ or $\overline{w}_g < U_e \leq w_g$. This then implies $k^-(e) \geq 1$ for all $e \in E_M \Delta E_P$ and, thus, the lemma.

We continue by proving the claim. Consider an edge $e \in E_P \setminus E_M$. (The argumentation for edges in $E_M \setminus E_P$ is symmetric, with the roles of w and $\overline{w}$ exchanged.) As e is not in E_M, replacing all intervals I_g for $g \in E \setminus \{e\}$ by their precise weights yields an instance that is solved. This means that for edge e one of the following cases applies:

(a) e is known to be in the MST. Then there is a cut X_e containing edge e (namely, the cut between the two vertex sets obtained from the MST by removing the edge e) such that e is known to be a minimum weight edge in the cut, i.e., every other edge g in the cut satisfies $w_g \geq U_e$.

(b) e is known not to be in the MST. Then there is a cycle C_e in G (namely, the cycle that is closed when e is added to the MST) such that e is a maximum weight edge in C_e, i.e., every other edge g in the cycle satisfies $w_g \leq L_e$.

As e is in E_P, replacing all intervals I_g for $g \in E \setminus \{e\}$ by their predicted weights yields an instance Π that is not solved. Let T' be the minimum spanning tree of $G' = (V, E \setminus \{e\})$ for Π. Let C' be the cycle closed in T' by adding e, and let f be an edge with the largest predicted weight in $C' \setminus \{e\}$. Then there are only two possibilities for the minimum spanning tree of G for Π: Either T' is also a minimum spanning tree of G (if $\overline{w}_e \geq \overline{w}_f$), or the minimum spanning tree is $T' \cup \{e\} \setminus \{f\}$. As the instance by assumption is not solved, it must be the case that we cannot determine whether e is in the minimum spanning tree or not without querying e. If e satisfied case (a) with cut X_e above, then there must be an edge g in $X_e \setminus \{e\}$ with $\overline{w}_g < U_e$, because otherwise e would also have to be in the MST of G for Π, a contradiction. Thus, $\overline{w}_g < U_e \leq w_g$. If e satisfied case (b) with cycle C_e above, then there must be an edge g in $C_e \setminus \{e\}$ with $\overline{w}_g > L_e$, because otherwise e would also be excluded from the MST of G for Π, a contradiction. Thus, $w_g \leq L_e < \overline{w}_g$. In conclusion $k^-(e) \geq 1$, which establishes claim and lemma. □

The proof of the lemma states that $e \in E_P \Delta E_M$ implies $k^-(e) \geq 1$. We remark that for this implication to hold it is not important that $e \in E_P \Delta E_M$ holds for the set E_P of edges that are prediction mandatory for the initially given instance. If $e \in E_P \Delta E_M$ holds at some point during the execution of an adaptive query algorithm for the set of edges E_P that are prediction mandatory for the current instance at that point of the execution, then $k^-(e) \geq 1$.

5.5 Making Instances Prediction Mandatory Free

In this section, we consider the implementation of the first framework phase for the MST problem under explorable uncertainty with predictions. Remember that the first phase considers problem instances that are *not* prediction mandatory free and queries sets of edges with strong local guarantees until the instance becomes prediction mandatory free. The goal is to transform the instance into a prediction mandatory free one while still being in range to match the consistency and robustness tradeoff lower bound (cf. Theorem 5.2.1) with a linear error dependency on k_h. Formally, we will give an algorithm that achieves the following theorem.

Theorem 5.5.1. *For every integer $\gamma \geq 2$. Given an instance of the MST problem under explorable uncertainty with predictions, there is an algorithm that queries a set of edges* ALG *such that the problem instance becomes prediction mandatory free after querying* ALG. *Furthermore, the algorithm satisfies* $|\mathrm{ALG}| \leq \min\{(1+\frac{1}{\gamma}) \cdot (|(\mathrm{ALG} \cup D) \cap \mathrm{OPT}| + k^+(\mathrm{ALG}) + k^-(\mathrm{ALG})), \gamma \cdot |(\mathrm{ALG} \cup D) \cap \mathrm{OPT}| + \gamma - 2\}$ *for an optimal query set* OPT *of the complete instance and a set $D \subseteq E \setminus \mathrm{ALG}$ of unqueried edges that do not occur in the remaining instance* after *executing the algorithm as they can w.l.o.g. be deleted or contracted after querying* ALG.

The set D of the theorem are edges that, even without being queried by the algorithm, are proven to be maximal in a cycle or minimal in a cut. Thus, they can be deleted or contracted w.l.o.g. and do not exist in the instance remaining *after* executing the algorithm anymore. This is an important property as it means that the remaining instance is independent of D and ALG (as all elements of ALG are already queried). Since the theorem compares $|\mathrm{ALG}|$ with $|(\mathrm{ALG} \cup D) \cap \mathrm{OPT}|$ instead of just $|\mathrm{OPT}|$, this will allow us to combine the given guarantee of the theorem with the guarantees of dedicated algorithms for prediction mandatory free instances. We will do so in Sections 5.6 and 5.7. However, we have to be careful with the additive term $\gamma - 2$, but we will see that we can charge this term against the improved robustness of our algorithms for prediction mandatory free instances.

For the remainder of this section, we design and analyze an algorithm that satisfies Theorem 5.5.1.

5.5.1 Algorithm and Overview of the Algorithmic Ideas

We present Algorithm 15 which transforms arbitrary instances into prediction mandatory free instances while maintaining the guarantees of Theorem 5.5.1. Within the algorithm, X_l for an $l \in T_L$ refers to the set of edges in the cut between the two connected components of $T_L \setminus \{l\}$. Furthermore, as usual, C_i refers to the cycle in $T_L \cup \{f_i\}$.

We start by giving a summary of the ideas behind the algorithm before we move on to the formal analysis.

In each iteration the algorithm starts by querying edges that are prediction mandatory for the current instance. The set of prediction mandatory elements can be computed in polynomial time as shown in [EH14]. We use their algorithm as a blackbox to compute such edges.

The algorithm sequentially queries such edges until either $\gamma - 2$ prediction mandatory edges have been queried or no more exist (cf. Line 1). After each query, the algorithm ensures unique $T_L = T_U$ by using Lemma 5.2.6. Note that the set of prediction mandatory elements with respect to the current instance can change when elements are queried, and therefore we query the elements sequentially. By Lemma 5.4.6, each of the at most $\gamma - 2$ elements is either mandatory or contributes one to the hop distance. This means that such queries will never violate the consistency and error-dependent part of the desired guarantees, i.e., the first term of the minimum in Theorem 5.5.1. They might however violate the γ-robustness.

In order to still guarantee robustness and in line with the basic framework described in Section 5.3, the algorithm afterwards tries to identify a *strengthened witness set*, i.e., a set of three edges such that every feasible solution must query at least two of them. The algorithm finds such sets under the assumption that the predictions are correct and queries them in a careful order such that it can decide after two queries whether the selected edges are indeed a strengthened witness set, independent of whether the predictions of not yet queried edges are correct or not. If the edges are indeed a strengthened witness set, the algorithm queries also the third edge, which guarantees that the $\gamma + 1$ queries of the iteration locally achieve the desired guarantee. Otherwise, we show that the two already queried edges form a witness set and that we can charge the missing third query against a distinct error.

Algorithm 15: Algorithm to make instances prediction mandatory free

```
   Input: Uncertainty graph $G = (V, E)$ and predictions $\overline{w}_e$ for each $e \in E$
1  Ensure unique $T_L = T_U$. Sequentially query prediction mandatory elements (while
   ensuring unique $T_L = T_U$) until either $\gamma - 2$ prediction mandatory elements are
   queried or the instance is prediction mandatory free;
2  Let $T_L$ be the lower limit tree and $f_1, \dots, f_l$ be the (non-trivial) edges in $E \setminus T_L$
     ordered by non-decreasing lower limit;
3  foreach $C_i$ with $i = 1$ to $l$ do
4      if $C_i$ is not prediction mandatory free then
5          Let $l_i$ be an edge with largest upper limit in $C_i \setminus \{f_i\}$;
6          if $\overline{w}_{f_i} \in I_{l_i}$ and $\overline{w}_{l_i} \in I_{f_i}$ then Query $\{f_i, l_i\}$ ; // cf. Fig. 5.9(a)
7          else if $\overline{w}_{f_i} \in I_{l_i}$ then // cf. Fig. 5.9(b)
8              if $\exists l'_i \in C_i \setminus \{f_i, l_i\}$ with $I_{l'_i} \cap I_{f_i} \neq \emptyset$ then
9                  $l'_i \leftarrow$ edge in $C_i \setminus \{f_i, l_i\}$ with the largest upper limit;
10                 Query $\{f_i, l_i\}$, query $l'_i$ only if $w_{f_i} \in I_{l_i}$ and $w_{l_i} \notin I_{f_j}$ for all $j$ with
                     $l_i \in C_j$;
11             else Query $l_i$, query $f_i$ only if $w_{l_i} \in I_{f_i}$ ;
12         else if $\overline{w}_{l'_i} \in I_{f_i}$ for some $l'_i \in C_i$ then // cf. Fig. 5.9(c)
13             $l'_i \leftarrow$ edge with the largest upper limit in $\{l \in C_i \setminus \{f_i\} \mid \overline{w}_l \in I_{f_i}\}$;
14             if $\exists f_j \in X_{l'_i} \setminus \{f_i, l'_i\}$ with $I_{f_j} \cap I_{l'_i} \neq \emptyset$ then
15                 $f_j \leftarrow$ edge in $X_{l'_i} \setminus \{f_i, l'_i\}$ with the smallest lower limit;
16                 Query $\{f_i, l'_i\}$, query $f_j$ only if $w_{l'_i} \in I_{f_j}$ and $w_{f_i} \notin I_e$ for all
                     $e \in C_i \setminus \{f_i\}$;
17             else Query $f_i$, query $l'_i$ only if $w_{f_i} \in I_{l'_i}$ ;
18         Restart at Line 1;
```

In order to identify such edges, the algorithm considers the lower limit tree T_L and the edges $f_1, \dots, f_l$ in $E \setminus T_L$ ordered by non-decreasing lower limit. As before, we denote by C_i the cycle in $T_L \cup \{f_i\}$. The algorithm iterates through $i \in \{1, \dots, l\}$ until the current cycle C_i either has $\overline{w}_e \in I_{f_i}$ or $\overline{w}_{f_i} \in I_e$ for some $e \in C_i \setminus \{f_i\}$. We call such cycles *not prediction mandatory free* and all other cycles prediction mandatory free. Note that if the current instance is still not prediction mandatory free, it must contain at least one not prediction mandatory free cycle by Lemma 5.4.4. Thus, the algorithm must find such a cycle in every iteration except the final one.

If it finds such a cycle C_i, closed by f_i, it queries edges on the cycle and possibly future cycles as follows. Let l_i denote the edge in $C_i \setminus \{f_i\}$ with largest upper limit. As C_i is not prediction mandatory free, the configuration of l_i and f_i and their predicted weights must be one of those illustrated in Fig. 5.9(a)–(c). In each case, the algorithm queries one to three edges while ensuring 1.5-consistency and 2-robustness *locally* for those queries, as well as a more refined guarantee depending on prediction errors that will be needed when the algorithm is used as part of our error-sensitive algorithm in Section 5.7. Line 6 handles the case of Fig. 5.9(a), Lines 7–11 the case of Fig. 5.9(b), and Lines 12–17 the case of Fig. 5.9(c). We now briefly sketch the analysis of these three cases and defer the formal statements and proof details to Lemmas 5.5.2, 5.5.3 and 5.5.4 in the consequent section.

In Line 6, we can show that $\{l_i, f_i\}$ is a witness set (giving local 2-robustness) and either $|\mathrm{OPT} \cap \{l_i, f_i\}| = 2$ (giving local 1-consistency) or $k^+(\{l_i, f_i\}) \geq 1$. For the queries made in Line 10, we can show that if the algorithm queries three edges, OPT must query at least

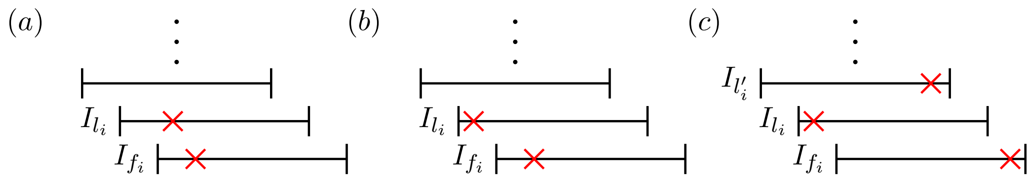


FIGURE 5.9: with predictions indicated as red crosses. (a) Intervals in a prediction mandatory free cycle. (b)–(d) Intervals in a cycle that is not prediction mandatory free.

two of them (ensuring 1.5-consistency and 1.5-robustness). If the algorithm queries only the two edges f_i and l_i, they form a witness set (2-robustness) and $k^+(\{f_i, l_i\}) \geq 1$. For Line 11, we show that if the algorithm queries only l_i, then $\{f_i, l_i\}$ is a witness set and f_i can be deleted from the instance without querying it (giving 1-consistency and 1-robustness). If the algorithm queries f_i and l_i, they form a witness set and $k^+(\{f_i, l_i\}) \geq 1$. The guarantees we can prove for queries made in Lines 12–17 are analogous, except that the edge l'_i can be contracted instead of deleted if the algorithm queries only f_i in Line 17. After processing C_i in this way, the algorithm restarts. The algorithm terminates when all C_i are prediction mandatory free, which holds at the latest when all edges in E have been queried.

We sketch the proof of Theorem 5.5.1. Elements queried in Line 1 to ensure unique $T_L = T_U$ are mandatory by Lemma 5.2.6 and can be ignored in the analysis. Each iteration of the algorithm queries a set P_i of up to $\gamma - 2$ prediction mandatory edges e in Line 1, each of which is mandatory or satisfies $k^-(e) \geq 1$ by Lemma 5.4.6, showing $|P_i| \leq |P \cap \mathrm{OPT}| + k^-(P_i)$. In the last iteration, these are the only queries, and they contribute the additive term $\gamma - 2$ to the bound. In each iteration prior to the last, a set W_i of at most 3 queries is made in Line 6, 10, 11, 16 or 17. These cases are covered by the following possibilities: (1) The set W_i consists of three edges, and OPT contains at least two of them, giving $|W_i| \leq 1.5 \cdot |\mathrm{OPT} \cap W_i|$ and $|W_i| \leq |\mathrm{OPT} \cap W_i| + 1$; (2) The set W_i consists of two edges, and either OPT contains both or OPT contains one of them and $k^+(W_i) \geq 1$, giving $|W_i| \leq 2 \cdot |\mathrm{OPT} \cap W_i|$ and $|W_i| \leq |\mathrm{OPT} \cap W_i| + k^+(W_i)$; (3) The set W_i contains a single edge e and we can delete or contract another edge $g(e)$ such that $\{e, g(e)\}$ is a witness set, giving $|W_i| \leq |\mathrm{OPT} \cap \{e, g(e)\}|$. Combining these bounds over all iterations yields Theorem 5.5.1, where the union of the edges $g(e)$ constitutes the set D.

To conclude we observe that all edges queried by Algorithm 15 can, w.l.o.g., be contracted or deleted: Since all queried edges are trivial and we ensure unique $T_L = T_U$, we can observe that each queried $e \in T_L = T_U$ is minimal on a cut and can be contracted, and each queried $e \notin T_L = T_U$ is maximal on a cycle and can be deleted. This allows us to treat the instance after the execution of the algorithm independently of all previous queries, which will turn out to be a useful property when combining the algorithm with dedicated algorithms for prediction mandatory free instances in Sections 5.6 and 5.7.

5.5.2 Formal Analysis of the Algorithm

We continue by turning the ideas of the previous section into a formal analysis of the algorithm. The main part of the analysis considers the three different configurations of not prediction mandatory free cycles (cf. Figure 5.9) and proves the local guarantees of the queries that the algorithm executes for the respective case. To this end, we prove the following three lemmas stating that, in each iteration of Algorithm 15, the queries made in one excution of Lines 6–17 locally satisfy 1.5-consistency and 2-robustness. All lemmas consider a cycle C_i such that all C_j with $j < i$ are prediction mandatory free, l_i is the edge with the largest upper limit in $C_i \setminus \{f_i\}$ and predictions are as indicated in Figure 5.9(a) (Lemma 5.5.2), Figure 5.9(b) (Lemma 5.5.3), and Figure 5.9(c) (Lemma 5.5.4).

Recall the following notation: As usual, we use T_L to refer to the lower limit tree of the current instance and $f_1, \dots, f_l$ to refer to the edges in $E \setminus T_L$ ordered by non-decreasing lower limits. By $G_i = (V, E_i)$ we denote the subgraph with the edges $E_i = T_L \cup \{f_1, \dots, f_i\}$ and for a query set Q we use Q_i to refer to the subset $Q \cap E_i$.

Lemma 5.5.2. *Let $\{f_i, l_i\}$ denote a pair of edges queried in Line 6 of Algorithm 15, then $\{f_i, l_i\}$ is a witness set, and either both edges are mandatory or $k^+(\{f_i, l_i\}) \geq 1$.*

Proof. By assumption, all C_j with $j < i$ are prediction mandatory free. We claim that this implies $l_i \notin C_j$ for all $j < i$. Assume, for the sake of contradiction, that there is a C_j with $j < i$ and $l_i \in C_j$. Then, $T_L = T_U$ and $j < i$ imply that f_i and f_j have larger upper and lower limits than l_i and, since $L_{f_i} \geq L_{f_j}$, it follows $I_{l_i} \cap I_{f_i} \subseteq I_{l_i} \cap I_{f_j}$. Thus, $\overline{w}_{l_i} \in I_{f_i}$ implies $\overline{w}_{l_i} \in I_{f_j}$, which contradicts cycle C_j being prediction mandatory free. This allows us to apply Lemma 5.4.1 and conclude that $\{f_i, l_i\}$ is a witness set.

Consider any feasible query set Q for the current instance G, then Q_{i-1} verifies the MST T_{i-1} for graph G_{i-1} and Q needs to identify the maximal edge on the unique cycle C in $T_{i-1} \cup \{f_i\}$ since Q_i has to be feasible for G_i by Lemma 5.2.9. Following the argumentation of Lemma 5.4.1, we can observe $l_i, f_i \in C$. Since we assume $T_L = T_U$, we can also observe that f_i has the largest upper limit in C after querying Q_{i-1}. By Lemma 5.4.2, l_i has the largest upper limit in $C \setminus \{f_i\}$ after querying $Q_{i-1} \setminus \{l_i\}$.

If $w_{l_i} \in I_{f_i}$, then f_i is mandatory according to Lemma 5.4.1. Otherwise, $w_{l_i} \leq L_{f_i} < \overline{w}_{l_i}$ and $k^+(l_i) \geq 1$. If $w_{f_i} \in I_{l_i}$, then l_i is mandatory according to Lemma 5.4.3. Otherwise, $w_{f_i} \geq U_{l_i} > \overline{w}_{f_i}$ and $k^+(f_i) \geq 1$. In conclusion, either $\{f_i, l_i\} \subseteq Q$ for any feasible query set Q or $k^+(\{f_i, l_i\}) \geq 1$. □

Lemma 5.5.3. *Let $\{f_i, l_i, l_i'\}$ denote the edges of Line 10. If the algorithm queries all three edges, then $|\{f_i, l_i, l_i'\} \cap \mathrm{OPT}| \geq 2$. Otherwise, $k^+(\{f_i, l_i\}) > 0$ and $|\{f_i, l_i\} \cap \mathrm{OPT}| \geq 1$.*

Let $\{l_i, f_i\}$ denote the edges of Line 11. If the algorithm queries only l_i, then $|\{f_i, l_i\} \cap \mathrm{OPT}| \geq 1$ and f_i can be deleted from the instance without querying it. Otherwise, $k^+(\{f_i, l_i\}) > 0$ and $|\{f_i, l_i\} \cap \mathrm{OPT}| \geq 1$.

Proof. By assumption, all C_j with $j < i$ are prediction mandatory free. According to Lemma 5.4.3, $\{f_i, l_i\}$ is a witness set.

Consider the first part of the lemma, i.e., the edges $\{f_i, l_i, l_i'\}$ of Line 10. Assume first that the algorithm queries all three edges. By Line 10, this means that $w_{f_i} \in I_{l_i}$ and $w_{l_i} \notin I_{f_j}$ for each j with $l_i \in C_j$. According to Lemma 5.4.3, $w_{f_i} \in I_{l_i}$ implies that l_i is mandatory. Consider the relaxed instance where l_i is already queried, then $w_{l_i} \notin I_{f_j}$ for each j with $l_i \in C_j$ implies that l_i is minimal in X_{l_i} (the cut between the two connected components of $T_L \setminus \{l_i\}$) and that the lower limit tree does not change by querying l_i. This implies that l_i' is the edge with the largest upper limit in $C_i \setminus \{f_i\}$ in the relaxed instance and, by Lemma 5.4.3, $\{f_i, l_i'\}$ is a witness set. Thus, $|\{f_i, l_i, l_i'\} \cap \mathrm{OPT}| \geq 2$.

Next, assume that the algorithm queries only $\{f_i, l_i\}$. Then, either $w_{f_i} \notin I_{l_i}$ or $w_{l_i} \in I_{f_j}$ for some j with $l_i \in C_j$. Note that if $w_{l_i} \in I_{f_j}$ holds for some j with $l_i \in C_j$, then, by the ordering of the edges in $E \setminus T_L$, the statement holds for some $j \leq i$. If $w_{f_i} \notin I_{l_i}$, then $w_{f_i} \geq U_{l_i} > \overline{w}_{f_i}$ and $k^+(f_i) \geq 1$ follows. If $w_{l_i} \in I_{f_j}$, then $\overline{w}_{l_i} \leq L_{f_j} < w_{l_i}$ and $k^+(l_i) \geq 1$ follows. Note that $\overline{w}_{l_i} \leq L_{f_j}$ holds for $i = j$ because the if-statement in Line 6 failed and for $j < i$ it holds as the corresponding cycle C_j is prediction mandatory free by assumption. In conclusion, if either $w_{f_i} \notin I_{l_i}$ or $w_{l_i} \in I_{f_j}$ for some j with $l_i \in C_j$, then $k^+(f_i, l_i) \geq 1$.

Consider the second part of the lemma, i.e., the elements $\{f_i, l_i\}$ of Line 11. Assume first that the algorithm queries only l_i. Clearly, $|\{f_i, l_i\} \cap \mathrm{OPT}| \geq 1$ as $\{f_i, l_i\}$ is a witness set. Consider the cycle C_i. As f_i is not queried, it follows $w_{l_i} \notin I_{f_i}$. Furthermore, by definition of Line 11, it holds $I_{l_i'} \cap I_{f_i} = \emptyset$ for all $l_i' \in C_i \setminus \{f_i, l_i\}$. Thus, f_i is proven to be uniquely maximal on cycle C_i and, therefore, can be deleted from the instance. Next, assume that the algorithm queries l_i and f_i. Clearly, $|\{f_i, l_i\} \cap \mathrm{OPT}| \geq 1$ still holds. As f_i is queried, we have $w_{l_i} > L_{f_i} \geq \overline{w}_{l_i}$ and, therefore, $k^+(\{f_i, l_i\}) \geq 1$. □

The next lemma can be shown using analogous arguments.

Lemma 5.5.4. *Let $\{f_i, l'_i, f_j\}$ denote the edges of Line 16. If the algorithm queries all three edges, then $|\{f_i, l'_i, f_j\} \cap \mathrm{OPT}| \geq 2$. Otherwise, $k^+(\{f_i, l'_i\}) > 0$ and $|\{f_i, l'_i\} \cap \mathrm{OPT}| \geq 1$.*

Let $\{l'_i, f_i\}$ denote the edges of Line 17. If the algorithm queries only f_i, then $|\{f_i, l'_i\} \cap \mathrm{OPT}| \geq 1$ and l'_i can be contracted from the instance without querying it. Otherwise, $k^+(\{f_i, l'_i\}) > 0$ and $|\{f_i, l'_i\} \cap \mathrm{OPT}| \geq 1$.

Proof. By assumption, all C_j with $j < i$ are prediction mandatory free. We claim that this implies $l'_i \notin C_j$ for all $j < i$. Assume, for the sake of contradiction, that there is a C_j with $j < i$ and $l'_i \in C_j$. Then, $T_L = T_U$ and $j < i$ imply that f_i and f_j have larger upper and lower limits than l'_i and, since $L_{f_i} \geq L_{f_j}$, it follows $I_{l'_i} \cap I_{f_i} \subseteq I_{l'_i} \cap I_{f_j}$. Thus, $\overline{w}_{l'_i} \in I_{f_i}$ implies $\overline{w}_{l'_i} \in I_{f_j}$, which contradicts C_j being prediction mandatory free. This allows us to apply Lemma 5.4.1 to conclude that $\{f_i, l'_i\}$ is a witness set.

Consider the first part of the lemma, i.e., the edges $\{f_i, l'_i, f_j\}$ of Line 16. Assume first that the algorithm queries all three elements. By Line 16, this means that $w_{l'_i} \in I_{f_i}$ and $w_{f_i} \notin I_e$ for each $e \in C_i$. According to Lemma 5.4.1, $w_{l'_i} \in I_{f_i}$ implies that f_i is mandatory. Consider the relaxed instance where f_i is already queried, then $w_{f_i} \notin I_e$ for each $e \in C_i$ implies that f_i is maximal in cycle C_i and that the lower limit tree does not change by querying f_i. It follows that f_j is the edge with the smallest index (observe that $j > i$ must hold by the argument at the beginning of the proof) and $l'_i \in C_j$ in the relaxed instance and, by Lemma 5.4.1, $\{f_j, l'_i\}$ is a witness set. Thus, $|\{f_i, l'_i, f_j\} \cap \mathrm{OPT}| \geq 2$.

Next, assume that the algorithm queries only $\{f_i, l'_i\}$. Then, either $w_{l'_i} \notin I_{f_i}$ or $w_{f_i} \in I_e$ for some $e \in C_i$. If $w_{l'_i} \notin I_{f_i}$, then $w_{l'_i} \leq L_{f_i} < \overline{w}_{l'_i}$ and $k^+(l'_i) \geq 1$ follows. If $w_{f_i} \in I_e$, then $\overline{w}_{f_i} \geq U_e > w_{f_i}$ and $k^+(f_i) \geq 1$ follows. Therefore, $w_{l'_i} \notin I_{f_i}$ or $w_{f_i} \in I_e$ for some $e \in C_i$ implies $k^+(\{f_i, l'_i\}) \geq 1$.

Consider the second part of the lemma, i.e., the elements $\{f_i, l'_i\}$ of Line 17. Assume first that the algorithm queries only f_i. Clearly, $|\{f_i, l'_i\} \cap \mathrm{OPT}| \geq 1$ as $\{f_i, l'_i\}$ is a witness set. Consider the cut $X_{l'_i}$ between the two connected components of $T_L \setminus \{l'_i\}$. As l'_i is not queried, it follows $w_{f_i} \notin I_{l'_i}$. Furthermore, by definition of Line 17, it holds $I_{f_j} \cap I_{l'_i} = \emptyset$ for all $f_j \in X_{l'_i} \setminus \{f_i, l'_i\}$. Thus, l'_i is proven to be uniquely minimal in cut $X_{l'_i}$ and, therefore, can be contracted from the instance. Next, assume that the algorithm queries f_i and l'_i. Clearly, $|\{f_i, l'_i\} \cap \mathrm{OPT}| \geq 1$ still holds. As l'_i is queried, we have $w_{f_i} < U_{l'_i} \leq \overline{w}_{f_i}$ and, therefore, $k^+(\{f_i, l'_i\}) \geq 1$. □

Using these three lemmas, we are finally ready to give a full proof of Theorem 5.5.1.

Proof of Theorem 5.5.1. Since Algorithm 15 only terminates if Line 4 determines each C_i to be prediction mandatory free, the instance after executing the algorithm is prediction mandatory free by definition and Lemma 5.4.4. All elements queried in Line 1 to ensure unique $T_L = T_U$ are mandatory by Lemma 5.2.6 and never worsen the performance guarantee.

During the course of the proof, we use *iteration* to refer to one execution of the algorithm starting from the first line until it restarts or terminates. Since the last iteration does not query in Lines 6, 10, 11, 16 and 17, the last iteration queries at most $\gamma - 2$ edges. All those edges are prediction mandatory and they cause the additive part in the second term of the minimum. In the following, we consider iterations i, that are not the last iteration. Each such iteration i queries a set P_i of $\gamma - 2$ prediction mandatory elements in Line 1 and a set W_i in Line 6, 10, 11, 16 or 17. By Lemma 5.4.6, each $e \in P_i$ is either mandatory or $k^-(e) \geq 1$.

Consider an arbitrary iteration i (that is not the last one). Then, $|P_i| \leq |\mathrm{OPT} \cap P_i| + k^-(P_i)$ holds by by the argument above.

Assume W_i was queried in Line 6, 10 or 16, then Lemmas 5.5.2 to 5.5.4 imply $|W_i| \leq \frac{3}{2} \cdot (|W_i \cap \mathrm{OPT}| + k^+(W_i))$. Thus, $|W_i \cup P_i| \leq (1+\frac{1}{\gamma}) \cdot (|\mathrm{OPT} \cap (W_i \cup P_i)| + k^+(W_i) + k^-(P_i))$. At the same time, the lemmas imply that either $|W_i| = 2$ and W_i is a witness set or $|W_i| = 3$ and each feasible query set must contain at last two members of W_i. In both cases we have $|W_i \cup P_i| \leq \gamma \cdot |\mathrm{OPT} \cap (W_i \cup P_i)|$.

Putting it together, we get $|W_i \cup P_i| \leq \min\{(1+\frac{1}{\gamma}) \cdot (|\mathrm{OPT} \cap (W_i \cup P_i)| + k^+(W_i) + k^-(P_i)), \gamma \cdot |\mathrm{OPT} \cap (W_i \cup P_i)|\}$. Let $\mathcal{K}_1$ denote the union of all sets $W_i \cup P_i$ that satisfy the assumption and let J_1 denote the index set of the corresponding sets. Note that the sets $W_i \cup P_i$ for different indices i are pairwise disjoint as the algorithm never queries edges multiple times. Thus, we get

$$\begin{aligned}
|\mathcal{K}_1| &= \sum_{i \in J_1} |W_i \cup P_i| \\
&\leq \sum_{i \in J_1} \min\{(1+\frac{1}{\gamma})(|\mathrm{OPT} \cap (W_i \cup P_i)| + k^+(W_i) + k^-(P_i)), \gamma|\mathrm{OPT} \cap (W_i \cup P_i)|\} \\
&\leq \min\{(1+\frac{1}{\gamma})(|\mathrm{OPT} \cap \mathcal{K}_1| + k^+(\mathcal{K}_1) + k^-(\mathcal{K}_1)), \gamma|\mathrm{OPT} \cap \mathcal{K}_1|\}.
\end{aligned}$$

Assume now that W_i was queried in Line 11 or 17. Let $e_i \in W_i$ denote the element that is queried first, and let $g(e_i)$ denote the element that either is queried second or deleted/contracted after the first query. Since each $g(e_i)$ is either queried or deleted/-contracted, no such edge is considered more than once. Lemmas 5.5.3 and 5.5.4 imply $|W_i| \leq (|(W_i \cup \{g(e_i)\}) \cap \mathrm{OPT}| + k^+(W_i))$ and $|W_i| \leq 2 \cdot |(W_i \cup \{g(e_i)\}) \cap \mathrm{OPT}|$. Thus, $|W_i \cup P_i| \leq (1+\frac{1}{\gamma}) \cdot (|\mathrm{OPT} \cap (W_i \cup \{g(e_i)\} \cup P_i)| + k^+(W_i) + k^-(P_i))$ and $|W_i| \leq \gamma \cdot |(W_i \cup \{g(e_i)\} \cup P_i) \cap \mathrm{OPT}|$. Let $\mathcal{K}_2$ denote the union of all sets $W_i \cup P_i$ that satisfy the assumption, let J_2 denote the index set of the corresponding sets, and let $\mathcal{G}$ denote the union of all corresponding $\{g(e_i)\}$:

$$\begin{aligned}
&|\mathcal{K}_2| \\
&= \sum_{W_i \in J_2} |W_i \cup P_i| \\
&\leq \sum_{W_i \in J_2} \min\{|\mathrm{OPT} \cap (W_i \cup \{g(e_i)\} \cup P_i)| + k^+(W_i \cup P_i), \gamma|\mathrm{OPT} \cap (W_i \cup \{g(e_i)\} \cup P_i)|\} \\
&\leq \min\{|\mathrm{OPT} \cap (\mathcal{K}_2 \cup \mathcal{G})| + k^+(\mathcal{K}_2) + k^-(\mathcal{K}_2), \gamma|\mathrm{OPT} \cap (\mathcal{K}_2 \cup \mathcal{G})|\}.
\end{aligned}$$

Let $\mathcal{K}_3$ denote the queries of the last iteration. Recall that D is the set of edges in $E \setminus \mathrm{ALG}$ that can be deleted/contracted by the algorithm *before* the final iteration. By definition, $\mathcal{G} \subseteq D$. Furthermore, note that $|\mathcal{K}_3| \leq \gamma - 2$ and $|\mathcal{K}_3| \leq |\mathrm{OPT} \cap \mathcal{K}_3| + k^-(\mathcal{K}_3)$. The latter holds by Lemma 5.4.6 as the edges in $\mathcal{K}_3$ are prediction mandatory for the current instance when they are queried.

Since $\mathcal{K}_1$, $\mathcal{K}_2 \cup \mathcal{G}$ and $\mathcal{K}_3$ are pairwise disjoint, we can conclude

$$\begin{aligned}
|\mathrm{ALG}_1| = |\mathcal{K}_1| + |\mathcal{K}_2| + |\mathcal{K}_3| \\
&\leq \min\{(1+\frac{1}{\gamma}) \cdot (|(\mathrm{ALG} \cup D) \cap \mathrm{OPT}| + k^+(\mathrm{ALG}) + k^-(\mathrm{ALG})) \\
&\quad , \gamma \cdot |(\mathrm{ALG} \cup D) \cap \mathrm{OPT}| + \gamma - 2\}.
\end{aligned}$$

$\square$

5.6 Optimal Consistency and Robustness Tradeoff

In this section, we design an algorithm that achieves the optimal tradeoff between consistency and robustness, matching the lower bound of Theorem 5.2.1. To this end, we prove the following theorem.

Theorem 5.6.1. *For every integer $\gamma \geq 2$, there exists a $(1 + \frac{1}{\gamma})$-consistent and γ-robust algorithm for the MST problem under explorable uncertainty with predictions.*

To show this result, we design an algorithm for prediction mandatory free instances with unique $T_L = T_U$, which corresponds to the second framework phase as described in Section 5.3. We run it after Algorithm 15, which obtains such special instances from arbitrary instances with the query guarantee of Theorem 5.5.1. The combination of both algorithms will achieve the optimal tradeoff.

Note that we only prove consistency and robustness for the algorithm of this section. We extend the algorithm to also achieve an error-dependency in Section 5.7.

5.6.1 Optimal Tradeoff for Prediction Mandatory Free Instances

We give a 1-consistent and 2-robust algorithm for prediction mandatory instances. Note that this tradeoff is optimal for such instances as it achieves optimality if the predictions are correct and otherwise matches the adversarial lower bound of two (cf. Theorem 2.2.3).

Theorem 5.6.2. *There exists a* 1*-consistent and* 2*-robust algorithm for preprocessed prediction mandatory free instances of the MST problem under explorable uncertainty with predictions.*

Consider a preprocessed prediction mandatory free instance with graph G and lower limit tree T_L. Recall the following notation: Let $f_1, \dots, f_l$ again denote the edges in $E \setminus T_L$ ordered by non-decreasing lower limits and let C_i with $i \in \{1, \dots, l\}$ denote the cycle in $T_L \cup \{f_i\}$. For an edge $l \in T_L$, let X_l denote the set of edges in the cut between the two connected components of $T_L \setminus \{l\}$.

In a prediction mandatory free instance, each $f_i \in E \setminus T_L$ is predicted to be maximal on cycle C_i, and each $l \in T_L$ is predicted to be minimal in X_l. This is a direct consequence of the characterization of such instances given in Lemma 5.4.4.

If these predictions are correct, then T_L is an MST and the optimal query set is a minimum vertex cover in a bipartite graph $\bar{G} = (\bar{V}, \bar{E})$ with $\bar{V} = E$ (excluding trivial edges) and $\bar{E} = \{\{f_i, e\} \subseteq \bar{V} \mid i \in \{1, \dots, l\}, e \in C_i \setminus \{f_i\}$ and $I_e \cap I_{f_i} \neq \emptyset\}$ [EH14; MMS17]. We refer to $\bar{G}$ as the *vertex cover instance*. The fact that the (predicted) optimal query set is a minimum vertex cover of the vertex cover instance was shown in [EH14], but we also discuss later in this section why this is the case.

As a consequence of this fact, just querying a minimum vertex cover of the vertex cover instance guarantees optimality if the predictions are correct and, thus, 1-consistency. Recall that for the hypergraph orientation problem of the previous chapter, the size of such a minimum vertex cover was also a lower bound on the size of the optimal query set independent of whether the predictions are correct or not. This property essentially allowed us to non-adaptively query the complete vertex cover and still guarantee 2-robustness. For the minimum spanning tree problem however, this property does not hold. The reason for this is that if a query reveals that an f_i is not maximal on C_i or an $l \in T_L$ is not minimal in X_l, then the vertex cover instance changes. Such a change can drastically increase and decrease the size of the minimum vertex cover.

To see this, consider the example instance of Figure 5.10. The figure shows a prediction mandatory free instance with $T_L = \{l_1, \dots, l_{n/2}\}$ and $E \setminus T_L = \{f_1, \dots, f_{n/2}\}$. Given the graph and uncertainty intervals of the figure, the corresponding vertex cover instance $\bar{G}$ is the

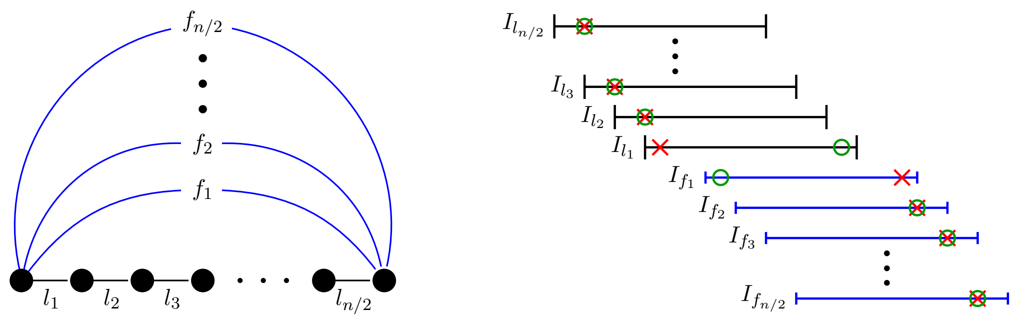


FIGURE 5.10: Graph $G = (V, E)$ (left) and uncertainty intervals of G (right). The black edges $(l_1, \dots, l_{n/2})$ form the lower limit tree T_L and the blue edges $(f_1, \dots, f_{n/2})$ are the edges outside of T_L. Circles illustrate precise weights and crosses illustrate the predicted weights.

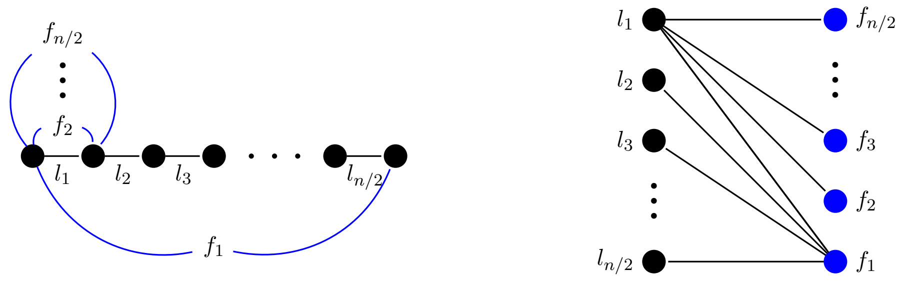


FIGURE 5.11: Uncertainty graph $G = (V, E)$ (left) and corresponding vertex cover instance $\bar{G}$ (right). The black edges $(l_1, \dots, l_{n/2})$ form the lower limit tree T_L.

complete bipartite graph with the partitions T_L and $E \setminus T_L$. Thus, both T_L and $E \setminus T_L$ are minimum vertex covers of size $n/2$. However, if the predictions are wrong and the precise weights are as depicted in the figure, then querying only l_1 and f_1 solves the instance. This show that the size of the initial minimum vertex cover is no lower bound on $|\mathrm{OPT}|$ and that changes in the vertex cover instance can decrease the size of the minimum vertex cover.

The example of Figure 5.11 shows that the inverse is true as well, i.e., that changes in the vertex cover instance can increase the size of the minimum vertex cover. The figure shows an instance with graph $G = (V, E)$ and the corresponding vertex cover instance $\bar{G}$ when using the uncertainty intervals of the previous Figure 5.10. The instance is prediction mandatory free and the minimum vertex cover of the vertex cover instance is $\{f_1, l_1\}$. However, for the precise weights of Figure 5.10, querying $\{f_1, l_1\}$ proves that f_1 is part of the MST but l_1 is not. After querying $\{f_1, l_1\}$, the new lower limit tree is the path $f_1, l_{n/2}, \dots, l_3, l_2, l_1$ and all edges outside of the lower limit tree connect the endpoints of the path. The resulting graph has the same form as the one in Figure 5.10 and the vertex cover instance is the complete bipartite graph. So, the size of the new minimum vertex cover is $(n/2) - 1$ (as l_1 and f_1 have already been queried), which shows that the size of the minimum vertex cover can increase.

These two examples illustrate that a 2-robust algorithm cannot afford to non-adaptively query the complete vertex cover of the vertex cover instance, which is a major difference to the algorithms for the hypergraph orientation problem of the previous chapter. Instead, the algorithm has to be more adaptive.

Let VC denote a minimum cardinality vertex cover of the vertex cover instance. The idea of our algorithm (cf. Algorithm 16) is to sequentially query each $e \in VC$ and charge for querying e by a distinct non-queried element $h(e)$ such that $\{e, h(e)\}$ is a witness set. Querying exactly one element per distinct witness set implies optimality. To identify $h(e)$ for each element $e \in VC$, we use the fact that *Kőnig-Egerváry*'s Theorem (cf., e.g., [BLW86]) on the duality between minimum vertex covers and maximum matchings in bipartite graphs implies that there is a matching h that maps each $e \in VC$ to a distinct $e' \notin VC$. While the sets $\{e, h(e)\}$ with $e \in VC$ in general are not witness sets, querying VC in a specific order until the vertex cover instance changes (because of a prediction error) guarantees that $\{e, h(e)\}$ is a witness set for each already queried e. The algorithm queries in this order until it detects a wrong prediction or solves the problem. If it does not find a prediction error, then it just queries VC and, thus, is 1-consistent. Otherwise, it finds a wrong prediction and therefore only has to guarantee 2-competitiveness. In that case, it queries the partner $h(e)$ of each already queried edge e. Since all those $\{e, h(e)\}$ are witness sets, querying them never worsens the competitive ratio below 2. Afterwards, the algorithm can just solve the remaining instance by using a 2-competitive algorithm, e.g., the witness set algorithm [MMS17; Hof+08] of Section 5.2.3.

The key part of this idea is to specify an order for querying VC that guarantees that $\{e, h(e)\}$ is a witness set for each $e \in VC$ that is queried before a prediction error is detected. The following lemma specifies this order and shows that it satisfies the property.

Lemma 5.6.3. *Let $l'_1, \dots, l'_k$ be the edges in $VC \cap T_L$ ordered by non-increasing upper limit and let d be such that the precise weight of each l'_i with $i < d$ is minimal in cut $X_{l'_i}$ between the components of $T_L \setminus \{l'_i\}$, then $\{l'_i, h(l'_i)\}$ is a witness set for each $i \leq d$. Let $f'_1, \dots, f'_g$ be the edges in $VC \setminus T_L$ ordered by non-decreasing lower limit and let b be such that the precise weight of each f'_i with $i < b$ is maximal in the cycle $C_{f'_i}$ of $T_L \cup \{f'_i\}$, then $\{f'_i, h(f'_i)\}$ is a witness set for each $i \leq b$.*

Proof. We separately prove the first and second statement of the lemma.

Proof of the first statement. Consider an arbitrary l'_i and $h(l'_i)$ with $i \leq d$. By definition of h, the edge $h(l'_i)$ is not part of the lower limit tree. Consider $C_{h(l'_i)}$, i.e., the cycle in $T_L \cup \{h(l'_i)\}$. We claim that $C_{h(l'_i)}$ only contains $h(l'_i)$ and edges in $\{l'_1, \dots, l'_k\}$ (and possibly irrelevant edges with intervals that do not intersect $I_{h(l'_i)}$). To see this, recall that $l'_i \in VC$ by definition of h implies $h(l'_i) \notin VC$. For VC to be a vertex cover, each $e \in C_{h(l'_i)} \setminus \{h(l'_i)\}$ must either be in VC or have an uncertainty interval that does not intersect $h(l'_i)$.

Consider the relaxed instance where the precise weight for each l'_j with $j < d$ and $j \neq i$ is already known. By assumption each such l'_j is minimal in its cut $X_{l'_j}$. Thus, we can w.l.o.g. contract each such edge. This means that in the relaxed instance l'_i has the largest upper limit in $C_{h(l'_i)} \setminus \{h(l'_i)\}$. According to Lemma 5.4.3, $\{l'_i, h(l'_i)\}$ is a witness set.

Proof of the second statement. Consider an arbitrary f'_i and $h(f'_i)$ with $i \leq b$. By definition of h, the edge $h(f'_i)$ is part of the lower limit tree. Let X_i be the cut between the two components of $T_L \setminus \{h(f'_i)\}$, then we claim that X_i only contains $h(f'_i)$ and edges in $\{f'_1, \dots, f'_g\}$ (and possibly irrelevant edges with uncertainty intervals that do not intersect $I_{h(f'_i)}$). To see this, assume an $f_j \in \{f_1, \dots, f_l\}$ with $f_j \notin VC$ and $I_{f_j} \cap I_{h(f'_i)} \neq \emptyset$ was part of X_i. Since $f_j \notin VC$, each edge in $C_j \setminus \{f_j\}$ must either be part of VC or have an uncertainty interval that does not intersect $I_{h(f'_i)}$, as otherwise VC would not be a vertex cover for the vertex cover instance. But if f_j is in cut X_i, then C_j must contain $h(f'_i)$. By definition, we have $h(f'_i) \notin VC$, which is a contradiction to VC being a vertex cover as $\{f_j, h(f'_i)\}$ would not be covered. We can conclude that X_i only contains $h(f'_i)$ and edges in $\{f'_1, \dots, f'_g\}$.

Consider the relaxed instance where the precise weight for each f'_j with $j < b$ and $j \neq i$ is already known. By assumption each such f'_j is maximal in its cycle $C_{f'_j}$. Thus, we can w.l.o.g. delete each such edge. This means that f'_i has the smallest lower limit in $X_i \setminus \{h(f'_i)\}$ in the relaxed instance, which allows us to apply Lemma 5.4.1 to conclude that $\{f'_i, h(f'_i)\}$ is a witness set. □

Using this lemma, it is not hard to prove Theorem 5.6.2.

Algorithm 16: 1-consistent and 2-robust algorithm for prediction mandatory free instances.

```
  Input: Prediction mandatory free graph $G = (V, E)$ with unique $T_L = T_U$.
1 Compute maximum matching $h$ and minimum vertex cover $VC$ for $\bar{G}$;
2 Set $W = \emptyset$, and let $f'_1, \dots, f'_g$ and $l'_1, \dots, l'_k$ be as described in Lemma 5.6.3;
3 for $e$ chosen sequentially from the ordered list $f'_1, \dots, f'_g, l'_1, \dots, l'_k$ do
4     Query $e$ and add $h(e)$ to $W$;
5     if $k^+(e) \neq 0$ then
6         Query set $W$ solve the instance with a 2-competitive algorithm;
```

Proof of Theorem 5.6.2. We first show 1-consistency. Assume that all predictions are correct, then VC is an optimal query set and $k^+(e) = 0$ holds for all $e \in E$. It follows that Line 6 never executes queries and the algorithm queries exactly VC. This implies 1-consistency.

Next, we prove the 2-robustness. If the algorithm never queries in Line 6, then the consistency analysis implies 1-robustness. Suppose Line 6 executes queries. Let Q_1 denote the set of edges that are queried before the queries of Line 6 and let $Q_2 = \{h(e) \mid e \in Q_1\}$. Then Q_2 corresponds to the set W as queried in Line 6. By Lemma 5.6.3, each $\{e, h(e)\}$ with $e \in Q_1$ is a witness set. Further, the sets $\{e, h(e)\}$ are pairwise disjoint. Thus, $|Q_1 \cup Q_2| \leq 2 \cdot |\mathrm{OPT} \cap (Q_1 \cup Q_2)|$. Apart from $Q_1 \cup Q_2$, the algorithm queries a set Q_3 in Line 6 to solve the remaining instance with a 2-competitive algorithm. So, $|Q_3| \leq 2 \cdot |\mathrm{OPT} \setminus (Q_1 \cup Q_2)|$ and, adding up the inequalities, $|\mathrm{ALG}| \leq 2 \cdot |\mathrm{OPT}|$. □

5.6.2 Optimal Tradeoff for General Instances

We show that the algorithm that first executes Algorithm 15 and then Algorithm 16 satisfies the guarantees of Theorem 5.6.1, i.e., achieves the optimal consistency and robustness tradeoff. In order to do so, we carefully combine Theorems 5.5.1 and 5.6.2.

Proof of Theorem 5.6.1. Let $\mathrm{ALG} = \mathrm{ALG}_1 \cup \mathrm{ALG}_2$ be the query set queried by the algorithm, where ALG_1 and ALG_2 are the queries of Algorithm 15 and Algorithm 16, respectively. Let $P \subseteq \mathrm{ALG}_1$ denote the edges queried in the last iteration of Algorithm 15. Furthermore, let D denote the set of edges in $E \setminus \mathrm{ALG}_1$ that can be deleted or contracted during the execution of Algorithm 15. This implies $D \cap \mathrm{ALG}_2 = \emptyset$.

Assume first that $\mathrm{ALG}_2 = \emptyset$. Then, querying ALG_1 solves the problem. Theorem 5.5.1 directly implies $(1 + \frac{1}{\gamma})$-consistency. However, due to the additive term of $\gamma - 2$ in the second term of the minimum, the theorem does not directly imply γ-robustness. Recall that the additive term is caused exactly by the queries to P. As the algorithm executes queries in the last iteration, it follows that the instance is not solved at the beginning of the iteration (a solved instance is prediction mandatory free and would lead to direct termination). Thus, $|\mathrm{OPT} \setminus ((\mathrm{ALG}_1 \cup D) \setminus P)| \geq 1$ and $|P| \leq (\gamma - 2) \cdot |\mathrm{OPT} \setminus ((\mathrm{ALG}_1 \cup D) \setminus P)|$. Ignoring the additive term caused by P, Theorem 5.5.1 implies $|\mathrm{ALG}_1 \setminus P| \leq \gamma \cdot |\mathrm{OPT} \cap ((\mathrm{ALG}_1 \cup D) \setminus P)|$. As $|\mathrm{ALG}| = |\mathrm{ALG}_1 \setminus P| + |P|$, adding up the inequalities implies $|\mathrm{ALG}| \leq \gamma \cdot |\mathrm{OPT}|$.

Now, assume that $\mathrm{ALG}_2 \neq \emptyset$. Let $\mathrm{OPT} = \mathrm{OPT}_1 \cup \mathrm{OPT}_2$ be an optimal query set with $\mathrm{OPT}_1 = \mathrm{OPT} \cap (\mathrm{ALG}_1 \cup D)$ and $\mathrm{OPT}_2 = \mathrm{OPT} \setminus (\mathrm{ALG}_1 \cup D)$. Theorem 5.5.1 implies $|\mathrm{ALG}_1 \setminus P| \leq \gamma \cdot |\mathrm{OPT}_1 \setminus P|$ (since the additive term in Theorem 5.5.1 is caused by P).

By Theorem 5.6.2, $\mathrm{ALG}_2 \neq \emptyset$ implies $\mathrm{OPT}_2 \neq \emptyset$. Thus, $|\mathrm{OPT}_2| \geq 1$ and $|P| \leq (\gamma - 2) \cdot |\mathrm{OPT}_2|$. Furthermore, Theorem 5.6.2 also implies $|\mathrm{ALG}_2| \leq 2 \cdot |\mathrm{OPT}_2|$ and, thus, $|\mathrm{ALG}_2 \cup P| \leq \gamma \cdot |\mathrm{OPT}_2|$ Adding up the inequalities for $|\mathrm{ALG}_1 \setminus P|$ and $|\mathrm{ALG}_2 \cup P|$, we get $|\mathrm{ALG}| \leq \gamma \cdot |\mathrm{OPT}|$ and, therefore, γ-robustness.

In terms of consistency, Theorem 5.5.1 implies $|\mathrm{ALG}_1| \leq (1+\frac{1}{\gamma}) \cdot |\mathrm{OPT}_1|$ if all predictions are correct and Theorem 5.6.1 implies $|\mathrm{ALG}_2| = |\mathrm{OPT}_2|$ if all predictions are correct. Together, the inequalities imply $(1+\frac{1}{\gamma})$-consistency. □

5.7 An Error-Sensitive Algorithm

In this section, we extend the algorithm of Section 5.6 to obtain error sensitivity. To that end, we show the following theorem.

Theorem 5.7.1. *For every integer $\gamma \geq 2$, there exists a $\min\{1+\frac{1}{\gamma}+\frac{5\cdot k_h}{|\mathrm{OPT}|}, \gamma+1\}$-competitive algorithm for the MST problem under explorable uncertainty with predictions.*

The guarantee of the theorem does not quite match the tradeoff lower bound of Theorem 5.2.1, as the robustness of the algorithm is slightly worse. We remark that the precise robustness of our algorithm is $\max\{3, \gamma + \frac{1}{|\mathrm{OPT}|}\}$, which might be smaller than $\gamma+1$ but still does not match the lower bound. It remains open whether an algorithm with linear error-dependency on k_h that matches the tradeoff lower bound is possible.

To achieve the guarantee of the theorem, we again design a dedicated algorithm for prediction mandatory free instances and use it in combination with Algorithm 15, which transforms general instances into prediction mandatory free ones.

5.7.1 Error-Sensitive Algorithm for Prediction Mandatory Free Instances

We give an algorithm for prediction mandatory free instances that asymptotically matches the error-dependent guarantee of Theorem 5.2.3 at the cost of a slightly worse robustness. The following theorem formalizes the precise competitive ratio of the algorithm.

Theorem 5.7.2. *There exists a $\min\{1+\frac{5\cdot k_h}{|\mathrm{OPT}|}, 3\}$-competitive algorithm for preprocessed and prediction mandatory free instances of the MST problem under explorable uncertainty with predictions.*

We first describe the ideas behind the algorithm that satisfies the theorem and in particular describe the similarities and differences to Algorithm 16 of the previous section. Afterwards, we move on to formally analyze the algorithm.

Algorithmic ideas. We follow the same basic strategy as Algorithm 16, the algorithm for prediction mandatory free instances without error-dependency. The main difference is that Algorithm 16 just executes a 2-competitive algorithm for the problem without predictions once it detects an error. This is sufficient to achieve the optimal tradeoff as we, if an error occurs, only have to guarantee 2-competitiveness. To obtain an error-sensitive guarantee however, we have to ensure both, $|\mathrm{ALG}| \leq 3 \cdot |\mathrm{OPT}|$ *and* $|\mathrm{ALG}| \leq |\mathrm{OPT}| + 5 \cdot k_h$, even if errors occur. We cannot guarantee this by using a 2-competitive algorithm as a black box whenever we detect an error. Furthermore, we might also not be able to afford queries to the complete set W (Algorithm 16, Line 6) in the case of an error as this might violate $|\mathrm{ALG}| \leq |\mathrm{OPT}| + 5 \cdot k_h$. Thus, we need a different algorithm to achieve error-dependency.

We adjust the algorithm to query elements of $f'_1, \dots, f'_g$ and $l'_1, \dots, l'_k$ as described in Lemma 5.6.3 not only until an error occurs but until the vertex cover instance changes. That is, until some edge f that at the beginning of the iteration is not part of T_L becomes part of the lower limit tree, or some edge l that at the beginning of the iteration is part of T_L is not part of the lower limit tree anymore. We again use the set W, to store all matching partner $h(e)$ of queried edges e. Once the instance changes, we recompute both, the bipartite graph $\bar{G}$ as well as the matching h and minimum vertex cover VC for $\bar{G}$. Instead of querying the complete

Algorithm 17: Error-sensitive algorithm for prediction mandatory free instances of the MST problem under explorable uncertainty

```
Input: Preprocessed and prediction mandatory free instance of the MST problem
       under explorable uncertainty with graph $G = (V, E)$ and lower limit tree $T_L$
1  Compute maximum matching $h$ and minimum vertex cover $VC$ for $\bar{G}$ and set $W = \emptyset$;
2  Let $f'_1, \dots, f'_g$ and $l'_1, \dots, l'_k$ be as described in Lemma 5.6.3 for $VC$ and $h$;
3  $L \leftarrow T_L$, $N \leftarrow E \setminus T_L$;
4  for $e$ chosen sequentially from the ordered list $f'_1, \dots, f'_g, l'_1, \dots, l'_k$ do
5      If $e$ is non-trivial, i.e., has not been queried yet, query $e$ and add $h(e)$ to $W$;
6      Ensure unique $T_L = T_U$. For each query $e'$, if $\exists a$ s.t. $\{e', a\} \in h$, query $a$ ;
7      Let $\bar{G}' = (\bar{V}', \bar{E}')$ be the vertex cover instance for the current instance;
8      if Some $e' \in L$ is not in $T_L$ or some $e' \in N$ is in $T_L$ then
9          repeat
10             Let $\bar{G} = \bar{G}'$ and $\bar{h} = \{\{e', e''\} \in h \mid \{e', e''\} \in \bar{E}'\}$;
11             Compute $h$ and $VC$ by completing $\bar{h}$ with an augmenting path algorithm;
12             Query $R = (VC \cup h(VC)) \cap (W \cup \{e' \mid \exists e \in W$ with $\{e, e'\} \in h\})$ ;
13             Ensure unique $T_L = T_U$. For each query $e'$, if $\exists a$ s.t. $\{e', a\} \in h$, query $a$;
14             Let $\bar{G}' = (\bar{V}', \bar{E}')$ be the vertex cover instance for the current instance;
15         until $R = \emptyset$;
16         Restart at Line 2. In particular, do not reset $W$;
```

set W as in Algorithm 16, we only query the elements of W that occur in the recomputed matching, as well as the new matching partners of those elements. And instead of switching to a 2-competitive algorithm, we restart the algorithm with the recomputed matching and vertex cover.

Algorithm 17 formalizes this approach. In the algorithm, T_L and T_U always refer to the current lower and upper limit trees after all previous queries. Furthermore, h denotes a matching that matches each $e \in VC$ to a distinct $h(e) \notin VC$; we use the notation $\{e, e'\} \in h$ to indicate that h matches e and e'. For a subset $U \subseteq VC$ let $h(U) = \{h(e) \mid e \in U\}$. For technical reasons, the algorithm does not recompute an arbitrary matching h but follows the approach of Lines 10 and 11: Whenever the vertex cover instance changes from $\bar{G}$ to $\bar{G}'$, the algorithm considers the partial matching $\bar{h}$ consisting of all elements that are part of the matching for $\bar{G}$ and still form edges for $\bar{G}'$. Based on this partial matching $\bar{h}$, the algorithm then computes a new matching for $\bar{G}'$ by using a standard augmenting path algorithm. After that, the algorithm queries all elements e of set W that became part of the new matching as well as their new matching partners $h(e)$. The idea behind computing the new matching in this particular way is that an arbitrary maximum matching h might contain too many elements of W, which would lead to too many additional queries.

The final difference to the previous Algorithm 16 is that whenever the new algorithm queries mandatory edges e' to ensure unique $T_L = T_U$ in Lines 6 and 13, it also queries the current matching partner $h(e')$ of e' if e' is part of the current matching. Since each such queried edge e' is mandatory but not prediction mandatory, it not only must be part of every optimal query set but also incurs an error by Lemma 5.4.6. Thus, the algorithm can afford to query both e' and $h(e')$ without violating the target guarantee.

Formal analysis. We continue by formally analyzing the algorithm. Let ALG denote the queries of Algorithm 17 on a preprocessed and prediction mandatory free instance and let OPT denote a fixed optimal query set. We show Theorem 5.7.2 by proving $|\mathrm{ALG}| \leq |\mathrm{OPT}| + 5 \cdot k_h$ and $|\mathrm{ALG}| \leq 3 \cdot |\mathrm{OPT}|$.

Before we show the two inequalities, we state some key observations about the algorithm. We argue that an element e' can never be part of a partial matching $\bar{h}$ in an execution of Line 10 *after* it was already added to set W. Recall that the vertex cover instances only contain non-trivial elements. Thus, if an element e is queried in Line 5 and the current partner $e' = h(e)$ is added to set W, then the vertex cover instance at the next execution of Line 10 does not contain the edge $\{e, e'\}$ and, therefore, e' is not part of the partial matching $\bar{h}$ of that line. As long as e' is not added to the matching by Line 11, it, by definition, can never be part of a partial matching $\bar{h}$ in an execution of Line 10. As soon as the element e' is added to the matching in some execution of Line 11, it is queried in the following execution of Line 12. Therefore, e' can also not be part of a partial matching $\bar{h}$ in an execution of Line 10 after it is added to the matching again. This leads to the following observation.

Observation 5.7.3. *An element e' can never be part of a partial matching $\bar{h}$ in an execution of Line 10* after *it is added to set W. Once e' is added to the matching again in an execution of Line 11, it is queried directly afterwards in Line 12, and cannot occur in Line 5 anymore.*

We first analyze all queries that are *not* executed in Line 12. Let $Q_1 \subseteq \mathrm{ALG}$ denote the queries of Line 5, i.e., Q_1 contains the "regular" queries that are executed as part of the vertex cover in the order of Lemma 5.6.3. For each $e \in Q_1$ let $h^*(e)$ be the matching partner of e *at the time it was queried*, and let $h^*(Q_1) = \bigcup_{e \in Q_1} \{h^*(e)\}$. Finally, let Q_2 denote the queries of Lines 6 and 13 to elements of $h^*(Q_1)$, and let Q_3 denote the remaining queries of those lines. With the following lemma we show that the queries of $Q_1 \cup Q_2 \cup Q_3$ satisfy the target guarantees.

Lemma 5.7.4. $|Q_1 \cup Q_3 \cup h^*(Q_1)| \leq 2 \cdot |\mathrm{OPT} \cap (Q_1 \cup Q_3 \cup h^*(Q_1))|$ *and* $|Q_1 \cup Q_2 \cup Q_3| \leq |\mathrm{OPT} \cap (Q_1 \cup Q_3 \cup h^*(Q_1))| + k^-(Q_2 \cup Q_3)$.

Proof. First, consider Q_1 and $h^*(Q_1)$. By Lemma 5.4.5, the instance is prediction mandatory free at the beginning of each restart of the algorithm. By Lemma 5.6.3, each $\{e, h^*(e)\}$ with $e \in Q_1$ is a witness set. We claim that all such $\{e, h^*(e)\}$ are pairwise disjoint, which implies $|Q_1 \cup h^*(Q_1)| \leq 2 \cdot |\mathrm{OPT} \cap (Q_1 \cup h^*(Q_1))|$. If the sets were not pairwise disjoint, an element of $\{e, h^*(e)\}$ must occur a second time in Line 5 after e is queried and $h^*(e)$ is added to W. Thus, either e or $h^*(e)$ must become part of a recomputed matching in line 10. Edge e will not be part of such matching as it has already been queried. By Observation 5.7.3, edge $h^*(e)$ will also never be part of a such a matching. Thus, the sets $\{e, h^*(e)\}$ with $e \in Q_1$ are pairwise disjoint and we get $|Q_1 \cup h^*(Q_1)| \leq 2 \cdot |\mathrm{OPT} \cap (Q_1 \cup h^*(Q_1))|$.

Consider an $e \in Q_2 \subseteq h^*(Q_1)$ and let $e' \in Q_1$ with $h^*(e') = e$. Since $e' \in Q_1$, it was queried in Line 5. Observe that e must have been queried after e', as otherwise either e' would not have been queried in Line 5 (but together with e in Line 6 or 13), or e would not have been the matching partner of e' when it was queried; both contradict $e' \in Q_1$ and $h^*(e') = e$. This and Observation 5.7.3 imply that, at the time e is queried, its current matching partner is either the trivial e' or it has no current partner. So, e must have been queried in Line 6 or 13 because it was mandatory and not as the matching partner of a mandatory edge. Thus, each query of Q_2 is mandatory but, by Lemma 5.4.5, not prediction mandatory at the beginning of the iteration in which it is queried. Therefore, Lemma 5.4.6 implies that all mandatory elements e of Q_2 have $k^-(e) \geq 1$. Thus, $k^-(Q_2) \geq |Q_2|$. Together with the observation that all $\{e, h^*(e)\}$ are pairwise disjoint witness sets, this implies $|Q_1 \cup Q_2| \leq |\mathrm{OPT} \cap (Q_1 \cap h^*(Q_1))| + k^-(Q_2)$.

By the argument above, no element of Q_3 was queried as the matching partner to an element of $Q_2 \cup Q_1$. Thus, by Lemma 5.2.6 and the definition of the algorithm, at least half the elements of Q_3 are mandatory, and we have $|Q_3| \leq 2 \cdot |\mathrm{OPT} \cap Q_3|$, which implies $|Q_1 \cup Q_3 \cup h^*(Q_1)| \leq 2 \cdot |\mathrm{OPT} \cap (Q_1 \cup Q_3 \cup h^*(Q_1))|$.

Since at least half the elements of Q_3 are mandatory, we have $\frac{1}{2}|Q_3| \leq |\text{OPT} \cap Q_3|$. By the same argument as for Q_2, all mandatory elements e of Q_3 have $k^-(e) \geq 1$. Thus, $k^-(Q_3) \geq \frac{1}{2} \cdot |Q_3|$. Combining $k^-(Q_3) \geq \frac{1}{2} \cdot |Q_3|$ and $\frac{1}{2}|Q_3| \leq |\text{OPT} \cap Q_3|$ implies $|Q_3| \leq |\text{OPT} \cap Q_3| + k^-(Q_3)$. Together with the inequality for $|Q_1 \cup Q_2|$, we get $|Q_1 \cup Q_2 \cup Q_3| \leq |\text{OPT} \cap (Q_1 \cup Q_3 \cup h^*(Q_1))| + k^-(Q_2 \cup Q_3)$. □

Note that the first part of Lemma 5.7.4, i.e., $|Q_1 \cup Q_3 \cup h^*(Q_1)| \leq 2 \cdot |\text{OPT} \cap (Q_1 \cup Q_3 \cup h^*(Q_1))|$, captures not only all queries outside of Line 12 but also all queries of Line 12 to elements of set $W = h^*(Q_1)$. Thus, to prove $|\text{ALG}| \leq 3 \cdot |\text{OPT}|$, it remains to analyze the remaining queries of Line 12 that do not go to elements of set W. Let Q_4' denote the set of such queries. Observe that Line 12 only executes queries to members of W and their recomputed matching partners. By Observation 5.7.3, each element of W is considered at most once in Line 12. This implies $|Q_4'| \leq |W|$. Since $|W| \leq |\text{OPT}|$ (as shown in the proof of the previous lemma), we can conclude the following lemma.

Lemma 5.7.5. $|\text{ALG}| \leq 3 \cdot |\text{OPT}|$.

Next, we show $|\text{ALG}| \leq |\text{OPT}| + 5 \cdot k_h$. Lemma 5.7.4 implies $|Q_1 \cup Q_2 \cup Q_3| \leq |\text{OPT} \cap (Q_1 \cup Q_3 \cup h^*(Q_1))| + k^-(Q_2 \cup Q_3)$. Hence, it remains to upper bound $|Q_4|$ with $Q_4 = \text{ALG} \setminus (Q_1 \cup Q_2 \cup Q_3)$ by $4 \cdot k_h$. By definition, Q_4 only contains edges that are queried in Line 12. Thus, at least half the queries of Q_4 are elements of W that are part of the matching h. By Observation 5.7.3, no element of W is part of the partial matching $\bar{h}$ in Line 10. So, in each execution of Line 12, at least half the queries are not part of $\bar{h}$ in Line 10 but added to h in Line 11. Our goal is to bound the number of such elements.

We start with some definitions. Define G_j as the problem instance at the j'th execution of Line 11, and let G_0 denote the initial problem instance. For each G_j, let $\bar{G}_j = (\bar{V}_j, \bar{E}_j)$, T_L^j and T_U^j denote the corresponding vertex cover instance and lower and upper limit trees. Observe that each G_j has unique $T_L^j = T_U^j$, and, by Lemma 5.4.5, is prediction mandatory free. Let Y_j denote the set of queries made by the algorithm to transform instance G_{j-1} into instance G_j. We partition Q_4 into subsets S_j, where S_j contains the edges of Q_4 that are queried in the j'th execution of Line 12. We claim $|S_j| \leq 4 \cdot k^+(Y_j)$ for each j, which implies $|Q_4| \leq \sum_j |S_j| \leq 4 \cdot \sum_j k^+(Y_j) \leq 4 \cdot k_h$. To show the claim, we rely on the following lemma.

Lemma 5.7.6. *Let l, f be non-trivial edges in G_j such that $\{l, f\} \in \bar{E}_{j-1} \Delta \bar{E}_j$, then $k^-(l), k^-(f) \geq 1$. Furthermore, $k^+(Y_j) \geq |U|$ for the set U of all endpoints of such vertex cover instance edges $\{l, f\}$.*

Proof. We separately consider non-trivial edges l, f with $\{l, f\} \in \bar{E}_{j-1} \setminus \bar{E}_j$ and $\{l, f\} \in \bar{E}_j \setminus \bar{E}_{j-1}$

Case $\{l, f\} \in \bar{E}_{j-1} \setminus \bar{E}_j$. Since $\{l, f\} \in \bar{E}_{j-1}$, one edge in $\{l, f\}$ must be in T_L^{j-1} and one must be outside of T_L^{j-1} by definition of the vertex cover instance. Assume w.l.o.g. that $l \in T_L^{j-1}$ and $f \notin T_L^{j-1}$. Let C_f^{j-1} be the unique cycle in $T_L^{j-1} \cup \{f\}$ and let X_l^{j-1} be the set of edges between the two connected components of $T_L^{j-1} \setminus \{l\}$. Then, $l \in C_f^{j-1}$ and $f \in X_l^{j-1}$ hold again by the definition of the vertex cover instance. By Lemma 5.2.8 and because l and f are non-trivial in G_j, the fact that $l \in T_L^{j-1}$ and $f \notin T_L^{j-1}$ implies $l \in T_L^j$ and $f \notin T_L^j$.

Since $\{l, f\} \notin \bar{E}_j$, we have $l \notin C_f^j$ and $f \notin X_l^j$. Each cycle containing f must contain an edge of $X_l^{j-1} \setminus \{f\}$ and, therefore, there must be some $l' \in C_f^j$ with $l' \in X_l^{j-1} \setminus \{f, l\}$. This implies $l' \in T_L^j$. On the other hand, $T_L^{j-1} \cap X_l^{j-1} = \{l\}$ and, therefore, $l' \notin T_L^{j-1}$.

Lemma 5.2.8 implies that l' must have been queried while transforming instance G_{j-1} into instance G_j.

As G_{j-1} is prediction mandatory free, $\overline{w}_{l'} \geq U_l > L_f$ holds by Lemma 5.4.4. However, we have $w_{l'} \leq L_f$ as $w_{l'} > L_f$ would be a contradiction to T_L^j being a lower limit tree. We can conclude $\overline{w}_{l'} \geq U_l > w_{l'}$ and $\overline{w}_{l'} > L_f \geq w_{l'}$, which implies $k^-(l), k^-(f) \geq 1$.

Case $\{l,f\} \in \bar{E}_j \setminus \bar{E}_{j-1}$. Assume w.l.o.g. that $l \in T_L^j$ and $f \notin T_L^j$. Since l and f are non-trivial, Lemma 5.2.8 implies $e \in T_L^{j-1}$ and $f \notin T_L^{j-1}$.

As $\{f,l\} \in \bar{E}_j$ and $f \notin T_L^j$, the definition of the vertex cover instance $\bar{G}_j$ implies $l \in C_f^j$ such that $I_f \cap I_l \neq \emptyset$, where C_f^j is the unique cycle in $T_L^j \cup \{f\}$.

Remember that $f \in X_l^j$ since $f \notin T_L^j$ and $l \in C_f^j$. The fact that $\{f,l\} \notin \bar{E}_{j-1}$ implies $l \notin C_f^{j-1}$. Thus, there must be an element $l' \in T_L^{j-1} \cap (X_l^j \setminus \{l,f\})$ such that $l' \in C_f^{j-1}$.

As the instance is prediction mandatory free, $l' \in C_f^{j-1}$ implies $\overline{w}_{l'} \notin I_f$ (more precisely $\overline{w}_{l'} \leq L_f$). Further, $X_l^j \cap T_L^j = \{l\}$ holds by definition and implies $l' \notin T_L^j$. According to Lemma 5.2.8, $l' \in T_L^{j-1} \setminus T_L^j$ implies that l' must have been queried in order to transform instance G_{j-1} into instance G_j. If $w_{l'} \in I_l$, then $w_{l'} < U_l$, which implies that l does not have the smallest upper limit in cut X_l^j. This is a contradiction to T_L^j being an upper limit tree, which is the case as instance G_j is preprocessed.

If $w_{l'} \notin I_l$, then also $w_{l'} \geq U_l$ ($w_{l'} \leq L_l$ cannot be the case because l' would have the smallest lower limit in X_l^j which would contradict $l' \notin T_L^j$). As $I_l \cap I_f \neq \emptyset$ holds by definition of the vertex cover instance G_j, we can conclude that $w_{l'} \geq U_l$ implies $w_{l'} > L_f$. So we have $\overline{w}_{l'} \leq L_f < w_{l'}$, which implies $k^-(f) \geq 1$.

Similarly, $l \in C_f^j$, $I_l \cap I_f \neq \emptyset$ and l, f being non-trivial imply $L_f < U_l$. Thus, we have $\overline{w}_{l'} \leq L_f < U_l \leq w_{l'}$, which implies $k^-(l) \geq 1$.

Since all errors of both cases are caused by elements that have been queried to transform G_{j-1} into G_j, i.e., are contained in Y_j, the arguments in both cases also imply $k^+(Y_j) \geq |U|$ for the set U of all endpoints of vertex cover instance edges $\{l,f\}$ that are covered by one of the cases. □

Lemma 5.7.7. $|\mathrm{ALG}| \leq |\mathrm{OPT}| + 5 \cdot k_h$.

Proof. We show $|S_j| \leq 4 \cdot k^+(Y_j)$ for each j, which, in combination with Lemma 5.7.4, implies the lemma. Consider an arbitrary S_j and the corresponding set Y_j. Further, let h_{j-1} denote the maximum matching computed and used by the algorithm for vertex cover instance $\bar{G}_{j-1}$, and let $\bar{h}_{j-1} = \{\{e,e'\} \in h_{j-1} \mid \{e,e'\} \in \bar{E}_j\}$. Finally, let h_j denote the matching that the algorithm uses for vertex cover instance $\bar{G}_j$. That is, h_j is computed by completing $\bar{h}_{j-1}$ with a standard augmenting path algorithm. As already argued, at least half the elements of S_j are not matched by $\bar{h}_{j-1}$ but are matched by h_j (cf. Observation 5.7.3).

We bound the number of such elements that are not matched in $\bar{h}_{j-1}$ but become matched in h_j by exploiting that h_j is constructed from $\bar{h}_{j-1}$ via a standard augmenting path algorithm. By definition, each iteration of the augmenting path algorithm increases the size of the current matching (starting with $\bar{h}_{j-1}$) by one and, in doing so, matches two new elements. In total, at most $2 \cdot (|h_j| - |\bar{h}_{j-1}|)$ previously unmatched elements become part of the matching, where $|h_j|$ and $|\bar{h}_{j-1}|$ denote the size of the respective matching. Thus, $|S_j| \leq 4 \cdot (|h_j| - |\bar{h}_{j-1}|)$.

We show $(|h_j| - |\bar{h}_{j-1}|) \leq k^+(Y_j)$. According to Kőnig-Egerváry's Theorem, the size of h_j is equal to the size $|VC_j|$ of the minimum vertex cover for $\bar{G}_j$. We show $|VC_j| \leq |\bar{h}_{j-1}| + k^+(Y_j)$, which implies $(|h_j| - |\bar{h}_{j-1}|) = |VC_j| - |\bar{h}_{j-1}| \leq k^+(Y_j)$, and, thus, the claim. Let $\overline{VC}_{j-1} = \{e \in VC_{j-1} \mid \exists e' \text{ s.t. } \{e,e'\} \in \bar{h}_{j-1}\}$, then $|\overline{VC}_{j-1}| = |\bar{h}_{j-1}|$.

We prove that we can construct a vertex cover for $\bar{G}_j$ by adding at most $k^+(Y_j)$ elements to $\overline{VC}_{j-1}$, which implies $|VC_j| \leq |\bar{h}_{j-1}| + k^+(Y_j)$. Consider vertex cover instance $\bar{G}_j$ and set $\overline{VC}_{j-1}$. By definition, $\overline{VC}_{j-1}$ covers all edges that are part of partial matching $\bar{h}_{j-1}$.

Consider the elements of $\bar{V}_j$ that are an endpoint of an edge in $\{e,f\} \in \bar{E}_j \Delta \bar{E}_{j-1}$ with e,f non-trivial in G_j. By Lemma 5.7.6, $k^-(e) \geq 1$ for each such e and $k^+(Y_j) \geq |U|$ for the set U of all such elements. Thus, we can afford to add U to the vertex cover.

Next, consider an edge $\{e,f\} \in \bar{E}_j$ that is not covered by $\overline{VC}_{j-1} \cup U$. Since $\{e,f\}$ is not covered by U, it must hold that $\{e,f\} \in \bar{E}_j \cap \bar{E}_{j-1}$. Thus, $\{e,f\}$ was covered by VC_{j-1} but is not covered by $\overline{VC}_{j-1}$. This implies $\{e,f\} \cap VC_{j-1} \neq \emptyset$ but $\{e,f\} \cap \overline{VC}_{j-1} = \emptyset$. Assume w.l.o.g. that $e \in VC_{j-1}$. Then, there must be an e' such that $\{e,e'\} \in h_{j-1}$ but $\{e,e'\} \notin \bar{h}_{j-1}$.

This gives us that $\{e,e'\} \in \bar{E}_j \Delta \bar{E}_{j-1}$. Since $\{e,f\}$ is not covered by U, the edge e' must be trivial in G_j but non-trivial in G_{j-1} as otherwise $\{e,e'\} \in \bar{E}_j \Delta \bar{E}_{j-1}$ would imply $e \in U$. Thus, e' must have been queried (i) as a mandatory element (or a matching partner) in Line 6 or 13, (ii) as part of VC_{j-1} in Line 5 or (iii) in Line 12. Case (ii) implies $e' \in VC_{j-1}$, contradicting $e \in VC_{j-1}$. Cases (i) or (iii) imply a query to the matching partner e of e', which contradicts $\{e,f\} \in \bar{E}_j$ (as e would be trivial). Since all cases lead to a contradiction, we get that $\{e,f\}$ is covered by $\overline{VC}_{j-1} \cup U$, which implies that $\overline{VC}_{j-1} \cup U$ is a vertex cover for $\bar{G}_j$. Lemma 5.7.6 implies $|U| \leq k^+(Y_j)$. So $|VC_j| \leq |\bar{h}_{j-1}| + k^+(Y_j)$, which concludes the proof. □

The Lemmas 5.7.5 and 5.7.7 directly imply Theorem 5.7.2.

5.7.2 Error-Sensitive Algorithm for General Instances

We show that the algorithm that first executes Algorithm 15 and then Algorithm 17 satisfies the guarantees of Theorem 5.7.1. In order to do so, we carefully combine Theorems 5.5.1 and 5.7.2.

Proof of Theorem 5.7.1. We remark that we actually show a robustness of $\max\{3, \gamma + \frac{1}{|\text{OPT}|}\}$, which might be smaller than $\gamma + 1$.

Let $\text{ALG} = \text{ALG}_1 \cup \text{ALG}_2$ be the query set queried by the algorithm, where ALG_1 and ALG_2 are the queries of Algorithm 15 and Algorithm 17, respectively. Let $P \subseteq \text{ALG}_1$ denote the edges queried in the last iteration of Algorithm 15. Furthermore, let D denote the set of edges in $E \setminus \text{ALG}_1$ that can be deleted or contracted during the execution of Algorithm 15. It follows $D \cap \text{ALG}_2 = \emptyset$

Assume first that $\text{ALG}_2 = \emptyset$. Then, querying ALG_1 solves the problem. Theorem 5.5.1 directly implies

$$|\text{ALG}| = |\text{ALG}_1| \leq (1 + \frac{1}{\gamma}) \cdot (|\text{OPT} \cap (\text{ALG}_1 \cup D)| + k^+(\text{ALG}_1) + k^-(\text{ALG}_1))$$

and

$$|\text{ALG}_1 \setminus P| \leq \gamma \cdot |\text{OPT} \cap (\text{ALG}_1 \cup D)|$$

as the additive term of $\gamma - 2$ is caused by P.

Thus, the error-dependent guarantee follows immediately. However, due to the additive term of $\gamma - 2$ in the second term of the minimum, the second inequality does not directly transfer to $|\text{ALG}|$ as $|\text{ALG}| \leq (\gamma + 1) \cdot |\text{OPT}|$ might not hold. Recall that the additive term is caused exactly by the queries to P. Since the algorithm executes queries in the last iteration, the instance is not solved at the beginning of the iteration (a solved instance is prediction mandatory free and would lead to direct termination). Thus, $|\text{OPT} \setminus ((\text{ALG}_1 \cup D) \setminus P)| \geq 1$ and, therefore $|P| \leq (\gamma - 2) \cdot |\text{OPT} \setminus ((\text{ALG}_1 \cup D) \setminus P)|$. Ignoring the additive term caused by P, Theorem 5.5.1 implies $|\mathrm{ALG}_1 \setminus P| \leq \gamma \cdot |\mathrm{OPT} \cap ((\mathrm{ALG}_1 \cup D) \setminus P)|$. We can conclude

$$\begin{aligned}|\mathrm{ALG}| &= |\mathrm{ALG}_1 \setminus P| + |P| \\ &\leq \gamma \cdot |\mathrm{OPT} \cap ((\mathrm{ALG}_1 \cup D) \setminus P)| + (\gamma - 2) \cdot |\mathrm{OPT} \setminus ((\mathrm{ALG}_1 \cup D) \setminus P)| \\ &\leq \gamma \cdot |\mathrm{OPT}|.\end{aligned}$$

Now, assume that $\mathrm{ALG}_2 \neq \emptyset$. Let $\mathrm{OPT} = \mathrm{OPT}_1 \cup \mathrm{OPT}_2$ be an optimal query set with $\mathrm{OPT}_1 = \mathrm{OPT} \cap (\mathrm{ALG}_1 \cup D)$ and $\mathrm{OPT}_2 = \mathrm{OPT} \setminus (\mathrm{ALG}_1 \cup D)_,$. By Theorem 5.6.2, $\mathrm{ALG}_2 \neq \emptyset$ implies $\mathrm{OPT}_2 \neq \emptyset$. Thus, $|\mathrm{OPT}_2| \geq 1$. Furthermore, by Theorem 5.7.2, we have $|\mathrm{ALG}_2| \leq 3 \cdot |\mathrm{OPT}_2|$. As $|P| \leq \gamma - 2$ and $|\mathrm{OPT}_2| \geq 1$, this gives us $|\mathrm{ALG}_2 \cup P| \leq (\gamma + 1) \cdot |\mathrm{OPT}_2|$. Theorem 5.5.1 implies $|\mathrm{ALG}_1 \setminus P| \leq \gamma \cdot |\mathrm{OPT}_1 \setminus P|$ (as the additive term in Theorem 5.5.1 is caused by P). Together, the inequalities imply $|\mathrm{ALG}| \leq (\gamma + 1) \cdot |\mathrm{OPT}|$ and, thus $(\gamma + 1)$-robustness.

We remark that, for $\gamma \geq 3$, the robustness improves for increasing $|\mathrm{OPT}|$. To see this, note that if $\gamma \geq 3$, then $|\mathrm{ALG}_2 \cup P| \leq \gamma \cdot |\mathrm{OPT}_2| + 1$ and, in combination with $|\mathrm{ALG}_1 \setminus P| \leq \gamma \cdot |\mathrm{OPT}_1 \setminus P|$, also $|\mathrm{ALG}| \leq \gamma \cdot |\mathrm{OPT}| + 1$. Thus, for $\gamma \geq 3$, the robustness of the algorithm is actually $\gamma + \frac{1}{|\mathrm{OPT}|}$. For $\gamma = 2$, the robustness is $3 = \gamma + 1$. Combined, the robustness is $\max\{3, \gamma + \frac{1}{|\mathrm{OPT}|}\}$.

We continue by showing $|\mathrm{ALG}| \leq (1 + \frac{1}{\gamma}) \cdot |\mathrm{OPT}| + 5 \cdot k_h$. Theorem 5.5.1 implies $|\mathrm{ALG}_1| \leq (1 + \frac{1}{\gamma}) \cdot (|\mathrm{OPT}_1| + 2 \cdot \max\{k^+(\mathrm{ALG}_1), k^-(\mathrm{ALG}_1)\})$. Since $\gamma \geq 2$, we have $(1+\frac{1}{\gamma}) \cdot 2 < 5$ and can rewrite $|\mathrm{ALG}_1| \leq (1+\frac{1}{\gamma}) \cdot |\mathrm{OPT}_1| + 5 \cdot \max\{k^+(\mathrm{ALG}_1), k^-(\mathrm{ALG}_1)\}$. Theorem 5.7.2 implies $|\mathrm{ALG}_2| \leq |\mathrm{OPT}_2| + 5 \cdot k'_h$, where k'_h is the error for the input instance of the second phase. That is, the instance that does not contain any edge of ALG_1 since those can be deleted/contracted after the first phase. This implies for the error k_h of the complete instance that no error that is counted by $\max\{k^+(\mathrm{ALG}_1), k^-(\mathrm{ALG}_1)\}$ is considered by k'_h and, therefore, $\max\{k^+(\mathrm{ALG}_1), k^-(\mathrm{ALG}_1)\} + k'_h \leq k_h$. Thus, we can combine the inequalities to $|\mathrm{ALG}| \leq (1 + \frac{1}{\gamma}) \cdot |\mathrm{OPT}| + 5 \cdot k_h$. □

5.8 Concluding Remarks

In this chapter, we showed how to utilize untrusted predictions to achieve the optimal consistency and robustness tradeoff for the MST problem under explorable uncertainty. For accurate predictions, the bound improves upon the adversarial lower bound of two, while it matches the lower bound for arbitrarily wrong predictions. We also designed an algorithm with linear error-dependency on the hop distance k_h, which, in combination with the results of the previous chapter, nicely illustrates the utility of this measure for several problems under explorable uncertainty. Since this latter algorithm does not quite match the optimal consistency and robustness tradeoff, a next research step would be to investigate whether such a tradeoff is possible with error-dependency.

We remark that we can show PAC-learnability of the predictions w.r.t. k_h by using the same proof as in Section 4.5 for the hypergraph orientation problems, since this proof is problem independent. Furthermore, we also remark that the integrality condition on parameter γ can be removed via randomization using the same technique as in the proof of Theorem 4.3.12 for the hypergraph orientation problem with the same small additive loss in the guarantee.

Further generalizations of this problem include to consider arbitrary query costs and arbitrary matroids. All previous results on the MST problem under explorable uncertainty without predictions extend to the more general problem of finding a minimum weight base of a matroid [Hof+08; MMS17]. A next research step would be to check whether the same holds for the results of this chapter.

Chapter 6

Set Selection under Explorable Stochastic Uncertainty via Covering Techniques

In this chapter, we consider the set selection problem under explorable stochastic uncertainty. Given subsets of uncertain weights, we study the problem of identifying the subset of minimum total weight (sum of the uncertain weights contained in the set) by querying as few weights as possible. This *set selection problem* is of intrinsic importance within the field of explorable uncertainty as it implies strong adversarial lower bounds for a wide range of interesting combinatorial problems such as knapsack and matchings [Mei18]. We consider a stochastic problem variant and give algorithms that, in expectation, improve upon these adversarial lower bounds. The key to our results is to prove a strong structural connection to a seemingly unrelated covering problem with uncertainty in the constraints via a linear programming formulation. We exploit this connection to derive an algorithmic framework that can be used to solve both problems under uncertainty, obtaining nearly tight bounds on the competitive ratio. This is the first non-trivial stochastic result concerning the sum of unknown weights without further structure known for the set. In contrast to most existing results on explorable uncertainty, the analysis of our algorithm does not rely on witness sets at all. We hope that our novel approach for tackling the set selection problem lays the foundation for solving more general problems in the area of explorable uncertainty.

Bibliographic remark: This chapter is mainly based on joint work with N. Megow [MS23]. Therefore, some parts correspond to or are identical with [MS23].

Contents

6.1 Introduction

In the previous chapters of this thesis, we considered hypergraph orientation, sorting and the minimum spanning tree problem under explorable uncertainty. All these problems have in common that they admit constant competitive ratios, even in the adversarial setting, that can be improved to smaller constants beyond the worst-case. In this chapter, we move on to problems that adversarially only allow competitive ratios linear in the input size and our goal is to improve to sublinear competitive ratios beyond the worst-case.

To this end, we mainly consider the *set selection problem* (MINSET) under explorable uncertainty. In this problem, we are given a set of n uncertain weights represented by uncertainty intervals $\mathcal{I} = \{I_1, \dots, I_n\}$ and a family of m sets $\mathcal{S} = \{S_1, \dots, S_m\}$ with $S \subseteq \mathcal{I}$ for all $S \in \mathcal{S}$. A precise weight w_i lies in its uncertainty interval I_i, is initially unknown, and can be revealed via a query. The precise weight of a subset $S \in \mathcal{S}$ is $w(S) = \sum_{I_i \in S} w_i$. Our goal is to determine a subset of minimum precise weight as well as the corresponding precise weight by using a minimal number of queries. It can be seen as an optimization problem with uncertainty in the coefficients of the objective function:

$$\begin{array}{llll} \min & \sum_{j=1}^{m} x_j & \sum_{I_i \in S_j} w_i & \\ \text{s.t.} & \sum_{j=1}^{m} x_j & = 1 & \\ & x_j & \in \{0,1\} & \forall j \in \{1, \dots, m\}. \end{array} \qquad (\text{SETSELIP})$$

Since the precise w_i's are uncertain, we do not always have sufficient information to just compute an optimal solution to (SETSELIP) and instead might have to execute queries in order to determine such a solution. An algorithm for MINSET under uncertainty can adaptively query intervals to reveal weights until it has sufficient information to determine an optimal solution to (SETSELIP). Adaptivity in this context means that the algorithm can take previous query results into account to decide upon the next query.

In this chapter, we consider the stochastic problem variant, where we assume that all weights w_i are drawn independently at random from their intervals I_i according to unknown distributions d_i. As usual, we analyze an algorithm ALG in terms of its *competitive ratio* (see also Section 2.2): for the set of problem instances $\mathcal{J}$, it is defined as $\max_{J \in \mathcal{J}} \mathbb{E}[\text{ALG}(J)]/\mathbb{E}[\text{OPT}(J)]$, where $\text{ALG}(J)$ is the number of queries needed by ALG to solve instance J, and $\text{OPT}(J)$ is the minimum number of queries necessary to solve the instance.

MINSET is a fundamental problem and of intrinsic importance within the field of explorable uncertainty. The majority of existing works considers the *adversarial setting*, where query outcomes are not stochastic but returned in a worst-case manner. Selection type problems have been studied in the adversarial setting and constant (matching) upper and lower bounds are known, e.g., for selecting the minimum [Kah91], the k-th smallest element [Kah91; Fed+03], a minimum spanning tree [Hof+08; EH14; MMS17; Erl+22], sorting [HL21] and geometric problems [Bru+05]. However, these problems essentially boil down to comparing *single* uncertainty intervals and identifying the minimum of two unknown weights. Once we have to compare two (even disjoint) sets and the corresponding *sums* of unknown weights, no deterministic algorithm can have a better adversarial competitive ratio than n, the number of uncertainty intervals. This has been shown by Erlebach et al. [EHK16] for MINSET, and it implies adversarial lower bounds for classical combinatorial problems, such as, knapsack [Mei18] and matchings [Mei18], and solving integer linear programs (ILPs) with uncertainty in the cost coefficients [Mei18] as in (SETSELIP) above. Thus, solving MINSET under stochastic uncertainty is an important step towards obtaining improved results for this range of problems. As a main result, we provide substantially better algorithms for MINSET under stochastic

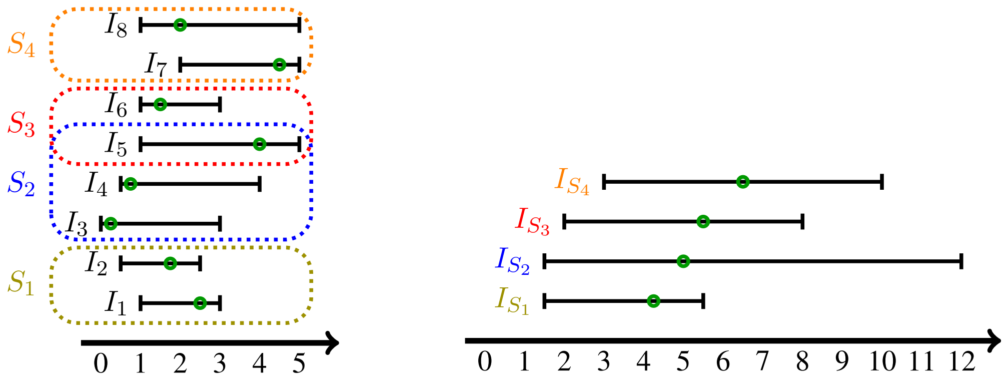


FIGURE 6.1: Instance for set selection under explorable uncertainty with intervals $\mathcal{I} = \{I_1, I_2, \dots, I_8\}$ and sets $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$ with $S_1 = \{I_1, I_2\}$, $S_2 = \{I_3, I_4, I_5\}$, $S_3 = \{I_4, I_5, I_6\}$ and $S_4 = \{I_7, I_8\}$ (left) and the corresponding uncertainty intervals I_{S_j} for the sets S_j with $j \in \{1, \dots, 4\}$ (right). Green circles illustrate the precise weights.

uncertainty. This is a key step for breaching adversarial lower bounds for a wide range of problems.

For the *stochastic setting*, the only related results we are aware of concern sorting [Cha+21] and hypergraph orientation [Bam+21] (see also Chapter 3). Asking for the *sum* of unknown weights is substantially different.

6.1.1 The Covering Point of View

Our key observation is that we can view MINSET as a covering problem with uncertainty in the constraints. To see this, we focus on the structure of the uncertainty intervals and how a query affects it. We assume that each interval $I_i \in \mathcal{I}$ is either open (*non-trivial*) or *trivial*, i.e., $I_i = (L_i, U_i)$ or $I_i = \{w_i\}$; a standard technical assumption in explorable uncertainty (cf. Section 2.2.1 in Chapter 2). In the latter case, $L_i = U_i = w_i$. We call L_i and U_i the *lower and upper limit*. For a set $S \in \mathcal{S}$, we define the lower limit $L_S = \sum_{I_i \in S} L_i$ and upper limit $U_S = \sum_{I_i \in S} U_i$. If S contains only trivial uncertainty intervals, then we define $I_S = [L_S, U_S] = \{w(S)\}$ and call I_S *trivial*. Otherwise, we define $I_S = (L_S, U_S)$. Clearly, the weight $w(S)$ of a set $S \in \mathcal{S}$ is contained in the interval I_S, i.e., $w(S) \in I_S$. We call I_S the *uncertainty interval* of set S. See Figure 6.1 for an example.

Since the intervals I_S of the sets $S \in \mathcal{S}$ can overlap, we might have to execute queries to determine the set of minimum precise weight. A query to an interval I_i reveals the precise weight w_i and, thus, replaces both, L_i and U_i, with w_i. In a sense, a query to an $I_i \in S$ reduces the range (L_S, U_S) in which $w(S)$ might lie by increasing L_S by $w_i - L_i$ and decreasing U_S by $U_i - w_i$. See Figure 6.2 for an example. We use L_S and U_S to refer to the initial limits and $L_S(Q)$ and $U_S(Q)$ to denote the limits of a set $S \in \mathcal{S}$ *after* querying a set of intervals $Q \subseteq \mathcal{I}$.

Let $w^* = \min_{S \in \mathcal{S}} w(S)$ be the initially uncertain minimum precise set weight. To solve the problem, we have to adaptively query a set of intervals Q until $U_{S^*}(Q) = L_{S^*}(Q) = w^*$ holds for some $S^* \in \mathcal{S}$ and $L_S(Q) \geq w^*$ holds for all $S \in \mathcal{S}$. Only then, we know for sure

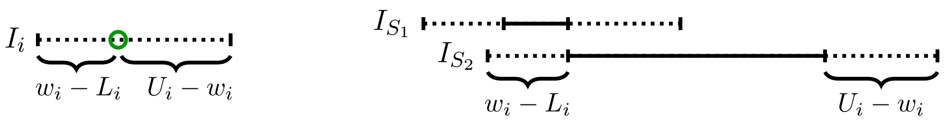


FIGURE 6.2: Example of how a query to an interval I_i changes the intervals of two sets S_1, S_2 with $I_i \in S_1 \cap S_2$ in the set selection problem under explorable uncertainty.

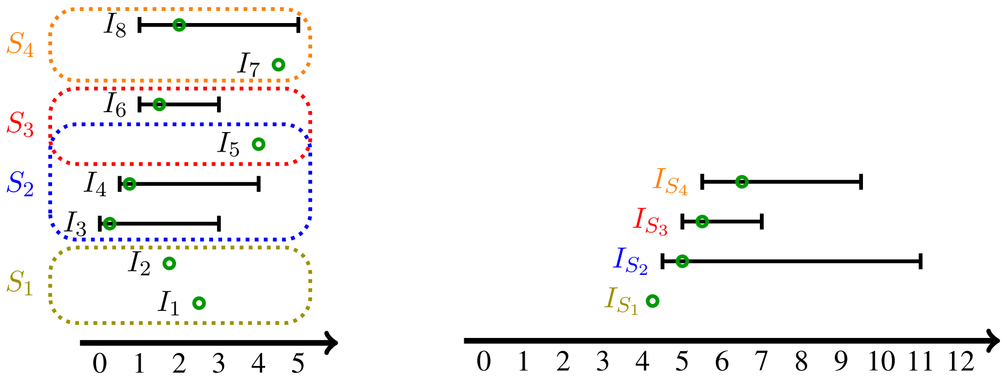


FIGURE 6.3: Instance of Figure 6.1 after querying $Q = \{I_1, I_2, I_5, I_7\}$: Updated uncertainty intervals $\mathcal{I}$ (left) and updated set uncertainty intervals (right).

that w^* is indeed the minimum set weight and that S^* achieves this weight. Figure 6.3 shows the structure of an instance that has been solved. For an instance $(\mathcal{I}, \mathcal{S})$ of MINSET, the following integer linear program (ILP) with $a_i = w_i - L_i$ for all $I_i \in \mathcal{I}$ and $b_S = w^* - L_S$ for all $S \in \mathcal{S}$ formulates this problem:

$$\begin{array}{lll} \min & \sum_{I_i \in \mathcal{I}} x_i & \\ \text{s.t.} & \sum_{I_i \in S} x_i \cdot a_i \geq b_S & \forall S \in \mathcal{S} \\ & x_i \in \{0,1\} & \forall I_i \in \mathcal{I} \end{array} \qquad \text{(MINSETIP)}$$

Here, the variable x_i, $I_i \in \mathcal{I}$, indicates whether interval I_i is selected to be queried ($x_i = 1$) or not ($x_i = 0$) and our objective is to minimize the number of queries.

Observe that this ILP is a special case of the *multiset multicover problem* (see, e.g., [RV98]). If $a_i = w_i - L_i = 1$ for all $I_i \in \mathcal{I}$ and $b_S = w^* - L_S = 1$ for all $S \in \mathcal{S}$, then the problem is exactly the classical SETCOVER problem with $\mathcal{I}$ corresponding to the SETCOVER *sets* and $\mathcal{S}$ corresponding to the SETCOVER *elements*.

The optimal solution to (MINSETIP) is the optimal query set for the corresponding MINSET instance; this is not hard to see but we also formally prove it with the following lemma.

Lemma 6.1.1. *Solving* MINSET *is equivalent to solving* (MINSETIP).

Proof. We show the lemma by proving the following claim: A query set $Q \subseteq \mathcal{I}$ is feasible for MINSET if and only if vector x, with $x_i = 1$ for all $I_i \in Q$ and $x_i = 0$ otherwise, is a feasible solution for the corresponding (MINSETIP).

Let Q be feasible for MINSET, and let x be a vector with $x_i = 1$ for all $I_i \in Q$ and $x_i = 0$ otherwise. By definition, each feasible solution for MINSET must query all non-trivial elements of some set S^* with $w(S^*) = w^*$ as this is the only way to determine the minimum set weight w^*. Let $N(S^*)$ denote those non-trivial elements, then we can rewrite the initial lower limit of S^* as $L_{S^*} = w^* + \sum_{I_i \in N(S^*)}(L_i - w_i)$. This implies $w^* - L_{S^*} = \sum_{I_i \in N(S^*)}(w_i - L_i)$. As $N(S^*) \subseteq Q$, we get $\sum_{I_i \in S^*} x_i \cdot (w_i - L_i) = \sum_{I_i \in S^* \cap Q}(w_i - L_i) \geq w^* - L_{S^*}$. Thus, x satisfies the constraint for S^*.

Next, consider a set $S \in \mathcal{S}$ with $S \neq S^*$. Since Q is feasible, $L_S(Q)$ has to be at least w^* as otherwise querying Q would not prove that w^* is indeed the minimum set weight. After querying Q, the lower limit of set S is

$$\begin{aligned} L_S(Q) &= \sum_{I_i \in S \setminus Q} L_i + \sum_{I_i \in S \cap Q} w_i \\ &= \sum_{I_i \in S} L_i - \sum_{I_i \in S \cap Q} L_i + \sum_{I_i \in S \cap Q} w_i \\ &= L_S + \sum_{I_i \in S \cap Q} (w_i - L_i). \end{aligned}$$

Thus, $L_S(Q) = L_S + \sum_{I_i \in S \cap Q}(w_i - L_i) \geq w^*$ must hold, which implies $\sum_{I_i \in S \cap Q}(w_i - L_i) \geq w^* - L_S$ and $\sum_{I_i \in S} x_i \cdot (w_i - L_i) \geq w^* - L_S$. We can conclude that x is feasible.

For the other direction consider a feasible solution x for (MINSETIP) and the corresponding set $Q = \{I_i \mid x_i = 1\}$. Consider some set S^* with $w(S^*) = w^*$. As $\sum_{I_i \in N(S^*)}(w_i - L_i) = w^* - L_{S^*}$ and $\sum_{I_i \in S^* \setminus N(S^*)}(w_i - L_i) = \sum_{I_i \in S^* \setminus N(S^*)}(w_i - w_i) = 0$, it must hold $x_i = 1$ for all $I_i \in N(S^*)$ for x to be feasible. Thus, Q contains all non-trivial elements of some set S^* with $w(S^*) = w^*$. To show that Q is a feasible solution for MINSET, it remains to show that $L_S(Q) \geq w^*$ for all $S \neq S^*$. Consider an arbitrary $S \neq S^*$. As $\sum_{I_i \in S} x_i \cdot (w_i - L_i) \geq (w^* - L_S)$, we have $\sum_{I_i \in S \cap Q}(w_i - L_i) \geq w^* - L_S$, which implies $L_S(Q) = L_S + \sum_{I_i \in S \cap Q}(w_i - L_i) \geq w^*$. □

The lemma shows that (MINSETIP) models MINSET. Under uncertainty however, the coefficients $a_i = w_i - L_i$ and right-hand sides $b_S = w^* - L_S$ of the ILP are unknown to us. We only know that $a_i \in (L_i - L_i, U_i - L_i) = (0, U_i - L_i)$ because $a_i = (w_i - L_i)$ and $w_i \in (L_i, U_i)$. Only once we query an interval I_i, the precise weight w_i and, thus, the coefficient a_i is revealed to us. In a sense, to solve MINSET under uncertainty, we have to solve (MINSETIP) with uncertainty in the coefficients and with irrevocable decisions. For the rest of the chapter, we interpret MINSET under uncertainty in exactly that way: We have to solve (MINSETIP) without knowing the coefficients in the constraints. Whenever we *irrevocably* add an interval I_i to our solution (i.e., set x_i to 1), the information on the coefficients (in form of w_i) is revealed to us. Our goal is to add elements to our solution until it becomes feasible for (MINSETIP), and to minimize the number of added elements. In this interpretation, the terms “querying an element” and “adding it to the solution” are interchangeable, and we use them as such.

Our main contribution is an algorithmic framework that exploits techniques for classical covering problems and adapts them to handle uncertainty in the coefficients a_i and the right-hand sides b_S. This framework allows us to obtain improved results for MINSET under stochastic uncertainty and for a variant of that problem with deterministic right-hand sides.

6.1.2 Our Results

We design a polynomial-time algorithm for MINSET under stochastic uncertainty with competitive ratio $\mathcal{O}(\frac{1}{\tau} \cdot \log^2 m)$, where m is the number of sets (number of constraints in (MINSETIP)) and parameter τ characterizes how “balanced” the distributions of precise weights within the given intervals are. More precisely, $\tau = \min_{I_i \in \mathcal{I}} \tau_i$ and τ_i is the probability that w_i is larger than the center of I_i (e.g., for uniform distributions $\tau = \frac{1}{2}$). All our results assume $\tau > 0$. We remark that the hidden constants in the performance bound depend on the upper limits of the given intervals. Assuming those to be constant is a common assumption; see, e.g., [MY20]. Even greedy algorithms for covering problems similar to (MINSETIP) *without* uncertainty have such dependencies [RV98; Vaz01; Dob82]. While there exist non-greedy algorithms for covering problems without such dependencies [KY01; KY05], it remains open whether they can be adjusted to the setting with uncertainty and, in particular, irrevocable decisions.

Dependencies on parameters such as τ are quite standard and necessary [MY20; Blu+20; GV06a; Von07; BBD22]. For example, in [MY20] the upper bounds depend on the probability to draw the largest value of the uncertainty interval, which is an even stricter assumption that does not translate to open intervals.

Our result is the first stochastic result in explorable uncertainty concerning the sum of unknown weights and it builds on new methods that shall be useful for solving more general problems in this field. The ratio is independent of the number of elements, n. In particular for a small number of sets, m, this is a significant improvement upon the adversarial lower bound of n [EHK16].

As MINSET contains the classical SETCOVER problem, an approximation factor better than $\mathcal{O}(\log m)$ is unlikely, unless P=NP [DS14]. We show that this holds also in the stochastic setting, even with uniform distributions. We further show that $\frac{2}{\tau}$ is a lower bound for MINSET under stochastic explorable uncertainty, even if the sets are pairwise disjoint. Hence, the dependencies on $\log m$ and $\frac{2}{\tau}$ in our upper bounds are necessary.

In the special case that all given sets are disjoint, we provide a simpler algorithm with competitive ratio $\frac{2}{\tau}$, which matches the lower bound. This is a gigantic improvement compared to the adversarial setting, where the lower bound of n holds even for disjoint sets [EHK16].

We remark that all our results for MINSET translate to the maximization variant of the problem, where we have to determine the set of *maximum* weight (cf. Section 6.6).

Algorithmically, we exploit the covering point of view to introduce a class of greedy algorithms that use the same basic strategy as the classical SETCOVER greedy algorithm [Chv79]. However, we do not have sufficient information to compute and query an exact greedy choice under uncertainty as this choice depends on uncertain parameters. Instead, we show that it is sufficient to query a small number of elements that together achieve a similar greedy value to the exact greedy choice. If we do this repeatedly and the number of queries per iteration is small in expectation, then we achieve guarantees comparable to the approximation factor of a greedy algorithm with full information. It is worth noting that this way of comparing an algorithm to the optimal solution is a novelty in explorable uncertainty as all previous algorithms for adversarial explorable uncertainty (MINSET and other problems) exploit *witness sets*. A witness set is a set of queries Q such that each feasible solution has to query at least one element of Q, which allows to compare an algorithm with an optimal solution.

We also consider the variant of (MINSETIP) under uncertainty with *deterministic right-hand sides*. We give a simplified algorithm with improved competitive ratio $\mathcal{O}(\frac{1}{\tau} \cdot \log m)$.

6.1.3 Further Previous Work

Since MINSET under uncertainty can be interpreted as both, a query minimization problem and a covering problem with uncertainty, we in the following summarize related previous work from both fields.

Previous Related Work on Query Problems

For adversarial MINSET under uncertainty, Erlebach et al. [EHK16] show a (best possible) competitive ratio of $2d$, where d is the cardinality of the largest set. In the lower bound instances, $d \in \Omega(n)$. The algorithm repeatedly queries disjoint witness sets of size at most $2d$. This result was stated for the setting in which it is not necessary to determine the precise weight of the minimal set; if the weight has to be determined, the bounds change from $2d$ to d (cf. Section 2.3 in Chapter 2).

Further related work on MINSET includes the result by Maehara and Yamaguchi [MY20], who consider packing ILPs with (stochastic) uncertainty in the cost coefficients, which can be queried. They present a framework for solving several problems and bound the absolute number of iterations that it requires to solve them, instead of the competitive ratio. However, as we argue down below, their algorithm has competitive ratio $\Omega(n)$ for MINSET under uncertainty, even for uniform distributions. Thus, it does not improve upon the adversarial lower bound.

Also, Wang et al. [WGW22] consider selection-type problems in a somewhat related model. In contrast to our setting, they consider different constraints on the set of queries that, in a way, imply a budget on the number of queries. They solve optimization problems with respect to this budget, which has a very different flavor than our setting of minimizing the number of queries.

Furthermore, there is related work in a setting, where a query reveals the *existence* of entities instead of numeric values, e.g., the existence of edges in a graph, c.f. [Blu+20; GV06a; Von07]. For example, Behnezhad et al. [BBD22] showed that vertex cover can be approximated within a factor of $(2+\epsilon)$ with only a constant number of queried edges per vertex. As edges define constraints, the result considers uncertainty only in the right-hand sides.

Comparison with Maehara and Yamaguchi [MY20]

We consider the framework by Maehara and Yamaguchi [MY20] on the set selection problem. Since their algorithm is designed for maximization problems, we consider the maximization variant of MINSET, i.e., we have to find the set $S \in \mathcal{S}$ of *maximum* $w(S)$ and determine the corresponding weight. We remark that all our results also translate to the maximization variant (cf. Section 6.6).

The algorithm by Maehara and Yamaguchi, in each iteration, solves the LP-relaxation of the *optimistic* version of the given ILP, which assumes $w_i = U_i$ for all $I_i \in \mathcal{I}$. In this case, the ILP under consideration formalizes the set selection problem (and *not* the query minimization problem as formalized by (MINSETIP)). The following LP-relaxation formulates the optimistic LP for a set selection instance $(\mathcal{I}, \mathcal{S})$:

$$\begin{array}{lll}
\max & \sum_{I_i \in \mathcal{I}} x_i \cdot U_i & \\
\text{s.t.} & \sum_{I_i \in S} x_i \geq y_S \cdot |S| & \forall S \in \mathcal{S} \\
& \sum_{I_i \in \mathcal{I}} x_i \leq \sum_{S \in \mathcal{S}} y_S \cdot |S| & \\
& \sum_{S \in \mathcal{S}} y_S = 1 & \\
& 0 \leq x_i \leq 1 & \forall I_i \in \mathcal{I} \\
& 0 \leq y_S \leq 1 & \forall S \in \mathcal{S}
\end{array}$$

Here variable y_S models whether set S is selected as the set of maximum weight ($y_S = 1$) or not ($y_S = 0$) and the third constraint makes sure that, at least integrally, exactly one set is selected. The variables x_i model whether an interval I_i is part of the selected set or not, and the first two constraints ensure that (integrally) exactly the members of the selected set S (with $y_S = 1$) are selected. Note that we use this ILP instead of (SETSELIP) because the algorithm by Maehara and Yamaguchi requires variables that correspond to elements that can be queried. The algorithm in each iteration solves the LP-relaxation to obtain an optimal fractional solution (x, y), and queries each I_i with probability x_i.

In the following, we give an instance of the set selection problem under stochastic explorable uncertainty for which the algorithm has a competitive ratio of $\Omega(n)$. Let $\mathcal{I} = \{I_0, \ldots, I_n\}$ with $I_0 = \{n\}$ and $I_i = (0, 1+\epsilon)$, $i > 0$, for some small $\epsilon > 0$. Let $\mathcal{S} = \{S_1, S_2\}$ with $S_1 = \{I_0\}$ and $S_2 = \{I_1, \ldots, I_n\}$. Assume uniform distributions and consider the algorithm that queries the intervals of S_2 in an arbitrary order. In expectation, this algorithm only needs a constant number of queries to solve the instance and prove $w(S_1) > w(S_2)$.

The algorithm by Maehara and Yamaguchi on the other hand, in the first iteration, solves the LP-relaxation and obtains the optimal solution (x, y) with $x_0 = 0$ and $x_i = 1$ for all $i \geq 1$. This means that all elements of S_2 are queried with a probability of 1. Thus, the algorithm queries at least n elements, which implies a competitive ratio of $\Omega(n)$. This means that applying the algorithm by Maehara and Yamaguchi [MY20] to the set selection problem does not improve upon the adversarial lower bound.

Previous Work on Covering Problems with Uncertainty

We continue by summarizing previous work on covering problems in different adversarial and stochastic settings.

In the *online* version of SETCOVER [Alo+09], we are given a ground set of elements and a family of subsets of these elements. In contrast to offline SETCOVER, we do not necessarily have to cover all elements of the ground set. Instead, the members of the ground set that we do actually have to cover arrive online in an adversarial manner. Whenever an element arrives, we have to cover it by irrevocably adding a set containing the element to our solution, unless a previously added set already contains the element. In a sense, online SETCOVER is a variant of (MINSETIP) under uncertainty, where only the right-hand sides are uncertain in $\{0, 1\}$ and all left-hand side coefficients are known and either one or zero. In contrast to MINSET under uncertainty, the adversary for online SETCOVER is in a sense more powerful when selecting the right-hand sides as they do not depend on a common weight w^*. Because of these differences, online SETCOVER has a very different flavor to MINSET under uncertainty. The same holds for the stochastic version of online SETCOVER [GKL23; Gra+13], where the subset of elements to be covered is drawn from a probability distribution.

A different stochastic variant of SETCOVER considers a *two-stage* version of the problem [SS04]. In the first stage, we do not yet know which members of the ground set actually need to be covered. After the first stage, the elements to be covered are drawn from a probability distribution and in the second stage we have full knowledge of the elements to be covered. The crux of this two-stage variant is that adding sets to the solution in the first stage can be cheaper than adding them to the solution during the second stage. This again leads to a very different flavor than our setting.

While these SETCOVER variants consider uncertainty in the set of elements that need to be covered, Goemans and Vondrák [GV06b] consider a variant where the elements to be covered are certain but there is uncertainty in which elements are covered by the sets. For each set, a vector describing the elements that are covered by the set is drawn according to a probability distribution. This corresponds to a variant of (MINSETIP), where all right-hand sides are one but the left-hand side coefficients are uncertain in $\{0, 1\}$. Even in comparison to MINSET under uncertainty with deterministic right-hand sides, there are several further difference besides the restriction of the coefficients to values in $\{0, 1\}$. For one, [GV06b] assumes access to the probability distributions. In particular, their algorithms are able to compute certain expected values. For our stochastic setting, we do not have sufficient information to compute expected values and the adversary still has some power in the selection of the unknown distributions as long as it respects the balancing parameter. On the other hand, their SETCOVER variant allows some distributions that are not possible in MINSET. In particular, an interval I_i in MINSET has the same coefficient $a_i = (w_i - L_i)$ in each constraint for a set S with $I_i \in S$. Such a restriction does not exist in the problem considered in [GV06b]. This in a sense makes their problem incomparable to MINSET under uncertainty. Furthermore, [GV06b] analyzes the approximation ratio instead of the competitive ratio. That is, they compare the expected objective value of an algorithm against the expected objective of the best possible algorithm instead of the expected optimum. To that end, they give an m-approximation for the stochastic SETCOVER variant. If sets can be added to the solution multiple times while each time drawing a new realization from the same distribution, they give a $\mathcal{O}(\log m)$-approximation.

Besides related work on stochastic SETCOVER variants, there is previous related work on the more general *(stochastic) submodular covering problem* (cf., e.g., [Wol82; AAK19; GGN21]). In the *submodular covering problem*, we are given a ground set of elements E and a submodular function $f\colon 2^E \to \mathbb{N}_+$. The goal is to find a subset $S \subseteq E$ of minimum cardinality such that $f(S) = f(E)$. This non-stochastic submodular covering problem contains offline MINSET [Wol82], i.e., (MINSETIP) with full knowledge of the coefficients and right-hand sides. To see this, consider an instance $(\mathcal{I}, \mathcal{S})$ of MINSET. We can interpret the intervals as the ground set of elements, i.e., $E = \mathcal{I}$ and use the submodular function $f(Q) = \sum_{S \in \mathcal{S}} \min\{\sum_{I_i \in S \cap Q} w_i - L_i, w^* - L_S\}$ for $Q \subseteq E$. Then, $f(E)$ is the sum of right-hand sides of (MINSETIP) and $f(Q) = f(E)$ holds for a subset $Q \subseteq \mathcal{I} = E$ if and only if Q is feasible for (MINSETIP). The best-known algorithm for the submodular covering problem achieves an approximation ratio of $\mathcal{O}(\log(f(E)))$ [Wol82] and no polynomial-time algorithm can be better unless P=NP [DS14].

In the *stochastic submodular covering problem*, we are given random variables $X_1, \ldots, X_n$ that independently realize to subsets of E according to known probability distributions. The task is to sequentially and irrevocably add random variables X_i to the solution $\mathcal{X}$ until $f(\bigcup_{X_i \in \mathcal{X}} X_i) = f(E)$. Whenever a random variable X_i is added to the solution, the realization of the variable is revealed. While this general setting is similar to MINSET under uncertainty, there are some differences. In the stochastic submodular covering problem, the value $f(\bigcup_{X_i \in \mathcal{X}} X_i)$ only depends on the realizations of the random variables in $\mathcal{X}$. For the submodular function defined above for a MINSET instance, the function value $f(Q)$ depends also on the uncertain w^* and, therefore, on elements outside of Q. Furthermore, as the intervals $\mathcal{I}$ in a MINSET instance are continuous, modeling them as a stochastic submodular covering instance would require some form of discretization. Independent of these differences, all results on the stochastic submodular covering problem (to our knowledge) assume known distributions and actively use them, which is in contrast to our stochastic setting. Furthermore, all these results analyze the approximation ratio instead of the competitive ratio. Thus, existing results for the stochastic submodular covering problem cannot directly be applied to our stochastic setting. This also holds for a range of problem variants that have been considered in the literature (see, e.g., [GK11; KNN17; INZ16; DHK16; NKN20]).

6.1.4 Outline

To start the rest of the chapter, we, in Section 6.2, consider the special case of MINSET under uncertainty with pairwise disjoint sets. For this special case, we give a lower bound of $\frac{2}{\tau}$ on the competitive ratio and a matching upper bound. These bounds nicely illustrate the challenges caused by the uncertainty and the techniques that we use to tackle them, also later on for the general problem.

Afterwards, in Section 6.3, we move on to the general MINSET and discuss the hardness of approximation as well as approximations of the offline problem variant. Based on observations for the offline problem, we introduce an algorithmic framework that can be used to solve MINSET under uncertainty.

For the remaining chapter, we show how to implement the framework for MINSET under uncertainty with deterministic right-hand sides (Section 6.4) and for the general MINSET (Section 6.5). Using these implementations and our observations for the special case of disjoint sets, we prove our algorithmic results.

6.2 Disjoint MINSET

Consider the special case of MINSET where all sets are pairwise disjoint, i.e., $S \cap S' = \emptyset$ for all $S, S' \in \mathcal{S}$ with $S \neq S'$. We call this special case *disjoint* MINSET. Disjoint MINSET is of particular interest as it gives lower bounds for several problems under adversarial explorable uncertainty, cf. [EHK16; Mei18]. To illustrate the challenges posed by having stochastic uncertainty in the input, we give the following lower bound. Recall that the balancing parameter is defined as $\tau = \min_{I_i \in \mathcal{I}} \tau_i$, where τ_i is the probability that w_i is larger than the center of I_i. We use ALG and OPT to refer to an algorithm and an optimal solution, respectively. Slightly abusing the notation, we use the same terms to also refer to the corresponding numbers of queries.

First, we show the following lower bound that even holds for known probability distributions. Afterwards, we prove a slightly stronger bound exploiting unknown distributions.

Theorem 6.2.1. *For any $\tau > 0$, no deterministic algorithm for* MINSET *under uncertainty has a competitive ratio better than $\frac{1}{\tau}$, even if all given sets are pairwise disjoint and the distributions are known.*

Proof. Consider an instance with the set of uncertainty intervals $\mathcal{I} = \{I_0, I_1, \dots, I_n\}$ with $I_0 = \{0.65\}$ and $I_i = (0, 1)$ for all $i \neq 0$, and sets $\mathcal{S} = \{S_1, S_2\}$ with $S_1 = \{I_0\}$ and $S_2 = \mathcal{I} \setminus \{I_0\}$. See Figure 6.4 for an illustration. Define the distributions d_i with $i \neq 1$ as $d_i(a) = (1 - \tau)$ if $a = \epsilon$, $d_i(a) = \tau$ if $a = 0.7$ and $d_i(a) = 0$ otherwise, for some infinitesimally small $\epsilon > 0$.

If there exists some $I_i \in S_2$ with $w_i = 0.7$, then OPT $= 1$ as a query to that interval already proves that S_1 is the set of minimum weight since $w(S_1) = 0.65 < 0.7$. Otherwise, OPT $= n$. Therefore, $\mathbb{E}[\text{OPT}] = (1 - (1 - \tau)^n) + (1 - \tau)^n \cdot n$ and $\lim_{n \to \infty} \mathbb{E}[\text{OPT}] = 1$.

Since I_0 is trivial and all I_i with $i \neq 0$ are identical with the same distribution, each deterministic algorithm ALG will just query the elements of S_2 in some order until it either reaches an I_i with $w_i = 0.7$ or has queried all intervals. This implies that ALG is a geometrical distribution with success probability τ and, therefore, $\mathbb{E}[\text{ALG}] = \min\{n, \frac{1}{\tau}\}$. For n towards infinity, we get

$$\lim_{n \to \infty} \frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OPT}]} = \frac{1}{\tau}.$$

$\square$

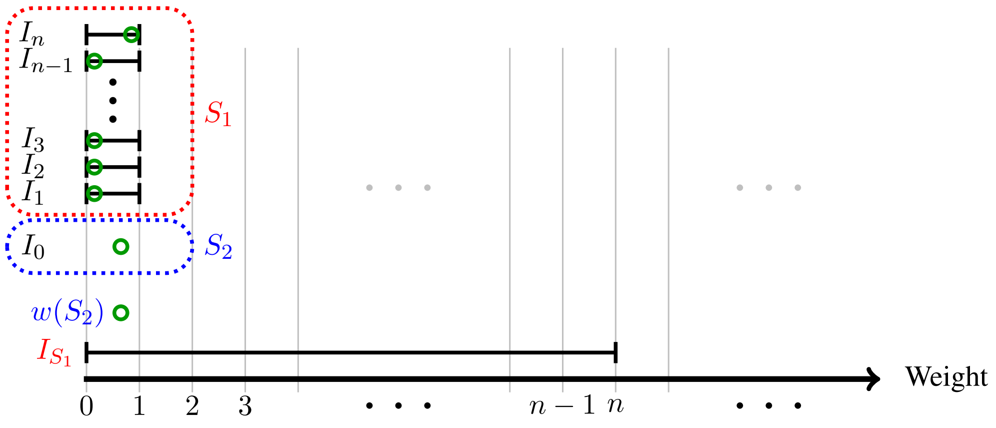


FIGURE 6.4: Lower bound example for the set selection problem under explorable uncertainty consisting of the intervals $\mathcal{I} = \{I_0, \dots, I_n\}$ and the sets $\mathcal{S} = \{S_1, S_2\}$ with $S_1 = \{I_1, \dots, I_n\}$, $S_2 = \{I_0\}$, $I_0 = \{0.65\}$ and $I_i = (0, 1)$ for $i \in \{1, \dots, d\}$.

Next, we show a slightly stronger bound for unknown distributions. The lower bound instance heavily exploits that, for unknown distributions, the adversary still has some power when selecting the probability distributions.

Theorem 6.2.2. *For any $\tau > 0$, no deterministic algorithm for* MINSET *under uncertainty has a competitive ratio better than $\frac{2}{\tau}$, even if all given sets are pairwise disjoint.*

Proof. Consider the same instance as in the proof of Theorem 6.2.1 but with different, now unknown distributions. Since the distributions are unknown, an algorithm cannot distinguish the intervals $I_1, \ldots, I_n$ even if they have different distributions. This means that the adversary still has some power and can set the distributions in a worst-case manner for the algorithm, as long as the distributions respect balancing parameter τ.

To that end, consider a fixed value τ and an arbitrary deterministic algorithm ALG. As ALG cannot distinguish the intervals $I_1, \ldots, I_n$, we can assume w.l.o.g. that it queries the intervals in order of their indices until the instance is solved. For all $0 < i < n$, the adversary sets the distribution to $d_i(a) = \tau$ for $a = 0.51$, $d_i(a) = (1-\tau)$ for $a = \epsilon$ and $d_i(a) = 0$ otherwise, for some infinitesimally small $\epsilon > 0$. Finally, the adversary sets distribution d_n to $d_n(a) = 1$ for $a = 0.7$ and $d_n(a) = 0$ otherwise. These distributions clearly respect the balancing parameter τ.

For these distributions, we always have $w(S_2) > w(S_1)$ as $w_n > w(S_1)$ holds with a probability of one. Thus, every algorithm has to query until the lower limit of set S_2 increases to a value of at least $w(S_1)$. For ALG, this is the case once it found two intervals I_i with $i < n$ and $w_i = 0.51$ or once it queries interval I_n in case no two such intervals exist. Thus, the expected query cost is $\mathbb{E}[\text{ALG}] \geq \min\{\frac{2}{\tau}, n\}$. The optimal solution on the other hand only queries I_n and is done after a single query. Therefore, $\mathbb{E}[\text{OPT}] = 1$ and the competitive ratio of ALG is at least $\min\{\frac{2}{\tau}, n\}$. We can conclude the theorem by picking a sufficiently large value for n. ☐

We continue by giving a quite simple algorithm for disjoint MINSET that matches the lower bound of Theorem 6.2.2.

In disjoint MINSET, each I_i occurs in exactly one constraint for one set S in the corresponding (MINSETIP). Thus, each set S defines a disjoint subproblem and the optimal solution OPT of the instance is the union of optimal solutions for the subproblems. The optimal solution for a subproblem S is to query the elements I_i of S in order of non-decreasing $(w_i - L_i)$ until the sum of those coefficients is at least $(w^* - L_S)$.

Under uncertainty, we adapt this strategy and query in order of non-decreasing $(U_i - L_i)$. While this does not guarantee that we query the interval with maximum $(w_j - L_j)$ in S, it gives us a probability of τ to query an interval I_i such that $(w_i - L_i)$ is at least half the maximum $(w_j - L_j)$. We will prove that this is sufficient to achieve the guarantee. Since we do not know w^*, we do not know when to stop querying in a subproblem. We handle this by only querying in the set S of minimum current lower limit as the subproblem for this set is clearly not yet solved. Algorithm 18 formalizes this approach.

Theorem 6.2.3. *There is an algorithm for disjoint* MINSET *under uncertainty with competitive ratio at most $\frac{2}{\tau}$.*

Proof. Consider a fixed realization of weights w_i and the corresponding (MINSETIP) instance. For each $S \in \mathcal{S}$, a feasible solution Q must satisfy $\sum_{I_i \in S \cap Q}(w_i - L_i) \geq (w^* - L_S)$. This implies $|Q \cap S| \geq |P^*_S|$ for the minimum cardinality prefix P^*_S of $I_1, \ldots, I_k$ that satisfies $\sum_{I_i \in P^*_S}(w_i - L_i) \geq w^* - L_S$, where $S = \{I_1, \ldots, I_k\}$ and all I_i are indexed by non-increasing $w_i - L_i$. As the sets are disjoint, we get $\text{OPT} = \sum_{S \in \mathcal{S}} |P^*_S|$.

Using this, we show that Algorithm 18 satisfies the theorem. To this end, let X_j be a random variable denoting the number of queries in iteration j of the outer while-loop of

Algorithm 18: Algorithm for disjoint MinSet under uncertainty.

```
   Input: Instance of MinSet under uncertainty with pairwise disjoint sets.
1  $Q \leftarrow \emptyset$;
2  while the problem is not solved do
3      $S_{\min} \leftarrow \arg\min_{S \in \mathcal{S}} L_S(Q)$;
4      repeat
5          $I_i \leftarrow \arg\max_{I_j \in S_{\min} \setminus Q} U_j - L_j$; Query $I_i$; $Q \leftarrow Q \cup \{I_i\}$;
6      until $w_i - L_i \geq \frac{1}{2} \cdot (U_i - L_i)$ or $S_{\min}$ has been completely queried;
```

Algorithm 18 and let Y_j be an indicator variable indicating whether iteration Y_j is actually executed ($Y_j = 1$) or not ($Y_j = 0$).

We prove the theorem by separately showing $\sum_j \mathbb{P}[Y_j = 1] \leq 2 \cdot \mathbb{E}[\text{OPT}]$ and $\mathbb{E}[X_i \mid Y_j = 1] \leq \frac{1}{\tau}$. Since

$$\begin{aligned}\mathbb{E}[\text{ALG}] &= \sum_j \mathbb{E}[X_j] = \sum_j \mathbb{P}[Y_j = 0]\, \mathbb{E}[X_j \mid Y_j = 0] + \sum_j \mathbb{P}[Y_j = 1]\, \mathbb{E}[X_j \mid Y_j = 1] \\ &= \sum_j \mathbb{P}[Y_j = 1]\, \mathbb{E}[X_j \mid Y_j = 1]\end{aligned}$$

follows from $\mathbb{E}[X_j \mid Y_j = 0] = 0$ and the law of total expectations, the two inequalities imply the theorem.

Note that $\sum_j \mathbb{P}[Y_j = 1]$ is just the expected number of iterations of the algorithm. Thus, if we show for each realization of precise weights that the number of iterations is at most $2 \cdot \text{OPT}$, we directly get $\sum_j \mathbb{P}[Y_j = 1] \leq 2 \cdot \mathbb{E}[\text{OPT}]$.

Consider a fixed realization. For each S, let h_S denote the number of iterations with $S_{\min} = S$. We claim that $h_S \leq 2 \cdot |P^*_S|$. Then, $\text{OPT} \geq \sum_{S \in \mathcal{S}} |P^*_S|$ implies $\sum_j \mathbb{P}[Y_j = 1] \leq 2 \cdot \mathbb{E}[\text{OPT}]$.

Let j be an iteration with $S_{\min} = S$, let G_j denote the queries of this iteration and let Q_j denote the set of all previous queries. Observe that $P^*_S \setminus Q_j \neq \emptyset$. Otherwise, the definition of P^*_S would either imply that the lower limit of S after querying Q_j is larger than w^*, which contradicts $S_{\min} = S$, or that the lower limit is equal to w^*, which implies that the problem is already solved.

We argue that we have $\sum_{I_i \in G_j} w_i - L_i \geq \frac{1}{2} \cdot \max_{I_i \in S \setminus Q_j} w_i - L_i$. Intuitively, this inequality means that ALG in each iteration with $S_{\min} = S$ increases L_S by at least half as much as even OPT could. In case that interval $I_i = \arg\max_{I_i \in S \setminus Q_j} w_i - L_i$ is contained in G_j, the inequality clearly holds. Otherwise, let $I_{i'}$ denote the last element that is queried in the iteration. Then, $w_{i'} - L_{i'} \geq \frac{1}{2} \cdot (U_{i'} - L_{i'}) \geq \frac{1}{2} \cdot \max_{I_i \in S \setminus Q_j} w_i - L_i$, where the first inequality holds as $I_{i'}$ is the last query of the iteration and the second inequality holds by the order in which the elements of S are queried by the algorithm. Thus, this last interval $I_{i'}$ alone satisfies the inequality.

The inequality suffices to conclude that $\sum_{I_i \in Q_j \cap S} w_i - L_i \geq \sum_{I_i \in P^*_S} w_i - L_i$ holds after at most $2 \cdot |P^*_S|$ iterations with $S_{\min} = S$. As $S_{\min} \neq S$ holds for all following iterations, the claim $h_S \leq 2 \cdot |P^*_S|$ follows.

Next, we show the second inequality, $\mathbb{E}[X_j \mid Y_j = 1] \leq \frac{1}{\tau}$. If an iteration j of the outer while-loop is executed ($Y_j = 1$), the repeat statement queries intervals I_i until either $(w_i - L_i) \geq \frac{1}{2} \cdot (U_i - L_i)$ or $S_{\min} \subseteq Q$. Thus, it terminates at the latest when it finds an I_i with $(w_i - L_i) \geq \frac{1}{2} \cdot (U_i - L_i)$. The number of queries until such an interval occurs is described by a geometric distribution with success probability at least τ. So, in expectation, this number is at most $\frac{1}{\tau}$ and we can conclude $\mathbb{E}[X_j \mid Y_j = 1] \leq \frac{1}{\tau}$. □

We remark that Theorems 6.2.2 and 6.2.3 imply that, even with full knowledge of the distributions, the competitive ratio for disjoint MinSet cannot be improved by more than a factor of two compared to the ratio with unknown distributions.

6.3 Algorithmic framework

In the previous section, we have seen an algorithm for disjoint MinSet under uncertainty with a tight competitive ratio. The key observation that allowed us to achieve that ratio was the simple characterization of an (offline) optimal solution. In this section, we consider the offline variant of the general MinSet and give inapproximability results that prevent such simple characterizations for optimal solutions of the general problem. Thus, we need alternative algorithms and, based on observations for the offline problem, present an algorithmic framework that can be used to solve MinSet under uncertainty.

6.3.1 Offline Problems and Hardness of Approximation

We refer to the problem of solving (MinSetIP) with full knowledge of the precise weights w_i (and w^*) as *offline* problem. This means that we have full knowledge of all coefficients of the ILP. For MinSet under uncertainty, we say that a solution is optimal, if it is an optimal solution for the corresponding offline problem. We use OPT to refer to an optimal solution and, slightly abusing the notation, to its objective value.

Offline MinSet contains the classical SetCover problem and, thus, it is as hard to approximate. This result transfers to the stochastic setting, even for uniform distributions. Results by Dinur and Steurer [DS14] imply the following, as we formally prove in Section 6.3.3.

Theorem 6.3.1. *For any fixed $\alpha > 0$, it is NP-hard to compute a query strategy that is $(1-\alpha)\cdot\ln(m-1)$-competitive for* MinSet *under uncertainty even if the precise weight w_i of each I_i is drawn independently and uniformly at random from (L_i, U_i). The same inapproximability holds also for offline* MinSet.

On the positive side, we can approximate offline MinSet by adapting covering results (see, e.g.,[Chv79; Dob82; RV98; KY01; KY05]). In particular, we want to use greedy algorithms that iteratively and irrevocably add elements to the solution that are selected by a certain greedy criterion. Recall that "adding an element to the solution" corresponds to both, setting the variable x_i of an interval $I_i \in \mathcal{I}$ in (MinSetIP) to one and querying I_i. While we are technically not restricted to greedy algorithms when solving *offline* MinSet, our goal is to later on generalize the offline algorithm to the setting with uncertainty and irrevocable decisions. Hence, greedy algorithms seem to be a suitable choice.

Since the greedy criterion for adding an element depends on previously added elements, we define a version of (MinSetIP) that is parametrized by the set $Q \subseteq \mathcal{I}$ of elements that have already been added to the solution and adjust the right-hand sides to the remaining covering requirement after adding Q. Recall that $a_i = w_i - L_i$ and $b_S = w^* - L_S$. Here, $b_S(Q) = \max\{b_S - \sum_{I_i \in Q \cap S} a_i, 0\}$ and $b(Q) = \sum_{S \in \mathcal{S}} b_S(Q)$.

$$\begin{array}{llll} \min & \sum_{I_i \in \mathcal{I}\setminus Q} x_i & & \\ \text{s.t.} & \sum_{I_i \in S\setminus Q} x_i \cdot a_i & \geq b_S(Q) & \forall S \in \mathcal{S} \\ & x_i & \in \{0,1\} & \forall I_i \in \mathcal{I}\setminus Q \end{array} \qquad \text{(MinSetIP-Q)}$$

Based on this ILP, we adjust the algorithm by Dobson [Dob82] for the multiset multicover problem to our setting (cf. Algorithm 19). The algorithm scales the coefficients such that all non-zero left-hand side coefficients are at least 1. We refer to such instances as *scaled*. Then it greedily adds the element to the solution that reduces the right-hand sides the most, i.e.,

Algorithm 19: Greedy algorithm by Dobson [Dob82] for offline MinSet.

```
Input: An instance of offline MinSet, i.e., an instance of (MinSetIP)
1 $s_{\min} = \min_{I_i \in \mathcal{I}\colon a_i > 0} a_i$; $\forall S \in \mathcal{S}\colon b'_S = \frac{b_S}{s_{\min}}$; $\forall I_i \in \mathcal{I}\colon a'_i = \frac{a_i}{s_{\min}}$;
2 while $\exists S \in \mathcal{S}\colon b'_S(Q) \geq 1$ do
3     $I_i \leftarrow \arg\max_{I_j \in \mathcal{I} \setminus Q} g_c(Q, I_j)$; Query $I_i$; $Q \leftarrow Q \cup \{I_i\}$;
4 while the problem is not solved do
5     $I_i \leftarrow \arg\max_{I_j \in \mathcal{I} \setminus Q} g_s(Q, I_j)$; Query $I_i$; $Q \leftarrow Q \cup \{I_i\}$;
```

the interval $I_i \in \mathcal{I} \setminus Q$ that maximizes $g_c(Q, I_i) = b'(Q) - b'(Q \cup \{I_i\})$ (a' and b' indicate scaled coefficients). For a subset $G \subseteq \mathcal{I}$, we define $g_c(Q, G) = b'(Q) - b'(Q \cup G)$.

After $b'_S(Q) < 1$ for all $S \in \mathcal{S}$, we can exploit that all scaled non-zero coefficients a'_i are at least one. This means that adding an element $I_i \in \mathcal{I} \setminus Q$ satisfies all remaining constraints of sets S with $I_i \in S$. Thus, the remaining problem reduces to a SetCover instance, which can be solved by using the classical greedy algorithm by Chvatal [Chv79]. This algorithm greedily adds the element $I_i \in \mathcal{I} \setminus Q$ that maximizes $g_s(Q, I_i) = A(Q) - A(Q \cup \{I_i\})$ with $A(Q) = |\{S \in \mathcal{S} \mid b'_S(Q) > 0\}|$, i.e., the element that satisfies the largest number of constraints that are not already satisfied by Q. For a subset $G \subseteq \mathcal{I}$, we define $g_s(Q, G) = A(Q) - A(Q \cup G)$.

During the course of this chapter, we refer to $g_c(Q, I_i)$, $g_s(Q, I_i)$, $g_c(Q, G)$ and $g_s(Q, G)$ as the *greedy values* of I_i and G, respectively.

Theorem 6.3.2 (Follows from Dobson [Dob82])**.** *Algorithm 19 is a polynomial-time $\mathcal{O}(\log m)$-approximation for offline* MinSet. *The precise approximation factor is $\rho(\gamma) = \lceil \ln(\gamma \cdot m \cdot \max_S(w^* - L_S)) \rceil + \lceil \ln(m) \rceil$ with $s_{\min} = \min_{I_i \in \mathcal{I}\colon (w_i - L_i) > 0}(w_i - L_i)$, $\gamma = 1/s_{\min}$ and $m = |\mathcal{S}|$.*

During the remaining course of the chapter, we will state the competitive ratios of our algorithms in terms of ρ. To that end, define $\bar{\rho}(\gamma) = \lceil \ln(\gamma \cdot m \cdot \max_{S,S'}(U_S - L_{S'}) \rceil + \lceil \ln(m) \rceil$, which is an upper bound on $\rho(\gamma)$. Under uncertainty, we compare against $\bar{\rho}$ to avoid the random variable w^*. For constant U_i's, $\bar{\rho}$ and ρ are asymptotically the same.

We remark again that the approximation ratio of Algorithm 19 has dependencies on the numerical input parameter $s_{\min}$ and $\max_S(w^* - L_S)$. While there exist algorithms that achieve an approximation ratio of $\mathcal{O}(\log m)$ for the offline problem without such dependencies [KY01; KY05], these algorithms are not greedy and it remains open whether there exist algorithms with this improved ratio that execute irrevocable decisions, even with full knowledge of the coefficients. Thus, we consider Algorithm 19 and aim at extending it for the setting under uncertainty.

6.3.2 Algorithmic framework

We introduce our algorithmic framework that we use to solve MinSet under uncertainty. Ideally, we would like to apply the offline greedy algorithm. However, since the coefficients $a_i = w_i - L_i$ and $b_S = w^* - L_S$ are unknown, we cannot apply Algorithm 19 to solve MinSet under uncertainty as we cannot compute the element that maximizes the greedy value g_c or g_s.

While we cannot precisely compute the greedy choice, our strategy is to *approximate* it and to show that approximating the greedy choice is sufficient to obtain the desired guarantees. To make this more precise, consider an *iterative* algorithm for (MinSetIP), i.e., an algorithm that iteratively adds pairwise disjoint subsets $G_1, \ldots, G_h$ of $\mathcal{I}$ to the solution. For each j, let $Q_j = \bigcup_{1 \leq j' \leq j-1} G_{j'}$, i.e., Q_j contains the elements that have been added to the solution before G_j. If the combined greedy value of G_j is within a factor of α to the best greedy value for the problem instance *after* adding Q_j, then we say that G_j α-approximates the greedy choice. The following technical definition makes this more precise and the subsequent lemma connects the definition to the actual greedy values while taking into account that there are two different greedy values g_c and g_s (cf. Algorithm 19).

Definition 6.3.3. *For a* (MINSETIP) *instance with scaled coefficients and optimal solution* OPT*, let* $\alpha \in \mathbb{R}_{\geq 1}$ *and consider the corresponding instance of* (MINSETIP-Q) *for some* $Q \subseteq \mathcal{I}$*. A set* $G \subseteq \mathcal{I} \setminus Q$ α-approximates *the current greedy choice after adding* Q *if either*

1. $A(Q \cup G) \leq (1 - \frac{1}{\alpha \cdot \mathrm{OPT}}) \cdot A(Q)$ *or*
2. $b'(Q) \geq 1$ *and* $b'(Q \cup G) \leq (1 - \frac{1}{\alpha \cdot \mathrm{OPT}}) \cdot b'(Q)$.

Intuitively, the two conditions of the following lemma seem like a more appropriate definition of approximating a greedy choice. While the conditions of the lemma imply that the definition above is satisfied, in our proofs it will sometimes be easier to directly show that the definition is satisfied, without using the lemma. Therefore, we use the more technical Definition 6.3.3 but the lemma captures the intuition behind the definition.

Lemma 6.3.4. *For a scaled instance of* (MINSETIP)*,* $Q \subseteq \mathcal{I}$*,* $\alpha \geq 1$ *and* $G \subseteq \mathcal{I} \setminus Q$*:*

1. *If* $b'_S(Q) < 1$ *for all* $S \in \mathcal{S}$ *and* $g_s(Q, G) \geq \frac{1}{\alpha} \cdot \max_{I_i \in \mathcal{I} \setminus Q} g_s(Q, I_i)$*, then* G *satisfies the first condition of Definition 6.3.3 and, thus,* α*-approximates the greedy choice.*
2. *If* $b'(Q) \geq 1$ *and* $g_c(Q, G) \geq \frac{1}{\alpha} \cdot \max_{I_i \in \mathcal{I} \setminus Q} g_c(Q, I_i)$*, then* G *satisfies the second condition of Definition 6.3.3 and, thus,* α*-approximates the greedy choice. This holds even if some non-zero coefficients* a'_i *are smaller than* 1.

Proof. First, assume that $b'_S(Q) < 1$ for all $S \in \mathcal{S}$ and consider a set $G \subseteq \mathcal{I} \setminus Q$ with $g_s(Q, G) \geq \frac{1}{\alpha} \cdot \max_{I_i \in \mathcal{I} \setminus Q} g_s(Q, I_i)$.

Let $I^* = \arg\max_{I_i \in \mathcal{I} \setminus Q} A(Q) - A(Q \cup \{I_i\}) = \arg\max_{I_i \in \mathcal{I} \setminus Q} g_s(Q, I_i)$. By assumption $b'_S(Q) < 1$ for all $S \in \mathcal{S}$ and, as we consider a scaled instance, $a'_i \geq 1$ for all $I_i \in \mathcal{I}$. Thus, the remaining instance is a set cover instance as adding an interval I_i to the solution satisfies all constraints S with $I_i \in S$ that have not already been satisfied by Q.

Let OPT_Q denote the optimal solution for the remaining instance after adding Q to the solution, i.e., the optimal solution to (MINSETIP-Q). Using a standard set cover argument, we can observe that $\frac{A(Q)}{\mathrm{OPT}_Q} \leq A(Q) - A(Q \cup \{I^*\})$ as the optimal solution satisfies the remaining constraints at cost OPT_Q, but a single interval can satisfy at most $A(Q) - A(Q \cup \{I^*\})$ constraints. Note that this argument only holds because all left-hand side coefficients are at least as large as the right-hand sides. Otherwise, adding an interval I_i later, i.e., after $Q' \supset Q$ has already been added to the solution, could satisfy more constraints, i.e., $A(Q) - A(Q \cup \{I_i\}) < A(Q') - A(Q' \cup \{I_i\})$. This is one of the reasons why the offline greedy algorithm uses two greedy criteria.

By assumption and definition of g_s, we have $\alpha \cdot (A(Q) - A(Q \cup G)) \geq \max_{I_i \in \mathcal{I} \setminus Q} A(Q) - A(Q \cup \{I_i\})$ and, therefore, $\frac{A(Q)}{\mathrm{OPT}_Q} \leq \alpha \cdot (A(Q) - A(Q \cup G))$. Rearranging the latter inequality, we obtain $A(Q \cup G) \leq A(Q) \cdot \left(1 - \frac{1}{\alpha \mathrm{OPT}_Q}\right)$. Since $\mathrm{OPT} \geq \mathrm{OPT}_Q$ for the optimal solution OPT of the complete instance, we get $A(Q \cup G) \leq A(Q) \cdot \left(1 - \frac{1}{\alpha \mathrm{OPT}}\right)$. This implies that G satisfies the first condition of Definition 6.3.3.

For the second part of the lemma, assume $b'(Q) \geq 1$ and consider a set $G \subseteq \mathcal{I} \setminus Q$ with $g_c(Q, G) \geq \frac{1}{\alpha} \cdot \max_{I_i \in \mathcal{I} \setminus Q} g_c(Q, I_i)$.

Let $I^* = \arg\max_{I_i \in \mathcal{I} \setminus Q} b'(Q) - b'(Q \cup \{I_i\}) = \arg\max_{I_i \in \mathcal{I} \setminus Q} g_c(Q, I_i)$. Observe that $\frac{b'(Q)}{\mathrm{OPT}_Q} \leq b'(Q) - b'(Q \cup \{I^*\})$ as the optimal solution covers the remaining constraints at cost OPT_Q, but a single interval can decrease the total slack between left-hand and right-hand sides of (MINSETIP-Q) by at most $b'(Q) - b'(Q \cup \{I^*\})$. By assumption and definition of g_c, we have $\alpha \cdot (b'(Q) - b'(Q \cup G)) \geq \max_{I_i \in \mathcal{I} \setminus Q} b'(Q) - b'(Q \cup \{I_i\})$ and, therefore, $\frac{b'(Q)}{\mathrm{OPT}_Q} \leq \alpha \cdot (b'(Q) - b'(Q \cup G))$. Rearranging the latter inequality, we obtain $b'(Q \cup G) \leq b'(Q) \cdot \left(1 - \frac{1}{\alpha \mathrm{OPT}_Q}\right)$. Since $\mathrm{OPT} \geq \mathrm{OPT}_Q$ for the optimal solution OPT of the complete instance, we get $b'(Q \cup G) \leq b'(Q) \cdot (1 - \frac{1}{\alpha \mathrm{OPT}})$. This and the assumption $b'(Q) \geq 1$ imply that G satisfies the second condition of Definition 6.3.3. Note that the argument for the second case does not use that all non-zero coefficients are at least one. Thus, the statement also holds if there are coefficients $0 < a_i' < 1$. □

With the following lemma, we bound the number of iterations j in which G_j α-approximates the current greedy choice via an adjusted set cover greedy analysis.

Lemma 6.3.5. *Consider an arbitrary algorithm for* (MINSETIP) *that scales the coefficients by factor γ and iteratively adds disjoint subsets $G_1, \dots, G_h$ of $\mathcal{I}$ to the solution until the instance is solved. The number of groups G_j that α-approximate the current greedy choice (after adding $Q_j = \bigcup_{1 \leq j' \leq j-1} G_{j'}$) is at most $\alpha \cdot \rho(\gamma) \cdot \mathrm{OPT}$.*

Proof. We first show that the number of iterations j with $b'(Q_j) \geq 1$ and $b'(Q_j \cup G_j) \leq (1 - \frac{1}{\alpha \cdot \mathrm{OPT}}) \cdot b'(Q_j)$, i.e., the number of iterations that satisfy the second condition of Definition 6.3.3, is at most $\alpha \lceil \ln(\gamma \cdot m \cdot \max_S(w^* - L_S)) \rceil \cdot \mathrm{OPT}$.

Let $\bar{G}_1, \dots, \bar{G}_k \subseteq \mathcal{I}$ denote the sets that are added to the solution by the algorithm *and* satisfy the second condition of Definition 6.3.3. Assume that the sets are indexed in the order they are added. For each $j \in \{1, \dots, k\}$, let $\bar{Q}_j \subseteq \mathcal{I}$ denote the set of intervals that are added to the solution *before* $\bar{G}_j$. Note that $\{\bar{G}_1, \dots, \bar{G}_{j-1}\} \subseteq \bar{Q}_j$, but $\bar{Q}_j$ might contain additional added groups that just do not satisfy the second condition of Definition 6.3.3.

By assumption, $b'(\bar{Q}_j \cup \bar{G}_j) \leq (1 - \frac{1}{\alpha \cdot \mathrm{OPT}}) \cdot b'(\bar{Q}_j)$. A recursive application of this inequality and the fact that $(1 - x) < e^{-x}$ for all $x \in \mathbb{R} \setminus \{0\}$ implies

$$b'(\bar{Q}_j \cup \bar{G}_j) \leq b'(\emptyset) \cdot \left(1 - \frac{1}{\alpha \mathrm{OPT}}\right)^j < b'(\emptyset) \cdot e^{-\frac{j}{\alpha \mathrm{OPT}}}.$$

Thus, after $j = \alpha \cdot \mathrm{OPT} \cdot \lceil \ln b'(\emptyset) \rceil$ iterations that satisfy the second condition of Definition 6.3.3, we have $b'(\bar{Q}_j \cup \bar{G}_j) < b'(\emptyset) \cdot e^{-\ln b'(\emptyset)} = 1$. But if $b'(\bar{Q}_j \cup \bar{G}_j) < 1$, then there can be no further iteration that satisfies the second condition of Definition 6.3.3. Thus, the number of such iterations is at most $\alpha \cdot \mathrm{OPT} \cdot \lceil \ln b'(\emptyset) \rceil$. Since $b'(\emptyset)$ is upper bounded by $\gamma \cdot m \cdot \max_S(w^* - L_S)$ as we have m constraints with scaled right-hand side values of at most $\gamma \cdot \max_S(w^* - L_S)$, the number of such iterations is at most $\alpha \cdot \lceil \ln(\gamma \cdot m \cdot \max_S(w^* - L_S) \rceil \cdot \mathrm{OPT}$.

Next, we show that the number of iterations j with $A(Q_j \cup G_j) \leq (1 - \frac{1}{\alpha \cdot \mathrm{OPT}}) \cdot A(Q_j)$, i.e., the number of iterations that satisfy the first condition of Definition 6.3.3, is at most $\alpha \lceil \ln(m) \rceil \cdot \mathrm{OPT}$. The proof is essentially a copy of the previous case.

Let $\bar{G}_1, \dots, \bar{G}_k \subseteq \mathcal{I}$ denote the sets that are added to the solution by the algorithm and satisfy the first condition of Definition 6.3.3. Assume that the sets are indexed in the order they are added. For each $j \in \{1, \dots, k\}$, let $\bar{Q}_j \subseteq \mathcal{I}$ again denote the set of intervals that are added to the solution *before* $\bar{G}_j$. Note that $\{\bar{G}_1, \dots, \bar{G}_{j-1}\} \subseteq \bar{Q}_j$, but $\bar{Q}_j$ might contain additional sets that just do not satisfy the first condition of Definition 6.3.3.

By assumption, $A(\bar{Q}_j \cup \bar{G}_j) \leq (1 - \frac{1}{\alpha \cdot \mathrm{OPT}}) \cdot A(\bar{Q}_j)$. A recursive application of this inequality and the fact that $(1 - x) < e^{-x}$ for all $x \in \mathbb{R} \setminus \{0\}$ implies

$$A(\bar{Q}_j \cup \bar{G}_j) \leq A(\emptyset) \cdot \left(1 - \frac{1}{\alpha \mathrm{OPT}}\right)^j < A(\emptyset) \cdot e^{-\frac{j}{\alpha \mathrm{OPT}}}.$$

Thus, after $j = \alpha \cdot \mathrm{OPT} \cdot \lceil \ln A(\emptyset) \rceil$ such iterations, we have $A(\bar{Q}_j \cup \bar{G}_j) < A(\emptyset) \cdot e^{-\ln A(\emptyset)} = 1$. But if $A(\bar{Q}_j \cup \bar{G}_j) < 1$, then $A(\bar{Q}_j \cup \bar{G}_j) = 0$ and the instance is solved and no further iteration is executed. Since $A(\emptyset)$ is upper bounded by the number of constraints m, the number of iterations that satisfy the first condition of Definition 6.3.3 is at most $\alpha \lceil \ln(m) \rceil \cdot \mathrm{OPT}$.

In total, at most $\alpha \lceil \ln(m) \rceil \cdot \mathrm{OPT}$ iterations satisfy the first condition of Definition 6.3.3 and at most $\alpha \cdot \lceil \ln(\gamma \cdot m \cdot \max_S(w^* - L_S)) \rceil \cdot \mathrm{OPT}$ iterations satisfy the second condition of Definition 6.3.3. In summation, there are at most $\alpha \cdot (\lceil \ln(\gamma \cdot n \cdot \max_{e \in E} b_e) \rceil + \lceil \ln(n) \rceil) \cdot \mathrm{OPT} = \alpha \cdot \rho(\gamma) \cdot \mathrm{OPT}$ iterations that satisfy Definition 6.3.3. □

The lemma states that the number of groups G_j that α-approximate their greedy choice is within a factor of α of the performance guarantee $\rho(\gamma)$ of the offline greedy algorithm. If each G_j α-approximates its greedy choice, the iterative algorithm achieves an approximation factor of $\max_j |G_j| \cdot \alpha \cdot \rho(\gamma)$. Thus, approximating the greedy choices by a constant factor using a constant group size is sufficient to only lose a constant factor compared to the offline greedy algorithm.

This insight gives us a framework to solve MinSet under uncertainty. Recall that the w_i's (and by extension the a_i's and b_S's) are uncertain and only revealed once we irrevocably add an $I_i \in \mathcal{I}$ to the solution. We refer to a revealed w_i as a *query result*, and to a fixed set of revealed w_i's for all $I_i \in \mathcal{I}$ as a *realization of query results*. Consider an iterative algorithm. The sets G_j can be computed and queried adaptively and are allowed to depend on (random) query results from previous iterations. Hence, $X_j = |G_j|$ is a random variable. Let Y_j be an indicator variable denoting whether the algorithm executes iteration j ($Y_j = 1$) or terminates beforehand ($Y_j = 0$). We define the following class of iterative algorithms and show that algorithms from this class achieve certain guarantees.

Definition 6.3.6. *An iterative algorithm is* (α, β, γ)-Greedy *if it satisfies:*

1. *For every realization of query results; each G_j α-approximates the greedy choice after querying Q_j for the instance with coefficients scaled by γ.*
2. *$\mathbb{E}[X_j \mid Y_j = 1] \leq \beta$ holds for all iterations j.*

Theorem 6.3.7. *Each* (α, β, γ)-Greedy *algorithm for* MinSet *under uncertainty achieves a competitive ratio of $\alpha \cdot \beta \cdot \bar{\rho}(\gamma) \in \mathcal{O}(\alpha \cdot \beta \cdot \log(m))$.*

Proof. Consider an (α, β, γ)-Greedy algorithm ALG for MinSet. The expected cost of ALG is $\mathbb{E}[\mathrm{ALG}] = \sum_j \mathbb{E}[X_j]$. Using the total law of expectations, we get

$$\begin{aligned}\mathbb{E}[\mathrm{ALG}] &= \sum_j \mathbb{P}[Y_j = 1]\, \mathbb{E}[X_j \mid Y_j = 1] + \mathbb{P}[Y_j = 0]\, \mathbb{E}[X_j \mid Y_j = 0] \\ &= \sum_j \mathbb{P}[Y_j = 1]\, \mathbb{E}[X_j \mid Y_j = 1],\end{aligned}$$

where the last inequality holds because $\mathbb{E}[X_j \mid Y_j = 0] = 0$ (if the algorithm terminates before iteration j, then it adds no more elements to the solution and, thus, $X_j = 0$). By the second property of Definition 6.3.6, this implies $\mathbb{E}[\mathrm{ALG}] \leq \beta \cdot \sum_j \mathbb{P}[Y_j = 1]$.

Thus, it remains to bound $\sum_j \mathbb{P}[Y_j = 1]$, which corresponds to the expected number of iterations of ALG. Consider a fixed realization of query results, then, by the first property of (α, β, γ)-Greedy, each G_j α-approximates its greedy choice for the (MinSetIP) instance of the realization scaled by factor γ. Then, Lemma 6.3.5 implies that the number of iterations is at most $\alpha \cdot \rho(\gamma) \cdot \mathrm{OPT}$, which is upper bounded by $\alpha \cdot \bar{\rho}(\gamma) \cdot \mathrm{OPT}$. As this upper bound on the number of iterations holds for every realization and OPT is the only random variable of that term (since we substituted ρ with $\bar{\rho}$), we can conclude $\sum_j \mathbb{P}[Y_j = 1] \leq \alpha \cdot \bar{\rho}(\gamma) \cdot \mathbb{E}[\mathrm{OPT}]$, which implies $\mathbb{E}[\mathrm{ALG}] \leq \alpha \cdot \beta \cdot \bar{\rho}(\gamma) \cdot \mathbb{E}[\mathrm{OPT}]$. □

6.3.3 Proof of the Hardness of Approximation

In the following, we show that (MINSETIP) is not only a special case of the multiset multicover problem but also contains the hard instances of this problem.

Erlebach et al. [EHK16] showed that offline MINSET, for the problem variant where it is not necessary to compute the weight w^*, is NP-hard via reduction from vertex cover. In the reduction by [EHK16] all intervals of the set S^* with $w(S^*) = w^*$ are trivial and, thus, the result translates to the problem variant where one has to compute w^*. In the following, we strengthen this result by showing that the offline problem is as hard to approximate as SETCOVER.

In SETCOVER, we are given a set of elements $U = \{1, \ldots, n\}$ and a family of sets $\bar{\mathcal{S}} = \{\bar{S}_1, \ldots, \bar{S}_m\}$ with $\bar{S}_j \subseteq U$. The goal is to find a subset $H \subseteq \bar{\mathcal{S}}$ of minimum cardinality such that $\bigcup_{\bar{S}_j \in H} \bar{S}_j = U$.

Theorem 6.3.8. *There is an approximation-factor preserving reduction from* SETCOVER *to offline* MINSET.

Proof. Given an instance $(U, \bar{\mathcal{S}})$ of SETCOVER, we construct an offline MINSET instance as follows:

1. Add a trivial interval $I_r = \{w_r\}$.
2. Add a single set $C = \{I_r\}$.
3. For each $j \in U$, add a set S_j.
4. For each $\bar{S}_i \in \bar{\mathcal{S}}$:
   (a) Add an interval $I_i = (L_i, U_i)$ with $L_i = 0$, $U_i = w_r + \delta$ and $w_i = w_r + \epsilon$ for a common $\delta > \epsilon > 0$ and some infinitesimally small $\epsilon > 0$.
   (b) For each $j \in \bar{S}_i$, add interval I_i to set S_j.

This reduction clearly runs in polynomial time. To finish the proof, we show the following claim: *There is a* SETCOVER *solution H of cardinality k if and only if there is a feasible query set Q for the constructed offline* MINSET *instance with $|Q| = k$.*

By definition of the constructed instance, set $C = \{I_r\}$ is the set of minimum weight $w^* = w_r$. Each feasible query set Q for the offline MINSET instance must prove that $L_S(Q) \geq w_r$ holds for each $S \in \mathcal{S} \setminus \{C\}$. Recall that $L_S(Q)$ is the lower limit of S after querying Q. By definition of the constructed intervals and sets, a query set Q is feasible if and only if $|Q \cap S| \geq 1$ for each $S \in \mathcal{S} \setminus \{C\}$, i.e., Q has to contain at least one element of each $S \in \mathcal{S} \setminus \{C\}$.

For the first direction, consider an arbitrary set cover H for the given SETCOVER instance and construct $Q = \{I_i \mid \bar{S}_i \in H\}$. Clearly $|Q| = |H|$. Since H is a set cover, each $j \in U$ is contained in at least one $\bar{S}_i \in H$. If $j \in U$ is contained in $\bar{S}_i$, then, by construction, I_i is contained in S_j. Thus, as H covers all elements $j \in U$, set Q contains at least one member of each $S_j \in \mathcal{S} \setminus \{C\}$ and, therefore, is a feasible query set.

For the second direction, consider an arbitrary feasible query set Q of the constructed instance and construct $H = \{\bar{S}_i \mid I_i \in Q\}$. Clearly, $|Q| = |H|$. Since Q is feasible, it contains at least one member of each $S_j \in \mathcal{S} \setminus \{C\}$. If $I_i \in S_j$ is contained in Q, then, by construction, set $\bar{S}_i \in H$ covers element j. As Q contains at least one member of each $S_j \in \mathcal{S} \setminus \{C\}$, it follows that H covers U. □

Dinur and Steurer [DS14] showed that it is NP-hard to approximate SETCOVER within a factor of $(1 - \alpha) \cdot \ln n$ for any $\alpha > 0$, where n is the number of elements in the instance,

Algorithm 20: MINSET with deterministic right-hand sides.

```
Input: Instance of MINSET with deterministic right-hand sides.
1 $Q = \emptyset$; Scale $a$ and $b$ by $\frac{2}{s_{\min}}$ to $a'$ and $b'$ for $s_{\min} = \min_{I_i \in \mathcal{I}\colon U_i - L_i > 0} U_i - L_i$;
2 while the problem is not solved do
3     if $b'(Q) \geq 1$ then $g = \bar{g}_c$ else $g = \bar{g}_s$;
4     repeat
5         $I_i \leftarrow \arg\max_{I_j \in \mathcal{I} \setminus Q} g(Q, I_j)$; Query $I_i$; $Q \leftarrow Q \cup \{I_i\}$;
6     until the problem is solved or $w_i - L_i \geq \frac{1}{2} \cdot (U_i - L_i)$;
```

via a reduction running in time $n^{1/\alpha}$. Consider the construction of Theorem 6.3.8. Since the sets in the constructed offline MINSET instance correspond to the elements in the input SETCOVER instance, the construction implies the following corollary. Note that the corollary uses $\ln(m-1)$ instead if $\ln(m)$ because the reduction introduces the extra set C.

Corollary 6.3.9. *For every $\alpha > 0$, it is NP-hard to approximate offline* MINSET *within a factor of $(1-\alpha) \cdot \ln(m-1)$, where $m = |\mathcal{S}|$ is the number of sets. The reduction runs in time $m^{1/\alpha}$.*

We show that Theorem 6.3.8 and Corollary 6.3.9 apply also to MINSET under stochastic explorable uncertainty, even if the precise weight w_i of each I_i is drawn independently and uniformly at random from (L_i, U_i).

Theorem 6.3.1. *For any fixed $\alpha > 0$, it is NP-hard to compute a query strategy that is $(1-\alpha) \cdot \ln(m-1)$-competitive for* MINSET *under uncertainty even if the precise weight w_i of each I_i is drawn independently and uniformly at random from (L_i, U_i). The same inapproximability holds also for offline* MINSET.

Proof. The hardness of approximation for offline MINSET follows directly from Corollary 6.3.9.

We continue to show the statement on MINSET under uncertainty with uniform distributions. Consider the reduction of Theorem 6.3.8 with w_r towards 0 and/or δ towards ∞. With w_r running towards 0 and/or δ running towards ∞, the probability that it is sufficient to query one $I_i \in S_j$ to show that $w_r \leq L_{S_j}(Q)$, for some query set Q, goes towards 1. Thus, the probability that any set Q that contains at least one member of each $S \in \mathcal{S}$ is feasible goes towards one as well. Thus, $\lim_{w_r \to 0} \mathbb{E}[\mathrm{OPT}] = \lim_{\delta \to \infty} \mathbb{E}[\mathrm{OPT}] = |H^*|$, where H^* is the optimal solution for the input SETCOVER instance. Therefore, by Theorem 6.3.8, in order to be $((1-\alpha) \cdot \ln m)$-competitive, the query strategy has to compute an $((1-\alpha) \cdot \ln n)$-approximation for set cover. This implies NP-hardness. □

6.4 MINSET with Deterministic Right-Hand Sides

We consider a variant of MINSET under uncertainty, where the right-hand sides b_S of the ILP representation (MINSETIP) are deterministic and explicit part of the input. Thus, only the coefficients $a_i = (w_i - L_i)$ remain uncertain within the interval $(0, U_i - L_i)$. For this problem variant, it can happen that the instance has no feasible solution. In that case, we require every algorithm (including OPT) to reduce the covering requirements as much as possible. As we consider the stochastic problem variant, recall that the balancing parameter is defined as $\tau = \min_{I_i \in \mathcal{I}} \tau_i$ for $\tau_i = \mathbb{P}[w_i \geq \frac{U_i + L_i}{2}]$.

Theorem 6.4.1. *For $\tau > 0$. There is an algorithm for* MINSET *under uncertainty with deterministic right-hand sides and a competitive ratio of $\frac{2}{\tau} \cdot \rho(\gamma) \in \mathcal{O}(\frac{1}{\tau} \cdot \log m)$ with $\gamma = 2/s_{\min}$ for $s_{\min} = \min_{I_i \in \mathcal{I}\colon U_i - L_i > 0} U_i - L_i$.*

The algorithm of the theorem loses only a factor $\frac{2}{\tau}$ compared to the greedy approximation factor $\rho(\gamma)$ on the corresponding offline problem. We show the theorem by proving that Algorithm 20 is an (α, β, γ)-GREEDY algorithm for $\alpha = 2$, $\beta = \frac{1}{\tau}$ and $\gamma = \frac{2}{s_{\min}}$ with $s_{\min} = \min_{I_i \in \mathcal{I} \colon U_i - L_i > 0} U_i - L_i$. Then, Theorem 6.3.7 implies the theorem. We remark that we scale by $\frac{2}{s_{\min}}$ instead of $\frac{1}{s_{\min}}$ because of technical reasons that will become clear in the proof of the theorem.

The algorithm scales the coefficients by factor γ; we use a' and b' to refer to the scaled coefficients. The idea of Algorithm 20 is to execute the greedy Algorithm 19 under the assumption that $a_i = U_i - L_i$ (and $a'_i = \gamma(U_i - L_i)$) for all $I_i \in \mathcal{I}$ that were not yet added to the solution. As $a_i = (w_i - L_i) \in (0, U_i - L_i)$, this means that we assume a_i to be slightly larger than its largest possible value. Consequently, $s_{\min}$ is the smallest (non-zero) coefficient a_i under this assumption. The algorithm computes the greedy choice based on the *optimistic greedy values*

$$\bar{g}_c(Q, I_i) = \sum_{S \in \mathcal{S} \colon I_i \in S} b'_S(Q) - \max\{0, b'_S(Q) - \gamma(U_i - L_i)\}$$

(if $b'(Q) \geq 1$) and

$$\bar{g}_s(Q, I_i) = |\{S \in \mathcal{S} \colon I_i \in S \mid b'_S(Q) > 0 \wedge b'_S(Q) - \gamma(U_i - L_i) \leq 0\}|$$

(otherwise). That is, the greedy values under the assumption $a_i = U_i - L_i$. We call these values optimistic as they might overestimate but never underestimate the actual greedy values. For subsets $G \subseteq \mathcal{I}$, we define $\bar{g}_s(Q, G)$ and $\bar{g}_c(Q, G)$ analogously.

In contrast to g_s and g_c, Algorithm 20 has sufficient information to compute $\bar{g}_s$ and $\bar{g}_c$, and, therefore, the best greedy choice based on the optimistic greedy values. The algorithm is designed to find, in each iteration, an element I_i with $g_c(Q, I_i) \geq \frac{1}{2} \cdot \bar{g}_c(Q, I_i)$ for the current Q (or analogously for $\bar{g}_s$ and g_s). We show that (i) this ensures that each iteration 2-approximates the greedy choice and (ii) that finding such an element takes only $\frac{1}{\tau}$ attempts in expectation.

Proof of Theorem 6.4.1. Let j be an arbitrary iteration of the outer while-loop, X_j denote the number of queries during the iteration, and Y_j indicate whether the algorithm executes iteration j ($Y_j = 1$) or not ($Y_j = 0$).

Assuming $Y_j = 1$, the algorithm during iteration j executes queries to elements I_i until either $w_i - L_i \geq \frac{1}{2}(U_i - L_i)$ or the problem is solved. Since $w_i \geq \frac{(U_i + L_i)}{2}$ implies $w_i - L_i \geq \frac{1}{2}(U_i - L_i)$ and $\mathbb{P}[w_i \geq \frac{(U_i + L_i)}{2}] \geq \tau$ holds by assumption, the number of attempts until the current I_i satisfies the inequality follows a geometric distribution with success probability at least τ. Hence, $\mathbb{E}[X_j \mid Y_j = 1] \leq \frac{1}{\tau}$; proving Property 2 of Definition 6.3.6.

We continue by proving Property 1 of Definition 6.3.6. Consider a fixed realization. Let $\bar{G}_j$ denote the queries of iteration j *except* the last one and let $I_{\bar{j}}$ denote the last query of iteration j. Then $G_j = \bar{G}_j \cup \{I_{\bar{j}}\}$ is the set of queries during the iteration. Finally, let Q_j denote the set of queries before iteration j. We show that G_j 2-approximates the greedy choice of the scaled instance, which implies Property 1 of Definition 6.3.6.

If the iteration solves the problem, then G_j clearly 1-approximates the greedy choice and we are done. Thus, assume otherwise. We distinguish between the two cases (1) $b'(Q_j) \geq 1$ and (2) $b'(Q_j) < 1$.

Case (1): We show first that G_j 2-approximates the greedy choice if $b'(Q_j) \geq 1$. In this case, we have $g = \bar{g}_c$ (cf. Line 3). By choice of $I_{\bar{j}}$, we have $\bar{g}_c(Q_j \cup \bar{G}_j, I_{\bar{j}}) = \max_{I_i \in \mathcal{I} \setminus (Q_j \cup \bar{G}_j)} \bar{g}_c(Q_j \cup \bar{G}_j, I_i)$, i.e., $I_{\bar{j}}$ has the best optimistic greedy value when it is chosen.

As the iteration does not solve the instance, we have $(w_{\bar{j}} - L_{\bar{j}}) \geq \frac{1}{2}(U_{\bar{j}} - L_{\bar{j}})$ by Line 6. This directly implies that the actual greedy value of $I_{\bar{j}}$ is at least half the optimistic greedy value, i.e., $g_c(Q_j \cup \bar{G}_j, I_{\bar{j}}) \geq \frac{1}{2} \cdot \bar{g}_c(Q_j \cup \bar{G}_j, I_{\bar{j}})$.

Since the best optimistic greedy value is never smaller than the best actual greedy value, we get $g_c(Q_j \cup \bar{G}_j, I_{\bar{j}}) \geq \frac{1}{2} \cdot \max_{I_i \in \mathcal{I} \setminus (Q_j \cup \bar{G}_j)} g_c(Q_j \cup \bar{G}_j, I_i)$. This allows us to apply Lemma 6.3.4 to get $b'(Q_j \cup \bar{G}_j \cup \{I_{\bar{j}}\}) \leq (1 - \frac{1}{2\mathrm{OPT}}) \cdot b'(Q_j \cup \bar{G}_j)$. Using $b'(Q_j \cup \bar{G}_j) \leq b'(Q_j)$ and $G_j = \bar{G}_j \cup \{I_{\bar{j}}\}$, we can conclude $b'(Q_j \cup G_j) \leq (1 - \frac{1}{2\mathrm{OPT}}) \cdot b'(Q_j)$, which shows that G_j satisfies Condition 2 of Definition 6.3.3.

Case (2): Next, we show that G_j 1-approximates the greedy choice if $b'(Q_j) < 1$. In this case, we have $g = \bar{g}_s$ (cf. Line 3). Similar to the previous case, we have $\bar{g}_s(Q_j \cup \bar{G}_j, I_{\bar{j}}) = \max_{I_i \in \mathcal{I} \setminus (Q_j \cup \bar{G}_j)} \bar{g}_s(Q_j \cup \bar{G}_j, I_i)$, i.e., $I_{\bar{j}}$ has the best optimistic greedy value when it is chosen.

From $b'(Q_j) = \sum_{S \in \mathcal{S}} b'_S(Q_j) < 1$ follows $b'_S(Q_j) < 1$ for all $S \in \mathcal{S}$. Furthermore, every element I_i with $w_i - L_i \geq \frac{1}{2}U_i - L_i$ satisfies $a_i = w_i - L_i \geq \frac{s_{\min}}{2}$ and, therefore $a'_i = \gamma a_i = \frac{2}{s_{\min}} \cdot a_i \geq 1$. This means that adding I_i to the solution satisfies *all* constraints for sets S with $I_i \in S$ that are previously not satisfied. Thus, the optimistic greedy value $\bar{g}_s(Q_j, I_i)$ and the actual greedy value $g_s(Q_j, I_i)$ are the same, i.e., $\bar{g}_s(Q_j, I_i) = g_s(Q_j, I_i)$, as adding I_i cannot satisfy more constraints even if the coefficient a_i was $U_i - L_i$. This observation is crucial for the remaining proof and the reason we scale with $\gamma = \frac{2}{s_{\min}}$ instead of $\frac{1}{s_{\min}}$.

Since the iteration does not solve the instance by assumption, we have $(w_{\bar{j}} - L_{\bar{j}}) \geq \frac{1}{2}(U_{\bar{j}} - L_{\bar{j}})$ by Line 6. As argued above, this implies that $I_{\bar{j}}$ has the best optimistic *and* actual greedy value when it is added to the solution, i.e., $g_s(Q_j \cup \bar{G}_j, I_{\bar{j}}) = \bar{g}_s(Q_j \cup \bar{G}_j, I_{\bar{j}}) = \max_{I_i \in \mathcal{I} \setminus (Q_j \cup \bar{G}_j)} \bar{g}_s(Q_j \cup \bar{G}_j, I_i)$. Thus, even under the assumption that all elements I_i of $\mathcal{I} \setminus (Q_j \cup \bar{G}_j)$ have coefficients $a_i = (U_i - L_i)$, interval $I_{\bar{j}}$ achieves the best greedy value.

Let LB denote the optimal solution value for the remaining instance after querying $Q_j \cup \bar{G}_j$ under exactly this assumption that $a_i = (U_i - L_i)$ and $a'_i = \gamma(U_i - L_i)$ for all $I_i \in \mathcal{I} \setminus (Q_j \cup \bar{G}_j)$. Clearly $LB \leq \mathrm{OPT}$.

Under the assumption that $a'_i = \gamma(U_i - L_i)$ for all $I_i \in \mathcal{I} \setminus (Q_j \cup \bar{G}_j)$, the instance is scaled (i.e., it satisfies that all non-zero coefficients are at least one). Thus, we can apply Lemma 6.3.4 under the assumption to get $A(Q_j \cup \bar{G}_j \cup \{I_{\bar{j}}\}) \leq (1 - \frac{1}{LB}) \cdot A(Q_j \cup \bar{G}_j)$. Since $LB \leq \mathrm{OPT}$, this gives us $A(Q_j \cup \bar{G}_j \cup \{I_{\bar{j}}\}) \leq (1 - \frac{1}{\mathrm{OPT}}) \cdot A(Q_j \cup \bar{G}_j)$. Using $A(Q_j \cup \bar{G}_j) \leq A(Q_j)$ and $G_j = \bar{G}_j \cup \{I_{\bar{j}}\}$, we can conclude $A(Q_j \cup G_j) \leq (1 - \frac{1}{\mathrm{OPT}}) \cdot A(Q_j)$, which shows that G_j satisfies Condition 2 of Definition 6.3.3 for $\alpha = 1$. □

6.5 MINSET under uncertainty

We consider the general MINSET under uncertainty. In contrast to the previous section, we now also have uncertainty in the right-hand sides of (MINSETIP). Since we consider the stochastic problem variant, recall that the balancing parameter is $\tau = \min_{I_i \in \mathcal{I}} \tau_i$ with $\tau_i = \mathbb{P}[w_i \geq \frac{U_i + L_i}{2}]$. Our goal is to iteratively add intervals from $\mathcal{I}$ to the solution until it becomes feasible for (MINSETIP). To that end, we prove the following main result.

Theorem 6.5.1. *For $\tau > 0$. There is an algorithm for* MINSET *under uncertainty with a competitive ratio of $\mathcal{O}(\frac{1}{\tau} \cdot \log m \cdot \bar{\rho}(\gamma)) \subseteq \mathcal{O}(\frac{1}{\tau} \cdot \log^2 m)$ with $\gamma = 2/s_{\min}$ for $s_{\min} = \min_{I_i \in \mathcal{I} \colon (U_i - L_i) > 0}(U_i - L_i)$.*

Exploiting Theorem 6.3.7, we prove the statement by providing Algorithm 21 and showing that it is an (α, β, γ)-GREEDY algorithm for $\alpha = 2$, $\gamma = 2/s_{\min}$ and $\beta = \frac{1}{\tau}(\lceil \log_{1.5}(m \cdot (2/s_{\min}) \cdot \max_{I_i \in \mathcal{I}}(U_i - L_i)) \rceil + \lceil \log_2(m) \rceil)$. Note that α and γ are defined as in the previous section for MINSET with deterministic right-hand sides and will be used analogously. For β on the other hand, we require a larger value to adjust for the additional uncertainty in the right-hand sides $b_S = w^* - L_S$ for the uncertain w^*. Notice that we do not have sufficient information to just execute Algorithm 20 for MINSET with deterministic right-hand sides as we need the right-hand side values to compute even the optimistic greedy values.

To handle this additional uncertainty, we want to ensure that each iteration of our algorithm α-approximates the greedy choice for *each* possible value of w^*. To do so, we compute and query the best optimistic greedy choice for several carefully selected possible values of w^*.

To state our algorithm, we define a parametrized variant of (MINSETIP) that states the problem under the assumptions that $w^* = w$ for some w and that the set $Q \subseteq \mathcal{I}$ has already been queried. The coefficients are scaled to $a'_i = (2/s_{\min})(w_i - L_i)$ and $b'_S(Q, w) = \max\{(2/s_{\min})(w - L_S) - \sum_{I_i \in Q \cap S} a'_i, 0\}$. As before, let $b'(Q, w) = \sum_{S \in \mathcal{S}} b'_S(Q, w)$ denote the sum of right-hand sides.

$$\begin{array}{lllll} \min & \sum_{I_i \in \mathcal{I} \setminus Q} x_i & & & \\ \text{s.t.} & \sum_{I_i \in S \setminus Q} x_i \cdot a'_i & \geq b'_S(Q, w) & \forall S \in \mathcal{S} & \\ & x_i & \in \{0, 1\} & \forall I_i \in \mathcal{I} & \end{array} \tag{MINSETIP-QW}$$

As the right-hand sides are unknown, we define the greedy values for every possible value w for w^*. To that end, let $g_c(Q, I_i, w) = b'(Q, w) - b'(Q \cup \{I_i\}, w)$ and $g_s(Q, I_i, w) = A(Q, w) - A(Q \cup \{I_i\}, w)$, where $A(Q, w) = |\{S \in \mathcal{S} \mid b'_S(Q, w) > 0\}|$ denotes the number of constrains in (MINSETIP-QW) that are not yet satisfied. As before, $g_c(Q, I_i, w)$ and $g_s(Q, I_i, w)$ describe how much adding I_i to the solution reduces the sum of right-hand sides and the number of non-satisfied constraints, respectively; now under the assumption that $w^* = w$. For subsets $G \subseteq \mathcal{I} \setminus Q$, we define the greedy values in the same way, i.e., $g_s(Q, G, w) = A(Q, w) - A(Q \cup G, w)$ and $g_c(Q, G, w) = b'(Q, w) - b'(Q \cup G, w)$.

Since our algorithm again does not have sufficient information to compute the precise greedy values $g_s(Q, I_i, w)$ and $g_c(Q, I_i, w)$ even for a fixed w, we again use the optimistic greedy values defined in the same way as in the previous section. That is

$$\bar{g}_c(Q, I_i, w) = \sum_{S \in \mathcal{S} \colon I_i \in S} b'_S(Q, w) - \max\{0, b'_S(Q, w) - \gamma(U_i - L_i)\}$$

and

$$\bar{g}_s(Q, I_i, w) = |\{S \in \mathcal{S} \colon I_i \in S \mid b'_S(Q, w) > 0 \land b'_S(Q, w) - \gamma(U_i - L_i) \leq 0\}|.$$

For subsets $G \subseteq \mathcal{I} \setminus Q$, the optimistic greedy values are defined analogously.

Similar to Algorithm 20, we would like to repeatedly compute and query the best optimistic greedy choice until the queried I_i satisfies $w_i - L_i \geq \frac{U_i - L_i}{2}$ (cf. the repeat-statement). However, we cannot decide which greedy value, $\bar{g}_c$ or $\bar{g}_s$, to use as deciding whether $b'_S(Q, w^*) < 1$ depends on the unknown w^*. Instead, we compute and query the best optimistic greedy choice for both greedy values (cf. the for-loop). Even then, the best greedy choice still depends on the unknown right-hand sides. Thus, we compute and query the best optimistic greedy choice for several carefully selected values w (cf. the inner while-loop) to make sure that the queries of the iteration approximate the greedy choice for every possible w^*. Additionally, we want to ensure that we use at most β queries in expectation within an iteration of the outer while-loop.

To illustrate the ideas of the algorithm, consider an iteration of the outer while-loop. In particular, consider the for-loop iteration with $g = \bar{g}_c$ within this iteration. Let Q' denote the set of queries that were executed before the start of the iteration. Since we only care about the greedy value g_c if there exists some $S \in \mathcal{S}$ with $b'_S(Q') \geq 1$ (otherwise we use $\bar{g}_s$ and g_s

Algorithm 21: Algorithm for MINSET under uncertainty.

```
   Input: Instance of MINSET under uncertainty.
1  Scale all coefficients with $\gamma = 2/s_{\min}$ for $s_{\min} = \min_{I_i \in \mathcal{I}:\, (U_i - L_i)>0}(U_i - L_i)$;
2  $Q \leftarrow \emptyset$, $w_{\min} \leftarrow$ minimum possible value $w^*$ (keep up-to-date);
3  while the problem is not solved do
4      foreach $g$ from the ordered list $\bar{g}_c, \bar{g}_s$ do
5          $d \leftarrow 1$; $Q' \leftarrow Q$;
6          if $g = \bar{g}_c$ then $w_{\max} \leftarrow$ max possible value $w^*$;
7          else $w_{\max} \leftarrow \max w$ s.t. $b'_S(Q, w) < 1$ for all $S \in \mathcal{S}$;
8          while $\exists w_{\min} \leq w \leq w_{\max}$ such that $\max_{I_h \in \mathcal{I} \setminus Q} g(Q, I_h, w) \geq d$ do
9              repeat
10                 $w \leftarrow \min w_{\min} \leq w \leq w_{\max}$ s.t. $\max_{I_h \in \mathcal{I} \setminus Q} g(Q, w, I_h) \geq d$ ;
11                 $I_i \leftarrow \arg\max_{I_h \in \mathcal{I} \setminus Q} g(Q, I_h, w)$; Query $I_i$; $Q \leftarrow Q \cup \{I_i\}$;
12                 $Q_{1/2} \leftarrow \{I_j \in Q \setminus Q' \mid w_j - L_j \geq \frac{U_j - L_j}{2}\}$;
13                 if $g = \bar{g}_c$ then $d \leftarrow g_c(Q', Q_{1/2}, w)$ else $d \leftarrow g_s(Q', Q_{1/2}, w)$;
14             until $w_i - L_i \geq \frac{U_i - L_i}{2}$ or $\nexists w \leq w_{\max}\colon \max_{I_h \in \mathcal{I} \setminus Q} g(Q, w, I_h) \geq d$ ;
```

instead), we assume that this is the case. If not, we use a separate analysis for the for-loop iteration with $g = \bar{g}_s$.

Our goal for the iteration is to query a set of intervals $\bar{Q}$ that 2-approximates the best greedy choice I^* after querying Q', i.e., it has a greedy value $g_c(Q', \bar{Q}, w^*) \geq \frac{1}{2} g_c(Q', I^*, w^*)$ and, thus, satisfies Lemma 6.3.4. To achieve this for the unknown w^*, the algorithm uses the parameter d, which is initialized with 1 (cf. Line 5), the minimum possible value for $\bar{g}_c(Q', I^*, w^*)$ under the assumption that there exists some $S \in \mathcal{S}$ with $b'_S(Q') \geq 1$. In an iteration of the inner while-loop, the algorithm repeatedly picks the minimal value w such that the best current optimistic greedy choice has an optimistic greedy value of at least d (cf. Line 10). If no such value exists, then the loop terminates (cf. Lines 8, 14). Afterwards, it queries the corresponding best optimistic greedy choice I_i for the selected value w (cf. Line 11). Similar to the algorithms of the previous section, this is done repeatedly until $w_i - L_i \geq (U_i - L_i)/2$.

The key idea to achieve the 2-approximation with an expected number of queries that does not exceed β, is to always reset the value d to $g_c(Q', Q_{1/2}, w)$, where $Q_{1/2}$ is the subset of all intervals I_j that have already been queried in the current iteration of the outer while-loop and satisfy $w_j - L_j \geq (U_j - L_j)/2$ (cf. Lines 12, 13). This can be seen as an implicit doubling strategy to search for an unknown value. It leads to an exponential increase of d over the iterations of the inner while-loop, which will allow us to bound their number.

With the following lemma, we prove that this choice of d also ensures that the queries of the iteration indeed 2-approximate the best greedy choice for w^* if there exists a $S \in \mathcal{S}$ with $b'_S(Q', w^*) \geq 1$. If there is no such set, we can use a similar proof w.r.t. greedy value g_s. For an iteration j of the outer while-loop, let G_j be the set of queries during the iteration and let $Q_j = \bigcup_{j' < j} G_j$ denote the queries before the iteration (cf. Q' in the algorithm).

Lemma 6.5.2. *If there is an $S \in \mathcal{S}$ with $b'_S(Q_j, w^*) \geq 1$, then G_j 2-approximates the greedy choice for the scaled instance with $w = w^*$ after querying Q_j.*

Proof. For an arbitrary but fixed realization, consider an iteration j of the outer while-loop such that there exists a set $S \in \mathcal{S}$ with $b'_S(Q_j, w^*) \geq 1$.

Consider the subset $\bar{G}_j \subseteq G_j$ of queries that were executed with $g = \bar{g}_c$ before the increasing value w (cf. Line 10) surpasses w^*. That is, $\bar{G}_j$ only contains intervals that were queried for a current value $w \leq w^*$. Let $\bar{I}_i$ be the element of $\bar{G}_j$ that is queried last. Finally, let $\bar{d}_j$ denote the value d computed by the algorithm in Line 13 directly after querying $\bar{I}_i$. We continue to show that G_j 2-approximates the greedy choice of (MINSETIP-QW) for $Q = Q_j$ and $w = w^*$.

Observe that $g_c(Q_j, \bar{G}_j, w^*) \geq \bar{d}_j$. To see this, recall that $\bar{d}_j$ was computed in Line 13 after $\bar{I}_i$ was queried. Thus, $\bar{d}_j = g_c(Q', Q_{1/2}, w)$ for $Q' = Q_j, Q_{1/2} = \{I_j \in \bar{G}_j \mid w_j - L_j \geq \frac{U_j - L_j}{2}\}$ and some value w with $w \leq w^*$ by assumption. Since $w^* \geq w$ and $Q_{1/2} \subseteq \bar{G}_j$, the greedy value $g_c(Q_j, \bar{G}_j, w^*)$ can never be smaller than $\bar{d}_j = g_c(Q', Q_{1/2}, w)$. This implies $g_c(Q_j, \bar{G}_j, w^*) \geq \bar{d}_j$.

We continue by showing that $d^* \leq 2 \cdot g_c(Q_j, \bar{G}_j, w^*)$ holds for the best greedy value d^* at the start of the iteration, i.e., $d^* = \max_{I_i \in \mathcal{I} \setminus Q_j} g_c(Q_j, I_i, w^*)$. As $\bar{G}_j \subseteq G_j$, this implies $d^* \leq 2 \cdot g_c(Q_j, G_j, w^*)$ and, thus, that G_j satisfies Definition 6.3.3.

To upper bound d^*, first observe that the best optimistic greedy value d' after querying $\bar{G}_j \cup Q_j$ is smaller than $\bar{d}_j$, i.e., $d' = \max_{I_i \in \mathcal{I} \setminus (Q_j \cup \bar{G}_j)} \bar{g}_c(Q_j \cup \bar{G}_j, I_i, w^*) < \bar{d}_j$. This follows directly from Line 10 as $\bar{I}_i$ is the last query for a value $w \leq w^*$ by assumption. As $g_c(Q_j, \bar{G}_j, w^*) \geq \bar{d}_j$, we get $g_c(Q_j, \bar{G}_j, w^*) \geq d'$.

By definition of g_c, the best greedy value after querying Q_j can never be larger than the sum of the greedy value of $\bar{G}_j$ after querying Q_j and the best optimistic greedy value after querying $\bar{G}_j \cup Q_j$. Thus, we have $d^* \leq g_c(Q_j, \bar{G}_j, w^*) + d' \leq 2 \cdot g_c(Q_j, \bar{G}_j, w^*)$. This proves that G_j satisfies Lemma 6.3.4 and, thus, concludes this proof. $\square$

Using a similar proof, we show the following lemma for the case where $b'_S(Q', w^*) < 1$ for all $S \in \mathcal{S}$, which together with Lemma 6.5.2 implies Property 1 of Definition 6.3.6. While the main arguments remain the same as for the previous lemma, the more discrete nature of the greedy values g_s and $\bar{g}_s$ poses several additional technical challenges that need to be taken care of. For an iteration j of the outer while-loop, let G_j again be the set of queries during the iteration and let $Q_j = \bigcup_{j' < j} G_j$ denote the queries before the iteration (cf. set Q' in the algorithm).

Lemma 6.5.3. *If $b'_S(Q_j, w^*) < 1$ for all $S \in \mathcal{S}$, then G_j 2-approximates the greedy choice for the scaled instance with $w = w^*$ after querying Q_j.*

Proof. For an arbitrary but fixed realization, consider an iteration j of the outer while-loop such that $b'_S(Q_j, w^*) < 1$ for all $S \in \mathcal{S}$. Our goal is to prove that G_j approximates the greedy choice for the scaled instance with $w = w^*$ after querying Q_j within a factor of two. That is, we have to prove $A(Q_j \cup G_j, w^*) \leq (1 - \frac{1}{2 \cdot \mathrm{OPT}}) \cdot A(Q_j, w^*)$. Recall that $A(Q_j, w^*)$ denotes the number of constraints that are not yet satisfied in the (MINSETIP-QW) instance for $Q = Q_j$ and $w = w^*$.

Consider the subset $\bar{G}_j \subseteq G_j$ of queries to intervals I_i that were executed with $g = \bar{g}_s$ for a current value $w \leq w^*$ during iteration j of the outer while-loop. Let $P_j \subseteq G_j$ denote the queries of the iteration that were executed before $\bar{G}_j$, i.e., that were executed during the iteration of the for-loop with $g = \bar{g}_c$. Note that $Q' = Q_j \cup P_j$ is the set of intervals queried before the beginning of the for-loop iteration with $g = \bar{g}_s$ during iteration j of the outer while-loop.

Proof outline. We start the proof by making some preliminary observations regarding greedy value g_s and the scaling factor γ that will be crucial for the remainder of the proof. Then we proceed by proving that $\bar{G}_j$ approximates the greedy choice of (MINSETIP-QW) for $Q = Q_j \cup P_j$ and $w = w^*$ within a factor of 2. To this end, we first derive a lower bound on the greedy value $g_s(Q_j \cup P_j, \bar{G}_j, w^*)$ and afterwards compare this lower bound with OPT. Finally, we use the fact that $\bar{G}_j$ 2-approximates the greedy choice after querying $Q_j \cup P_j$ to show that G_j approximates the greedy choice of (MINSETIP-QW) for $Q = Q_j$ and $w = w^*$ within a factor of 2.

Preliminary observations. Before we start with the proof, recall that an interval I_i with $(w_i - L_i) \geq \frac{1}{2} \cdot (U_i - L_i)$ satisfies $a'_i = \frac{2 \cdot (w_i - L_i)}{s_{\min}} \geq \frac{U_i - L_i}{s_{\min}} \geq 1$ by choice of the scaling parameter $\gamma = \frac{2}{s_{\min}}$. This implies that adding I_i to the solution satisfies all constraints for sets S with $I_i \in S$ as long as we are considering values w with $b'_S(Q_j, w) \leq 1$ for all $S \in \mathcal{S}$. Thus, for such intervals and values w, the greedy value g_s of I_i is then equal to the optimistic greedy value $\bar{g}_s$ as even under the assumption $a'_i = \gamma(U_i - L_i)$ adding interval I_i cannot satisfy more constraints. By assumption, this in particular holds for all values $w \leq w^*$. This also means that the greedy values g_s and $\bar{g}_s$ of such intervals I_i only increase with an increasing value w, as long as $b'_S(Q_j, w) \leq 1$ still holds for all $S \in \mathcal{S}$.

Lower bound on $g_s(Q_j \cup P_j, \bar{G}_j, w^*)$. We continue by deriving a lower bound on $g_s(Q_j \cup P_j, \bar{G}_j, w^*)$. Let $\bar{I}_i$ be the element of $\bar{G}_j$ that is queried last and let $\bar{d}_j$ denote the value d computed by the algorithm in Line 13 directly after querying $\bar{I}_i$. We first observe that $g_s(Q_j \cup P_j, \bar{G}_j, w^*) \geq \bar{d}_j$. To see this, recall that $\bar{d}_j$ was computed in Line 13 after $\bar{I}_i$ was queried. Thus, $\bar{d}_j = g_s(Q', Q_{1/2}, w)$ for $Q' = Q_j \cup P_j$, $Q_{1/2} = \{I_j \in \bar{G}_j \mid w_j - L_j \geq \frac{U_j - L_j}{2}\}$ and some value w with $w \leq w^*$ by assumption. Since $w^* \geq w$ and $Q_{1/2} \subseteq \bar{G}_j$, the greedy value $g_s(Q_j \cup P_j, \bar{G}_j, w^*)$ can never be smaller than $\bar{d}_j = g_s(Q', Q_{1/2}, w)$ according to the observations stated at the beginning of the proof. This implies $g_s(Q_j \cup P_j, \bar{G}_j, w^*) \geq \bar{d}_j$.

Assume for now that the algorithm queried $Q_s = \bar{G}_j \setminus Q_{1/2}$ before $Q_{1/2}$. We consider the best optimistic greedy value d^* after $Q_j \cup P_j \cup Q_s$ has already been queried, i.e, $d^* = \max_{I_i \in \mathcal{I} \setminus (Q_j \cup P_j \cup Q_s)} \bar{g}_s(Q_j \cup P_j \cup Q_s, I_i, w^*)$. To bound d^*, first observe that the best optimistic greedy value d' after querying $\bar{G}_j \cup Q_j \cup P_j$ is smaller than $\bar{d}_j$, i.e., $d' = \max_{I_i \in \mathcal{I} \setminus (Q_j \cup P_j \cup \bar{G}_j)} \bar{g}_s(Q_j \cup P_j \cup \bar{G}_j, I_i, w^*) < \bar{d}_j$. This follows directly from Line 10 as $\bar{I}_i$ is the last query for a value $w \leq w^*$ by assumption. Since we already showed $g_s(Q_j \cup P_j, \bar{G}_j, w^*) \geq \bar{d}_j$, we get $g_s(Q_j \cup P_j, \bar{G}_j, w^*) \geq d'$.

By definition of $\bar{g}_s$, definition of $Q_{1/2}$, and the assumption that we only consider values w with $b'_S(Q', w) < 1$ for all $S \in \mathcal{S}$, the best optimistic greedy value after querying $Q_j \cup P_j \cup Q_s$ can never be larger than the sum of the optimistic greedy value of $Q_{1/2}$ after querying $Q_j \cup P_j \cup Q_s$ and the best optimistic greedy value after querying $Q_j \cup P_j \cup Q_s \cup Q_{1/2} = Q_j \cup P_j \cup \bar{G}_j$. By assumption that we only consider values w with $b'_S(Q', w) < 1$ for all $S \in \mathcal{S}$, we have $\bar{g}_s(Q_j \cup P_j \cup Q_s, Q_{1/2}, w^*) = g_s(Q_j \cup P_j \cup Q_s, Q_{1/2}, w^*)$ (as argued at the beginning of the proof). Putting it together, we get

$$\begin{aligned} d^* &\leq \bar{g}_s(Q_j \cup P_j \cup Q_s, Q_{1/2}, w^*) + d' \\ &= g_s(Q_j \cup P_j \cup Q_s, Q_{1/2}, w^*) + d' \\ &\leq g_s(Q_j \cup P_j \cup Q_s, Q_{1/2}, w^*) + g_s(Q_j \cup P_j, \bar{G}_j, w^*). \end{aligned}$$

Proving that $\bar{G}_j$ approximates its greedy choice. We continue the proof by using the inequality for d^* in order to show that $\bar{G}_j$ approximates its greedy choice.

To this end, we consider a relaxed instance R of (MINSETIP-QW) with $Q = Q_j \cup P_j$ and $w = w^*$, i.e., we assume that $Q_j \cup P_j$ has already been queried and consider the right-hand sides $b'_S(Q_j \cup P_j, w^*)$ for all $S \in \mathcal{S}$. We relax the instance by increasing the left-hand side coefficients and decreasing the cost coefficients. Let $\hat{a}$ denote the relaxed coefficients. First consider the intervals in $\bar{G}_j$. For these intervals, we use the original scaled coefficients. That is, we set $\hat{a}_i = a'_i = \gamma(w_i - L_i)$. Recall that $Q_{1/2} = \{I_i \in \bar{G}_j \mid w_i - L_i \geq \frac{1}{2}(U_i - L_i)\}$ and $Q_s = \bar{G}_j \setminus Q_{1/2}$. For intervals $I_i \in Q_s$, we set the cost coefficients to zero. Thus, these intervals can be added to any solution without increasing the objective value. All other intervals keep their cost coefficients of one. For all other intervals $I_i \in \mathcal{I} \setminus (Q_j \cup P_j \cup \bar{G}_j)$, we set $\hat{a}_i = \gamma(U_i - L_i)$. That is, we assume that the coefficients of these intervals are slightly larger than their largest possible value. Since instance R compared to the original instance only increases left-hand side coefficients and decreases cost coefficients, it clearly is a relaxation and, thus, $LB \leq \mathrm{OPT}$ for the optimal objective value LB of the relaxed instance R.

Since we assume $b'_S(Q_j \cup P_j, w^*) < 1$ for all $S \in \mathcal{S}$, the right-hand sides of instance R are all strictly smaller than one. Furthermore, by choice of the scaling parameter γ, all intervals I_i satisfy $\gamma(U_i - L_i) \geq 1$. Thus, the intervals I_i in $Q \setminus (Q_j \cup P_j \cup \bar{G}_j)$ have coefficients $\hat{a}_i \geq 1$. By the observations at the beginning of the proof, the same holds for the intervals in $Q_{1/2}$. Only the intervals in Q_s can potentially have coefficients smaller than the right-hand sides. This also means that the greedy value g_s of an interval $I_i \in Q \setminus (Q_j \cup P_j \cup Q_s)$ for instance R can never be increased by previously adding intervals of Q_s to the solution, since adding I_i already satisfies all constraints for sets S with $I_i \in S$. For intervals in Q_s, it might be the case that previously adding further elements of Q_s to the solution increases the greedy value. This is, because the coefficient of an $I_i \in Q_s$ might be too small to satisfy a constraint S with $I_i \in S$ but previously adding further intervals from Q_s might decrease the right-hand side of S enough such that adding I_i can satisfy the constraint and, thus, increase the greedy value of I_i.

Now, consider the number of constraints that are initially not satisfied in R. This number is exactly the same as in the non-relaxed original instance after querying $Q_j \cup P_j$ since R uses the exact same right-hand sides. Thus, the number of not yet satisfied constraints in R is $A(Q_j \cup P_j, w^*)$. Let $\hat{d} = \max_{I_i \in \mathcal{I} \setminus (Q_j \cup P_j \cup Q_s)} |\{S \in S \mid b'_S(Q_j \cup P_j, w^*) > 0 \land b'_S(Q_j \cup P_j, w^*) - \hat{a}_i - \sum_{I_h \in Q_s} \hat{a}_h \leq 0\}|$ denote the maximum number of constraints that can be satisfied by adding Q_s and *one* interval I_i in $Q_j \cup P_j \cup Q_s$ to the solution for R. In a sense, $\hat{d}$ is the best possible greedy value g_s for instance R that can be achieve by adding such a set $Q_s \cup \{I_i\}$ to the solution. Remember that the elements of Q_s do not incur any costs, so $\hat{d}$ constraints can be satisfied at cost one. Following the arguments in the proof of Lemma 6.3.4, this implies $\hat{d} \geq \frac{A(Q_j \cup P_j, w^*)}{LB}$ as the optimal solution satisfies $A(Q_j \cup P_j, w^*)$ constraints at cost LB but it is impossible to satisfy more than $\hat{d}$ constraints with cost one. This last implication crucially uses the observation that the greedy values of intervals in $Q \setminus (Q_j \cup P_j \cup Q_s)$ cannot increase by previously adding other intervals to the solution. Because this is the case, even adding an interval later cannot satisfy more than $\hat{d}$ constraints, which gives us $\hat{d} \geq \frac{A(Q_j \cup P_j, w^*)}{LB}$. We remark that this also is the reason why the offline Algorithm 19 starts with greedy value g_c and only switches to greedy value g_s once the instance reduced to a SETCOVER instance.

Taking a closer look at $\hat{d}$, we can observe that this greedy value is exactly the sum of the greedy values $g_s(Q_j \cup P_j, Q_s, w^*)$ and $d^* = \max_{I_i \in Q \setminus (Q_j \cup P_j \cup Q_s)} \bar{g}_s(Q_j \cup P_j \cup Q_s, I_i, w^*)$ for the non-relaxed instance. This is, because the original instance uses the same right-hand sides as R, the coefficients of intervals in Q_s are the same in both instances, and the optimistic greedy value $\bar{g}_s(Q_j \cup P_j \cup Q_s, I_i, w^*)$ like instance R already assumes that all intervals in $Q \setminus (Q_j \cup P_j \cup Q_s)$ have coefficients larger than all right-hand sides. Thus, these three arguments imply $\hat{d} = g_s(Q_j \cup P_j, Q_s, w^*) + d^*$. Combining this equality with $\hat{d} \geq \frac{A(Q_j \cup P_j, w^*)}{LB}$ and the previously derived upper bound on d^* gives us:

$$\begin{aligned}\frac{A(Q_j \cup P_j, w^*)}{LB} &\leq \hat{d} \\ &= g_s(Q_j \cup P_j, Q_s, w^*) + d^* \\ &\leq g_s(Q_j \cup P_j, Q_s, w^*) + g_s(Q_j \cup P_j \cup Q_s, Q_{1/2}, w^*) + g_s(Q_j \cup P_j, \bar{G}_j, w^*) \\ &= 2 \cdot g_s(Q_j \cup P_j, \bar{G}_j, w^*).\end{aligned}$$

Here the second equality uses that

$$\begin{aligned}
&g_s(Q_j \cup P_j, Q_s, w^*) + g_s(Q_j \cup P_j \cup Q_s, Q_{1/2}, w^*)\\
=&(A(Q_j \cup P_j, w^*) - A(Q_j \cup P_j \cup Q_s, w^*))\\
+&(A(Q_j \cup P_j \cup Q_s, w^*) - A(Q_j \cup P_j \cup Q_s \cup Q_{1/2}, w^*))\\
=&A(Q_j \cup P_j, w^*) - A(Q_j \cup P_j \cup Q_s \cup Q_{1/2}, w^*)\\
=&A(Q_j \cup P_j, w^*) - A(Q_j \cup P_j \cup \bar{G}_j, w^*)\\
=&g_s(Q_j \cup P_j, \bar{G}_j, w^*).
\end{aligned}$$

Plugging in the definition of g_s and the inequality $LB \leq \mathrm{OPT}$ yields

$$\begin{aligned}
&\frac{A(Q_j \cup P_j, w^*)}{LB} \leq 2 \cdot (A(Q_j \cup P_j, w^*) - A(Q_j \cup P_j \cup \bar{G}_j, w^*))\\
\Leftrightarrow &A(Q_j \cup P_j \cup \bar{G}_j, w^*) \leq A(Q_j \cup P_j, w^*) \cdot (1 - \frac{1}{2 \cdot LB})\\
\Rightarrow &A(Q_j \cup P_j \cup \bar{G}_j, w^*) \leq A(Q_j \cup P_j, w^*) \cdot (1 - \frac{1}{2 \cdot \mathrm{OPT}}).
\end{aligned}$$

Thus, $\bar{G}_j$ approximates the greedy choice of (MINSETIP-QW) for $Q = Q_j \cup P_j$ and $w = w^*$ within a factor of 2.

Concluding the proof. Remember that our goal was to show that G_j approximates the greedy choice for $w = w^*$ after querying Q_j within a factor of 2. To that end, observe that $P_j \cup \bar{G}_j \subseteq G_j$ implies $A(Q_j \cup P_j \cup \bar{G}_j, w^*) \geq A(Q_j \cup G_j, w^*)$ and that $Q_j \subseteq Q_j \cup P_j$ implies $A(Q_j, w^*) \geq A(Q_j \cup P_j, w^*)$. Plugging these inequalities into the previously derived inequality for $A(Q_j \cup P_j \cup \bar{G}_j, w^*)$ yields $A(Q_j \cup G_j, w^*) \leq A(Q_j, w^*) \cdot (1 - \frac{1}{2 \cdot \mathrm{OPT}})$. This means that G_j also 2-approximates the greedy choice after querying Q_j for $w = w^*$, which concludes the proof. □

Since the Lemmas 6.5.2 and 6.5.3 imply Condition 1 of Definition 6.3.6, it remains to show Condition 2 in order to apply Theorem 6.3.7. The proof idea is to show that parameter d increases by a factor of at least 1.5 in each iteration of the inner while-loop with $g = \bar{g}_c$. As $\bar{g}_c(Q, I_i, w)$ is upper bounded by $m(2/s_{\min}) \max_{I_i \in \mathcal{I}}(U_i - L_i)$, this means the inner loop executes at most $\lceil \log_{1.5}(m(2/s_{\min}) \max_{I_i \in \mathcal{I}}(U_i - L_i)) \rceil$ iterations for $g = \bar{g}_c$. For $g = \bar{g}_s$, we can argue in a similar way that at most $\lceil \log_2(m) \rceil$ iterations are executed. Similar to the previous section on MINSET with deterministic right-hand sides, we can also show that each iteration of the inner while-loop executes at most $\frac{1}{\tau}$ queries in expectation. Combining these insights, we can bound the expected number of queries during an execution of the outer while-loop by β. Formally proving the increase of d by 1.5 requires to take care of several technical challenges. The basic idea is to exploit that the interval I_i queried in Line 11 has an optimistic greedy value of at least d. If it satisfies $w_i - L_i \geq \frac{1}{2}(U_i - L_i)$, we show that the actual greedy value is at least $d/2$. When d is recomputed in Line 13, then I_i is a new member of the set $Q_{1/2}$ and leads to the increase of $d/2$.

We conclude the section by formalizing this proof idea. For each iteration j of the outer while-loop, let X_j be a random variable denoting the number of queries executed during iteration j and let Y_j be a variable indicating whether the iteration is executed ($Y_j = 1$) or not ($Y_j = 0$). We prove the following lemma, which implies that the algorithm also satisfies the second condition of Definition 6.3.6 and, thus, satisfies Theorem 6.5.1.

Lemma 6.5.4. $\mathbb{E}[X_j \mid Y_j = 1] \leq \beta = \frac{1}{\tau} \cdot (\lceil \log_{1.5}(m \cdot \frac{2 \cdot \max_{I_i \in \mathcal{I}}(U_i - L_i)}{s_{\min}}) \rceil + \lceil \log_2(m) \rceil)$.

Proof. Consider an iteration j of the outer while-loop of Algorithm 21. In the following, all probabilities and expected values are under the condition $Y_j = 1$.

For iteration j, let A_{jk} and B_{jk} be random variables denoting the number of queries in iteration k of the inner while-loop for $g = \bar{g}_c$ and $g = \bar{g}_s$, respectively. Then, $\mathbb{E}[X_j] = \sum_k \mathbb{E}[B_{jk}] + \sum_k \mathbb{E}[A_{jk}]$.

Let $\bar{A}_{jk}$ and $\bar{B}_{jk}$ be random variables indicating whether iteration k of the inner while-loop with $g = \bar{g}_c$ and $g = \bar{g}_s$, respectively, is executed ($\bar{A}_{jk} = 1$, $\bar{B}_{jk} = 1$) or not. Given that such an iteration is executed, we can exploit that the termination criterion $w_i \geq \frac{1}{2} \cdot (U_i - L_i)$ occurs with probability $\mathbb{P}[(w_i - L_i) \geq (U_i - L_i)/2] \geq \mathbb{P}[w_i \geq (U_i + L_i)/2] \geq \tau$ to show $\mathbb{E}[A_{jk} \mid \bar{A}_{jk} = 1] \leq \frac{1}{\tau}$ and $\mathbb{E}[B_{jk} \mid \bar{B}_{jk} = 1] \leq \frac{1}{\tau}$. Using these bounds and exploiting that $\mathbb{E}[A_{jk} \mid \bar{A}_{jk} = 0] = \mathbb{E}[B_{jk} \mid \bar{B}_{jk} = 0] = 0$, we get

$$\begin{aligned}
\mathbb{E}[X_j] &= \sum_k (\mathbb{E}[A_{jk}] + \mathbb{E}[B_{jk}]) \\
&= \sum_k \mathbb{P}[\bar{A}_{jk} = 1] \cdot \mathbb{E}[A_{jk} \mid \bar{A}_{jk} = 1] + \mathbb{P}[\bar{A}_{jk} = 0] \cdot \mathbb{E}[A_{jk} \mid \bar{A}_{jk} = 0] \\
&+ \sum_k \mathbb{P}[\bar{B}_{jk} = 1] \cdot \mathbb{E}[B_{jk} \mid \bar{B}_{jk} = 1] + \mathbb{P}[\bar{B}_{jk} = 0] \cdot \mathbb{E}[B_{jk} \mid \bar{B}_{jk} = 0] \\
&= \sum_k \mathbb{P}[\bar{A}_{jk} = 1] \cdot \mathbb{E}[A_{jk} \mid \bar{A}_{jk} = 1] + \sum_k \mathbb{P}[\bar{B}_{jk} = 1] \cdot \mathbb{E}[B_{jk} \mid \bar{B}_{jk} = 1] \\
&\leq \frac{1}{\tau} \cdot \left(\sum_k \mathbb{P}[\bar{A}_{jk} = 1] + \sum_k \mathbb{P}[\bar{B}_{jk} = 1] \right).
\end{aligned}$$

Thus, it only remains to bound $\sum_k \mathbb{P}[\bar{A}_{jk} = 1] + \sum_k \mathbb{P}[\bar{B}_{jk} = 1]$. We do this by separately proving $\sum_k \mathbb{P}[\bar{A}_{jk} = 1] \leq \lceil \log_{1.5} (m \cdot (2/s_{\min}) \cdot \max_{I_i \in \mathcal{I}} (U_i - L_i)) \rceil$ and $\sum_k \mathbb{P}[\bar{B}_{jk} = 1] \leq \lceil \log_2(m) \rceil$.

Putting all bounds together, we then get $\mathbb{E}[X_j] \leq \frac{1}{\tau} \cdot (\sum_k \mathbb{P}[\bar{A}_{jk} = 1] + \mathbb{P}[\bar{B}_{jk} = 1]) \leq \frac{1}{\tau} \cdot (\lceil \log_{1.5}(m \cdot \frac{2 \cdot \max_{I_i \in \mathcal{I}}(U_i - L_i)}{s_{\min}}) \rceil + \lceil \log_2(m) \rceil)$.

Upper bound for $\sum_k \mathbb{P}[\bar{A}_{jk} = 1]$. Note that $\sum_k \mathbb{P}[\bar{A}_{jk} = 1]$ is just the expected number of iterations of the inner while-loop with $g = \bar{g}_c$ during iteration j of the outer while-loop. We bound this number by showing that, for every realization of precise weights, the value d increases by a factor of at least 1.5 in each iteration except possibly the first and last ones. As d is upper bounded by $a = (2/s_{\min}) \cdot m \cdot \max_{I_i \in \mathcal{I}} (U_i - L_i)$ (in the sense that the loop will terminate at the latest when $d > a$), the number of iterations then is at most $\lceil \log_{1.5} (a) \rceil$ for every realization of values.

To prove that d increases by a factor of 1.5 in each iteration of the inner while-loop, consider the last execution of the repeat-statement within an iteration of the inner while-loop with $g = \bar{g}_c$ (that is not the first or last one). Let I_i be the interval queried in that last iteration of the repeat-statement, let Q denote the set of I_i and all intervals that were queried before I_i, let Q' denote the set of all intervals queried before the current execution of the inner while-loop was started, and let $Q_{1/2} = \{I_j \in Q \setminus Q' \mid w_j - L_j \geq \frac{U_j - L_j}{2}\}$. Furthermore, let w denote the value of Line 10 computed before I_i was queried in the following line. Note that this notation matches the one used in the algorithm at the execution of Line 12 directly after I_i was queried.

Let $\bar{d}$ denote the value d computed by the algorithm *before* I_i was queried. That is, $\bar{d} = g_c(Q', Q_{1/2} \setminus \{I_i\}, w')$ for the value w' that was computed in the previous execution of Line 10. By choice of I_i, we have $\bar{g}_c(Q \setminus \{I_i\}, I_i, w) \geq \bar{d}$ and $g_c(Q \setminus \{I_i\}, I_i, w) \geq \bar{d}/2$ (since $w_i - L_i \geq (U_i - L_i)/2$).

The value d computed after querying I_i is defined as

$$d = g_c(Q', Q_{1/2}, w) \geq g_c(Q', Q_{1/2} \setminus \{I_i\}, w) + g_c(Q' \cup Q_{1/2} \setminus \{I_i\}, I_i, w)$$
$$\geq g_c(Q', Q_{1/2} \setminus \{I_i\}, w) + g_c(Q \setminus \{I_i\}, I_i, w),$$

where the first inequality holds by definition of g_c and the second inequality holds because $Q \setminus \{I_i\} \supseteq Q' \cup Q_{1/2} \setminus \{I_i\}$ implies $g_c(Q' \cup Q_{1/2} \setminus \{I_i\}, I_i, w) \geq g_c(Q \setminus \{I_i\}, I_i, w)$.

In an iteration of the for-loop with $g = g_c$, the current value w only increases, which implies $w' \leq w$. Since the greedy value g_c only increases for increasing values w, we get $g_c(Q', Q_{1/2} \setminus \{I_i\}, w) \geq g_c(Q', Q_{1/2} \setminus \{I_i\}, w')$ and, therefore $d = g_c(Q', Q_{1/2}, w) \geq g_c(Q', Q_{1/2} \setminus \{I_i\}, w') + g_c(Q \setminus \{I_i\}, I_i, w)$. Plugging in $\bar{d} = g_c(Q', Q_{1/2} \setminus \{I_i\}, w')$ and $g_c(Q \setminus \{I_i\}, I_i, w) \geq \bar{d}/2$ yields $d \geq 1.5 \cdot \bar{d}$.

Note that this only shows that d increases by a factor of 1.5 compared to $\bar{d}$, the old value computed in the previous execution of the repeat statement. Our goal was to show that d increases by factor 1.5 compared to the previous iteration of the inner-while loop. However, since the value w only increases, the greedy values g_c only increase. This implies that $\bar{d}$ can only be larger than the corresponding value at the end of the previous iteration of the inner while-loop. Thus, we have shown that d increases by a factor of 1.5 compared to its value at the end of the previous iteration of the inner-while loop. This implies $\sum_k \mathbb{P}[\bar{A}_{jk} = 1] \leq \lceil \log_{1.5} (m \cdot (2 \cdot \max_{I_i \in \mathcal{I}}(U_i - L_i))/s_{\min}) \rceil$.

Upper bound for $\sum_k \mathbb{P}[\bar{B}_{jk} = 1]$. Next, we use similar arguments to show $\sum_k \mathbb{P}[\bar{B}_{jk} = 1] \leq \lceil \log_2(m) \rceil$. Consider an iteration j of the outer while-loop.

For a fixed realization, we first show that, assuming $g = \bar{g}_s$, the value d increases by a factor of at least 2 in each iteration of the inner while-loop (except possibly the first and last ones) during iteration j of the outer while-loop.

By Line 6, iterations with $g = \bar{g}_s$ only consider values w with $b'_S(Q', w) < 1$ for all $S \in \mathcal{S}$, where Q' is the set of intervals queried before the start of the current execution of the inner while-loop. For an interval I_i with $(w_i - L_i) \geq \frac{1}{2} \cdot (U_i - L_i)$, we have $a'_i = \frac{2 \cdot (w_i - L_i)}{s_{s_{\min}}} \geq \frac{U_i - L_i}{s_{\min}} \geq 1$. This implies that adding I_i to the solution satisfies all constraints for sets S with $I_i \in S$ (at least for all values w with $b'_S(Q', w) < 1$ for all $S \in \mathcal{S}$). Thus, the greedy value g_s of I_i is then equal to the optimistic greedy value $\bar{g}_s$ as even if $a'_i = \gamma(U_i - L_i)$ the interval cannot satisfy more constraints.

Furthermore, as long as we only consider values w with $b'_S(Q', w) < 1$ for all $S \in \mathcal{S}$, the greedy values g_s and $\bar{g}_s$ of intervals I_i with $(w_i - L_i) \geq \frac{1}{2} \cdot (U_i - L_i)$ can only increase with increasing w and never increase for decreasing w. The reason for this is that such an increase in w can only lead to *more* constraints becoming *not* satisfied for the right-hand sides $b'_S(Q', w)$. Thus, the number of constraints that adding I_i can satisfy only increases. This also means, that the value w as used by the algorithm will never decrease during the current iteration of the outer while-loop with $g = \bar{g}_s$.

Using these observations, we can essentially repeat the proof of the previous case to show that the value d increases by a factor of at least 2 in each iteration of the inner while-loop with $g = \bar{g}_s$.

Let I_i be the interval queried in the last iteration of the repeat-statement within such an iteration of the inner while-loop, let Q denote the set of I_i and all intervals that were queried before I_i, let Q' denote the set of all intervals queried before the current execution of the inner while-loop was started, and let $Q_{1/2} = \{I_j \in Q \setminus Q' \mid w_j - L_j \geq \frac{U_j - L_j}{2}\}$. Furthermore, let w denote the value of Line 10 computed before I_i was queried in the following line. Note that this notation matches the one used in the algorithm at the execution of Line 12 directly after I_i was queried.

Let $\bar{d}$ denote the value d computed by the algorithm *before* I_i was queried. That is, $\bar{d} = g_s(Q', Q_{1/2} \setminus \{I_i\}, w')$ for the value w' that was computed in the previous execution of Line 10. By choice of I_i, we have $\bar{g}_s(Q \setminus \{I_i\}, I_i, w) \geq \bar{d}$ and $g_s(Q \setminus \{I_i\}, I_i, w) = \bar{g}_s(Q \setminus \{I_i\}, I_i, w) \geq \bar{d}$ (since $w_i - L_i \geq (U_i - L_i)/2$ and as argued above).

The value d computed after querying I_i is defined as

$$\begin{aligned} d = g_s(Q', Q_{1/2}, w) &\geq g_s(Q', Q_{1/2} \setminus \{I_i\}, w) + g_s(Q' \cup Q_{1/2} \setminus \{I_i\}, I_i, w) \\ &\geq g_s(Q', Q_{1/2} \setminus \{I_i\}, w) + g_s(Q \setminus \{I_i\}, I_i, w). \end{aligned}$$

Here, the first inequality holds by definition of g_s, definition of $Q_{1/2}$, and by the assumption that we only consider values w with $b'_S(Q', w) < 1$ for all $S \in \mathcal{S}$. The second inequality holds because $Q \setminus \{I_i\} \supseteq Q' \cup Q_{1/2} \setminus \{I_i\}$ implies $g_s(Q' \cup Q_{1/2} \setminus \{I_i\}, I_i, w) \geq g_s(Q \setminus \{I_i\}, I_i, w)$. Note that this implication only holds under the assumption that $b'_S(Q', w) < 1$ for all $S \in \mathcal{S}$ and for intervals I_i with $a'_i \geq 1$.

As argued above, the value w used by the algorithm only increases, i.e., $w' \leq w$. Furthermore, again as argued above, the greedy value g_s of intervals I_j with $a'_j \geq 1$ only increases for increasing values w. Thus, we get $g_s(Q', Q_{1/2} \setminus \{I_i\}, w) \geq g_s(Q', Q_{1/2} \setminus \{I_i\}, w')$ and, therefore, $d = g_s(Q', Q_{1/2}, w) \geq g_s(Q', Q_{1/2} \setminus \{I_i\}, w') + g_s(Q \setminus \{I_i\}, I_i, w)$. Plugging in $\bar{d} = g_s(Q', Q_{1/2} \setminus \{I_i\}, w')$ and $g_s(Q \setminus \{I_i\}, I_i, w) \geq \bar{d}$ yields $d \geq 2 \cdot \bar{d}$. This implies that d increased by a factor of at least 2 compared to its old value at the end of the previous iteration of the inner while-loop. As the greedy value g_s is upper bounded by m, we get $\sum_k \mathbb{P}[\bar{B}_{jk} = 1] \leq \lceil \log_2(m) \rceil$. □

The Lemmas 6.5.2 to 6.5.4 imply that Algorithm 21 satisfies Definition 6.3.6 for $\alpha = 2$, $\gamma = 2/s_{\min}$ and $\beta = \frac{1}{\tau}(\lceil \log_{1.5}(m \cdot 2(\max_{I_i \in \mathcal{I}}(U_i - L_i))/s_{\min}) \rceil + \lceil \log_2(m) \rceil)$. Thus, Theorem 6.3.7 implies Theorem 6.5.1.

6.6 The Maximization Variant of MinSet

Consider the set selection problem MaxSet, which has the same input as MinSet, but now the goal is to determine a set of *maximum weight* and to determine the corresponding weight.

In MaxSet, a set $Q \subseteq \mathcal{I}$ is *feasible* if all non-trivial elements of a set S^* of maximum weight w^* are in Q and $U_S(Q) \leq w^*$ for all sets $S \neq S^*$. If Q would not contain all non-trivial elements of some set S^* of maximum weight, then the maximum weight w^* would still be unknown and the problem would not be solved yet. If $U_S(Q) \geq w^*$ for some S, then both S^* and S could still be of maximum value or not and we have not yet determined the set of maximum value. Thus, the problem would not be solved yet. Our goal is again to adaptively query a feasible query set and minimize the number of queries.

In the following, we briefly sketch why all our results on MinSet transfer to MaxSet. Analogous to MinSet, one can show that the following ILP characterizes the problem. That is, a query set Q is feasible if and only if vector x with $x_i = 1$ if $I_i \in Q$ and $x_i = 0$ otherwise is feasible for the ILP.

$$\begin{array}{lll} \min & \sum_{I_i \in \mathcal{I}} x_i & \\ \text{s.t.} & \sum_{I_i \in S} x_i \cdot (U_i - w_i) \geq (U_S - w^*) & \forall S \in \mathcal{S} \\ & x_i \in \{0, 1\} & \forall I_i \in \mathcal{I} \end{array} \qquad (\text{MaxSetIP})$$

In the offline setting, the ILP is the exact same special case of the multiset multicover problem as (MinSetIP), which suffices to observe that all our observations on offline MinSet translate to offline MaxSet. Under uncertainty, the only difference to MinSet is, that, in contrast to (MinSetIP), a small weight w_i leads to a larger coefficient $(U_i - w_i)$ and a small weight w^* leads to larger right-hand sides. Using the inverse balancing parameter $\bar{\tau} = \min_{I_i \in \mathcal{I}} \bar{\tau}_i$ with $\bar{\tau}_i = \mathbb{P}[w_i \leq \frac{U_i+L_i}{2}]$, it is not hard to see, that, even under uncertainty, MINSET and MAXSET are essentially the same problem. Thus, all our results translate.

6.7 Concluding Remarks

In this chapter, we provided the first results for MINSET under stochastic explorable uncertainty by exploiting a connection to a covering problem and extending techniques for solving covering problems to our setting with uncertainty.

Since our results, in expectation, break adversarial lower bound instances for a number of interesting combinatorial problems under explorable uncertainty, e.g., matching, knapsack, solving ILPs [Mei18], we hope that our techniques lay the foundation for solving more general problems. In particular if we consider the mentioned problems with uncertainty in the cost coefficients and our goal is to query elements until we can identify an optimal solution to the underlying problem, then all these problems admit the same connection to covering problems as MINSET and can also be written as covering ILPs with uncertain coefficients and right-hand sides. The difference to MINSET is that the number of constraints in the covering representation for these problems might be exponential in the input size of the underlying optimization problem. In a sense, each feasible solution for the underlying optimization problem would define a constraint in the covering representation. This means that using the results of this chapter as a blackbox to solve such optimization problems under explorable uncertainty does not yield sublinear competitive ratios as the number of constraints becomes too large. For future research, we suggest to investigate whether one can exploit the additional structure in these constraints to still achieve improved competitive ratios.

With respect to the results of this chapter, we leave open whether the second $\log$ factor in our main result, Theorem 6.5.1, is necessary. Furthermore, the best competitive ratio achievable in exponential running time also remains open. Finally, we remark that we expect our algorithms to be parameterizable by the choice of the balancing parameter. We defined the parameter as the probability that the precise weight is larger than the center of the corresponding interval. Alternatively, one could pick any fraction of the interval, say one third, as a threshold and define the parameter as the probability that the precise weight lies outside the first third of its interval. As the only parts of our algorithm that use the definition of the balancing parameter are the termination criteria of the repeat-statements, the definition of the scaling factor γ and the definition of the set $Q_{1/2}$ in Algorithm 21, we expect that adjusting these parts to a different balancing parameter suffices to make our algorithms work for such parameters. The choice of the threshold for the balancing parameter would then influence the constant factor within the competitive ratios.

References

[AAK19] Arpit Agarwal, Sepehr Assadi, and Sanjeev Khanna. *Stochastic Submodular Cover with Limited Adaptivity*. In: *SODA*. SIAM, 2019, pp. 323–342. DOI: 10.1137/1.9781611975482.21.

[Abd+20] Mohamed Abdel-Nasser, Karar Mahmoud, Osama A. Omer, Matti Lehtonen, and Domenec Puig. *Link quality prediction in wireless community networks using deep recurrent neural networks*. In: *Alexandria Engineering Journal* 59.5 (2020), pp. 3531–3543. DOI: https://doi.org/10.1016/j.aej.2020.05.037.

[AD23] Jonatha Anselmi and Josu Doncel. *Load Balancing with Job-Size Testing: Performance Improvement or Degradation?* In: *CoRR* abs/2304.00899 (2023). DOI: 10.48550/arXiv.2304.00899.

[AE20] Susanne Albers and Alexander Eckl. *Explorable Uncertainty in Scheduling with Non-uniform Testing Times*. In: *WAOA*. Vol. 12806. Lecture Notes in Computer Science. Springer, 2020, pp. 127–142. DOI: 10.1007/978-3-030-80879-2_9.

[AE21] Susanne Albers and Alexander Eckl. *Scheduling with Testing on Multiple Identical Parallel Machines*. In: *WADS*. Vol. 12808. Lecture Notes in Computer Science. Springer, 2021, pp. 29–42. DOI: 10.1007/978-3-030-83508-8_3.

[AKL19] Sepehr Assadi, Sanjeev Khanna, and Yang Li. *The Stochastic Matching Problem with (Very) Few Queries*. In: *ACM Trans. Economics and Comput.* 7.3 (2019), 16:1–16:19. DOI: 10.1145/3355903.

[Alo+09] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. *The Online Set Cover Problem*. In: *SIAM J. Comput.* 39.2 (2009), pp. 361–370. DOI: 10.1137/060661946.

[ALT21] Yossi Azar, Stefano Leonardi, and Noam Touitou. *Flow time scheduling with uncertain processing time*. In: *STOC*. ACM, 2021, pp. 1070–1080. DOI: 10.1145/3406325.3451023.

[ALT22] Yossi Azar, Stefano Leonardi, and Noam Touitou. *Distortion-Oblivious Algorithms for Minimizing Flow Time*. In: *SODA*. SIAM, 2022, pp. 252–274. DOI: 10.1137/1.9781611977073.13.

[Ang+20] Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc P. Renault. *Online Computation with Untrusted Advice*. In: *ITCS*. Vol. 151. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 52:1–52:15. DOI: 10.4230/LIPIcs.ITCS.2020.52.

[Ant+20] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. *Secretary and Online Matching Problems with Machine Learned Advice*. In: *NeurIPS*. 2020.

[Ant+23] Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. *Online Metric Algorithms with Untrusted Predictions*. In: *ACM Trans. Algorithms* 19.2 (2023), 19:1–19:34. DOI: 10.1145/3582689.

[APT22] Yossi Azar, Debmalya Panigrahi, and Noam Touitou. *Online Graph Algorithms with Predictions*. In: *SODA*. SIAM, 2022, pp. 35–66. DOI: 10.1137/1.9781611977073.3.

[Ara+18] Luciana Arantes, Evripidis Bampis, Alexander V. Kononov, Manthos Letsios, Giorgio Lucarelli, and Pierre Sens. *Scheduling under Uncertainty: A Query-based Approach*. In: *IJCAI*. ijcai.org, 2018, pp. 4646–4652. DOI: 10.24963/ijcai.2018/646.

[Bam+21] Evripidis Bampis, Christoph Dürr, Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. *Orienting (Hyper)graphs Under Explorable Stochastic Uncertainty*. In: *ESA*. Vol. 204. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 10:1–10:18. DOI: 10.4230/LIPIcs.ESA.2021.10.

[Bam+22] Evripidis Bampis, Konstantinos Dogeas, Alexander V. Kononov, Giorgio Lucarelli, and Fanny Pascual. *Scheduling with Untrusted Predictions*. In: *IJCAI*. ijcai.org, 2022, pp. 4581–4587. DOI: 10.24963/ijcai.2022/636.

[Ban+12] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. *When LP Is the Cure for Your Matching Woes: Improved Bounds for Stochastic Matchings*. In: *Algorithmica* 63.4 (2012), pp. 733–762. DOI: 10.1007/s00453-011-9511-8.

[Bar+04] Reuven Bar-Yehuda, Keren Bendel, Ari Freund, and Dror Rawitz. *Local Ratio: A Unified Framework for Approximation Algorithms. In Memoriam: Shimon Even 1935-2004*. In: *ACM Comput. Surv.* 36.4 (2004), pp. 422–463. ISSN: 0360-0300. DOI: 10.1145/1041680.1041683.

[BBD22] Soheil Behnezhad, Avrim Blum, and Mahsa Derakhshan. *Stochastic Vertex Cover with Few Queries*. In: *SODA*. SIAM, 2022, pp. 1808–1846. DOI: 10.1137/1.9781611977073.73.

[BC12] Sébastien Bubeck and Nicoló Cesa-Bianchi. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. In: *Found. Trends Mach. Learn.* 5.1 (2012), pp. 1–122. DOI: 10.1561/2200000024.

[BE98] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[Beh+19] Soheil Behnezhad, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. *Stochastic Matching with Few Queries: New Algorithms and Tools*. In: *SODA*. SIAM, 2019, pp. 2855–2874. DOI: 10.1137/1.9781611975482.177.

[BGN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Vol. 28. Princeton Series in Applied Mathematics. Princeton University Press, 2009. URL: https://doi.org/10.1515/9781400831050.

[BL11] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. 2nd. New York, NY, USA: Springer-Verlag, 2011.

[Blu+20] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. *Ignorance Is Almost Bliss: Near-Optimal Stochastic Matching with Few Queries*. In: *Oper. Res.* 68.1 (2020), pp. 16–34. DOI: 10.1287/opre.2019.1856.

[BLW86] Norman Biggs, E Keith Lloyd, and Robin J Wilson. *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[BN15] Nikhil Bansal and Viswanath Nagarajan. *On the adaptivity gap of stochastic orienteering*. In: *Math. Program.* 154.1-2 (2015), pp. 145–172. DOI: 10.1007/s10107-015-0927-9.

[Bru+05] Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. *Efficient Update Strategies for Geometric Computing with Uncertainty*. In: *Theory Comput. Syst.* 38.4 (2005), pp. 411–423. DOI: 10.1007/s00224-004-1180-4.

[CC07] Miroslav Chlebik and Janka Chlebíková. *The Complexity of Combinatorial Optimization Problems on d-Dimensional Boxes*. In: *SIAM J. Discret. Math.* 21.1 (2007), pp. 158–169. DOI: 10.1137/050629276.

[Cha+21] Steven Chaplick, Magnús M. Halldórsson, Murilo S. de Lima, and Tigran Tonoyan. *Query minimization under stochastic uncertainty*. In: *Theor. Comput. Sci.* 895 (2021), pp. 75–95. DOI: 10.1016/j.tcs.2021.09.032.

[Che+09] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. *Approximating Matches Made in Heaven*. In: *ICALP (1)*. Vol. 5555. Lecture Notes in Computer Science. Springer, 2009, pp. 266–278. DOI: 10.1007/978-3-642-02927-1_23.

[Chv79] Vasek Chvátal. *A Greedy Heuristic for the Set-Covering Problem*. In: *Math. Oper. Res.* 4.3 (1979), pp. 233–235. DOI: 10.1287/moor.4.3.233.

[DGV08] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. *Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity*. In: *Math. Oper. Res.* 33.4 (2008), pp. 945–964. DOI: 10.1287/moor.1080.0330.

[DHK16] Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. *Approximation Algorithms for Stochastic Submodular Set Cover with Applications to Boolean Function Evaluation and Min-Knapsack*. In: *ACM Trans. Algorithms* 12.3 (2016), 42:1–42:28. DOI: 10.1145/2876506.

[Dob82] Gregory Dobson. *Worst-Case Analysis of Greedy Heuristics for Integer Programming with Nonnegative Data*. In: *Math. Oper. Res.* 7.4 (1982), pp. 515–531. DOI: 10.1287/moor.7.4.515.

[DS14] Irit Dinur and David Steurer. *Analytical approach to parallel repetition*. In: *STOC*. ACM, 2014, pp. 624–633. DOI: 10.1145/2591796.2591884.

[Dür+20] Christoph Dürr, Thomas Erlebach, Nicole Megow, and Julie Meißner. *An Adversarial Model for Scheduling with Testing*. In: *Algorithmica* 82.12 (2020), pp. 3630–3675. DOI: 10.1007/s00453-020-00742-2.

[Düt+21] Paul Dütting, Silvio Lattanzi, Renato Paes Leme, and Sergei Vassilvitskii. *Secretaries with Advice*. In: *EC*. ACM, 2021, pp. 409–429. DOI: 10.1145/3465456.3467623.

[Ebe+22] Franziska Eberle, Alexander Lindermayr, Nicole Megow, Lukas Nölke, and Jens Schlöter. *Robustification of Online Graph Exploration Methods*. In: *AAAI*. AAAI Press, 2022, pp. 9732–9740.

[EH14] Thomas Erlebach and Michael Hoffmann. *Minimum Spanning Tree Verification Under Uncertainty*. In: *WG*. Vol. 8747. Lecture Notes in Computer Science. Springer, 2014, pp. 164–175. DOI: 10.1007/978-3-319-12340-0_14.

[EH15] Thomas Erlebach and Michael Hoffmann. *Query-Competitive Algorithms for Computing with Uncertainty*. In: *Bull. EATCS* 116 (2015).

[EHK16] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. *Query-competitive algorithms for cheapest set problems under uncertainty*. In: *Theor. Comput. Sci.* 613 (2016), pp. 51–64. DOI: 10.1016/j.tcs.2015.11.025.

[EHL23] Thomas Erlebach, Michael Hoffmann, and Murilo Santos de Lima. *Round-Competitive Algorithms for Uncertainty Problems with Parallel Queries*. In: *Algorithmica* 85.2 (2023), pp. 406–443. DOI: 10.1007/s00453-022-01035-6.

[Erl+20] Thomas Erlebach, Michael Hoffmann, Murilo S. de Lima, Nicole Megow, and Jens Schlöter. *Untrusted Predictions Improve Trustable Query Policies*. In: *CoRR* abs/2011.07385 (2020).

[Erl+22] Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. *Learning-Augmented Query Policies for Minimum Spanning Tree with Uncertainty*. In: *ESA*. Vol. 244. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 49:1–49:18. DOI: 10.4230/LIPIcs.ESA.2022.49.

[Erl+23] Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. *Sorting and Hypergraph Orientation under Uncertainty with Predictions*. In: *CoRR* abs/2305.09245 (2023). DOI: 10.48550/arXiv.2305.09245.

[Fed+03] Tomás Feder, Rajeev Motwani, Rina Panigrahy, Chris Olston, and Jennifer Widom. *Computing the Median with Uncertainty*. In: *SIAM J. Comput.* 32.2 (2003), pp. 538–547. DOI: 10.1137/S0097539701395668.

[Fed+07] Tomás Feder, Rajeev Motwani, Liadan O'Callaghan, Chris Olston, and Rina Panigrahy. *Computing shortest paths with uncertainty*. In: *J. Algorithms* 62.1 (2007), pp. 1–18. DOI: 10.1016/j.jalgor.2004.07.005.

[FMM20] Jacob Focke, Nicole Megow, and Julie Meißner. *Minimum Spanning Tree under Explorable Uncertainty in Theory and Experiments*. In: *ACM J. Exp. Algorithmics* 25 (2020), pp. 1–20. DOI: 10.1145/3422371.

[Fra11] András Frank. *Connections in Combinatorial Optimization*. Oxford University Press, USA, 2011.

[GGN21] Rohan Ghuge, Anupam Gupta, and Viswanath Nagarajan. *The Power of Adaptivity for Stochastic Submodular Cover*. In: *ICML*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 3702–3712.

[GGW11] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed Bandit Allocation Indices*. 2nd. Wiley, 2011.

[GK11] Daniel Golovin and Andreas Krause. *Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization*. In: *J. Artif. Intell. Res.* 42 (2011), pp. 427–486. DOI: 10.1613/jair.3278.

[GKL23] Anupam Gupta, Gregory Kehne, and Roie Levin. *Set Covering with Our Eyes Wide Shut*. In: *CoRR* abs/2304.02063 (2023). DOI: 10.48550/arXiv.2304.02063.

[GN13] Anupam Gupta and Viswanath Nagarajan. *A Stochastic Probing Problem with Applications*. In: *IPCO*. Vol. 7801. Lecture Notes in Computer Science. Springer, 2013, pp. 205–216. DOI: 10.1007/978-3-642-36694-9_18.

[GNS16] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. *Algorithms and Adaptivity Gaps for Stochastic Probing*. In: *SODA*. SIAM, 2016, pp. 1731–1747. DOI: 10.1137/1.9781611974331.ch120.

[Goe+15] Marc Goerigk, Manoj Gupta, Jonas Ide, Anita Schöbel, and Sandeep Sen. *The robust knapsack problem with queries*. In: *Computers & Operations Research* 55 (2015), pp. 12–22. DOI: 10.1016/j.cor.2014.09.010.

[Gon+22] Mingyang Gong, Randy Goebel, Guohui Lin, and Eiji Miyano. *Improved approximation algorithms for non-preemptive multiprocessor scheduling with testing*. In: *J. Comb. Optim.* 44.1 (2022), pp. 877–893. DOI: 10.1007/s10878-022-00865-y.

[GP19] Sreenivas Gollapudi and Debmalya Panigrahi. *Online Algorithms for Rent-Or-Buy with Expert Advice*. In: *ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2319–2327.

[Gra+13] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. *Set Covering with Our Eyes Closed*. In: *SIAM J. Comput.* 42.3 (2013), pp. 808–830. DOI: 10.1137/100802888.

[GSS16] Manoj Gupta, Yogish Sabharwal, and Sandeep Sen. *The Update Complexity of Selection and Related Problems*. In: *Theory of Computing Systems* 59.1 (2016), pp. 112–132. DOI: 10.1007/s00224-015-9664-y.

[Gup+15] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. *Running Errands in Time: Approximation Algorithms for Stochastic Orienteering*. In: *Math. Oper. Res.* 40.1 (2015), pp. 56–79. DOI: 10.1287/moor.2014.0656.

[Gup+19] Anupam Gupta, Haotian Jiang, Ziv Scully, and Sahil Singla. *The Markovian Price of Information*. In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2019, pp. 233–246. DOI: 10.1007/978-3-030-17953-3_18.

[GV06a] Michel X. Goemans and Jan Vondrák. *Covering minimum spanning trees of random subgraphs*. In: *Random Struct. Algorithms* 29.3 (2006), pp. 257–276. DOI: 10.1002/rsa.20115.

[GV06b] Michel X. Goemans and Jan Vondrák. *Stochastic Covering and Adaptivity*. In: *LATIN*. Vol. 3887. Lecture Notes in Computer Science. Springer, 2006, pp. 532–543. DOI: 10.1007/11682462_50.

[Hal02] Eran Halperin. *Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs*. In: *SIAM Journal on Computing* 31.5 (2002), pp. 1608–1623. DOI: 10.1137/S0097539700381097.

[HL21] Magnús M. Halldórsson and Murilo Santos de Lima. *Query-competitive sorting with uncertainty*. In: *Theor. Comput. Sci.* 867 (2021), pp. 50–67. DOI: 10.1016/j.tcs.2021.03.021.

[Hoe94] Wassily Hoeffding. *Probability inequalities for sums of bounded random variables*. In: *The collected works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[Hof+08] Michael Hoffmann, Thomas Erlebach, Danny Krizanc, Matús Mihalák, and Rajeev Raman. *Computing Minimum Spanning Trees with Uncertainty*. In: *STACS*. Vol. 1. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2008, pp. 277–288. DOI: 10.4230/LIPIcs.STACS.2008.1358.

[INZ16] Sungjin Im, Viswanath Nagarajan, and Ruben van der Zwaan. *Minimum Latency Submodular Cover*. In: *ACM Trans. Algorithms* 13.1 (2016), 13:1–13:28. DOI: 10.1145/2987751.

[Kah91] Simon Kahan. *A Model for Data in Motion*. In: *STOC*. ACM, 1991, pp. 267–277. DOI: 10.1145/103418.103449.

[KNN17] Prabhanjan Kambadur, Viswanath Nagarajan, and Fatemeh Navidi. *Adaptive Submodular Ranking*. In: *IPCO*. Vol. 10328. Lecture Notes in Computer Science. Springer, 2017, pp. 317–329. DOI: 10.1007/978-3-319-59250-3_26.

[KT01] Sanjeev Khanna and Wang Chiew Tan. *On Computing Functions with Uncertainty*. In: *PODS*. ACM, 2001. DOI: 10.1145/375551.375577.

[Kum+19] Ravi Kumar, Manish Purohit, Aaron Schild, Zoya Svitkina, and Erik Vee. *Semi-Online Bipartite Matching*. In: *ITCS*. Vol. 124. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019, 50:1–50:20. DOI: 10.4230/LIPIcs.ITCS.2019.50.

[KY01] Stavros G. Kolliopoulos and Neal E. Young. *Tight Approximation Results for General Covering Integer Programs*. In: *FOCS*. IEEE Computer Society, 2001, pp. 522–528. DOI: 10.1109/SFCS.2001.959928.

[KY05] Stavros G. Kolliopoulos and Neal E. Young. *Approximation algorithms for covering/packing integer programs*. In: *J. Comput. Syst. Sci.* 71.4 (2005), pp. 495–505. DOI: 10.1016/j.jcss.2005.05.002.

[Lat+20] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. *Online Scheduling via Learned Weights*. In: *SODA*. SIAM, 2020, pp. 1859–1877. DOI: 10.1137/1.9781611975994.114.

[LM22] Alexander Lindermayr and Nicole Megow. *Permutation Predictions for Non-Clairvoyant Scheduling*. In: *SPAA*. ACM, 2022, pp. 357–368. DOI: 10.1145/3490148.3538579.

[LM23] Alexander Lindermayr and Nicole Megow. *Repository of papers on algorithms with predictions*. 2023. URL: https://algorithms-with-predictions.github.io/.

[LMS19] Retsef Levi, Thomas L. Magnanti, and Yaron Shaposhnik. *Scheduling with Testing*. In: *Manag. Sci.* 65.2 (2019), pp. 776–793.

[LMS22] Alexander Lindermayr, Nicole Megow, and Bertrand Simon. *Double Coverage with Machine-Learned Advice*. In: *ITCS*. Vol. 215. LIPIcs. 2022, 99:1–99:18. DOI: 10.4230/LIPIcs.ITCS.2022.99.

[Lu+21] Pinyan Lu, Xuandi Ren, Enze Sun, and Yubo Zhang. *Generalized Sorting with Predictions*. In: *SOSA*. SIAM, 2021, pp. 111–117. DOI: 10.1137/1.9781611976496.13.

[LV21] Thodoris Lykouris and Sergei Vassilvitskii. *Competitive Caching with Machine Learned Advice*. In: *J. ACM* 68.4 (2021), 24:1–24:25. DOI: 10.1145/3447579.

[LX21] Shi Li and Jiayi Xian. *Online Unrelated Machine Load Balancing with Predictions Revisited*. In: *ICML*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 6523–6532. URL: https://proceedings.mlr.press/v139/li21w.html.

[Ma18] Will Ma. *Improvements and Generalizations of Stochastic Knapsack and Markovian Bandits Approximation Algorithms*. In: *Math. Oper. Res.* 43.3 (2018), pp. 789–812. DOI: 10.1287/moor.2017.0884.

[MÇ22] Corinna Mathwieser and Eranda Çela. *Special Cases of the Minimum Spanning Tree Problem under Explorable Edge and Vertex Uncertainty*. In: *CoRR* abs/2211.15611 (2022).

[Mei18] Julie Meißner. *Uncertainty Exploration: Algorithms, Competitive Analysis, and Computational Experiments*. PhD thesis. Technischen Universität Berlin, 2018. DOI: 10.14279/depositonce-7327.

[Mit20] Michael Mitzenmacher. *Scheduling with Predictions and the Price of Misprediction*. In: *Proceedings of ITCS*. Vol. 151. LIPIcs. 2020, 14:1–14:18. DOI: 10.4230/LIPIcs.ITCS.2020.14.

[MMS17] Nicole Megow, Julie Meißner, and Martin Skutella. *Randomization Helps Computing a Minimum Spanning Tree under Uncertainty*. In: *SIAM Journal on Computing* 46.4 (2017), pp. 1217–1240. DOI: 10.1137/16M1088375.

[MS19] Arturo I. Merino and José A. Soto. *The Minimum Cost Query Problem on Matroids with Uncertainty Areas*. In: *Proceedings of ICALP*. Vol. 132. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 83:1–83:14.

[MS21] Nicole Megow and Jens Schlöter. *Explorable Uncertainty Meets Decision-Making in Logistics*. In: *Dynamics in Logistics: Twenty-Five Years of Interdisciplinary Logistics Research in Bremen, Germany*. Springer, 2021, pp. 35–56.

[MS23] Nicole Megow and Jens Schlöter. *Set Selection Under Explorable Stochastic Uncertainty via Covering Techniques*. In: *IPCO*. Vol. 13904. Lecture Notes in Computer Science. Springer, 2023, pp. 319–333. DOI: 10.1007/978-3-031-32726-1_23.

[MV17] Andres Muñoz Medina and Sergei Vassilvitskii. *Revenue Optimization with Approximate Bid Predictions*. In: *NIPS*. 2017, pp. 1858–1866.

[MY20] Takanori Maehara and Yutaro Yamaguchi. *Stochastic packing integer programs with few queries*. In: *Math. Program.* 182.1 (2020), pp. 141–174. DOI: 10.1007/s10107-019-01388-x.

[NJ75] George L. Nemhauser and Leslie E. Trotter Jr. *Vertex packings: Structural properties and algorithms*. In: *Math. Program.* 8.1 (1975), pp. 232–248. DOI: 10.1007/BF01580444.

[NKN20] Fatemeh Navidi, Prabhanjan Kambadur, and Viswanath Nagarajan. *Adaptive Submodular Ranking and Routing*. In: *Oper. Res.* 68.3 (2020), pp. 856–877. DOI: 10.1287/opre.2019.1889.

[OW00] Chris Olston and Jennifer Widom. *Offering a Precision-Performance Tradeoff for Aggregation Queries over Replicated Data*. In: *VLDB*. Morgan Kaufmann, 2000, pp. 144–155.

[Pol74] S. Poljak. *A note on stable sets and colorings of graphs*. In: *Commentationes Mathematicae Universitatis Carolinae* 15.2 (1974), pp. 307–309. URL: https://eudml.org/doc/16622.

[PSK18] Manish Purohit, Zoya Svitkina, and Ravi Kumar. *Improving Online Algorithms via ML Predictions*. In: *NeurIPS*. 2018, pp. 9684–9693.

[Rob39] Herbert E. Robbins. *A Theorem on Graphs, with an Application to a Problem of Traffic Control*. In: *The American Mathematical Monthly* 46.5 (1939), pp. 281–283. URL: http://www.jstor.org/stable/2303897.

[Roh20] Dhruv Rohatgi. *Near-Optimal Bounds for Online Caching with Machine Learned Advice*. In: *SODA*. SIAM, 2020, pp. 1834–1845.

[Rou20] Tim Roughgarden, ed. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

[RV98] Sridhar Rajagopalan and Vijay V Vazirani. *Primal-dual RNC approximation algorithms for set cover and covering integer programs*. In: *SIAM Journal on Computing* 28.2 (1998), pp. 525–540. DOI: 10.1137/S0097539793260763.

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[Sch03] Alexander Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.

[Sha16] Yaron Shaposhnik. *Exploration vs. Exploitation: reducing uncertainty in operational problems*. PhD thesis. Massachusetts Institute of Technology, 2016.

[Sin18] Sahil Singla. *The Price of Information in Combinatorial Optimization*. In: *SODA*. SIAM, 2018, pp. 2523–2532. DOI: 10.1137/1.9781611975031.161.

[SS04] David B. Shmoys and Chaitanya Swamy. *Stochastic Optimization is (Almost) as easy as Deterministic Optimization*. In: *FOCS*. IEEE Computer Society, 2004, pp. 228–237. DOI: 10.1109/FOCS.2004.62.

[Tho33] William R. Thompson. *On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples*. In: *Biometrika* 25.3/4 (1933), pp. 285–294.

[Vad01] Salil P. Vadhan. *The Complexity of Counting in Sparse, Regular, and Planar Graphs*. In: *SIAM J. Comput.* 31.2 (2001), pp. 398–427. DOI: 10.1137/S0097539797321602.

[Val84] Leslie G. Valiant. *A Theory of the Learnable*. In: *Commun. ACM* 27.11 (1984), pp. 1134–1142. DOI: 10.1145/1968.1972.

[Vap92] Vladimir Vapnik. *Principles of risk minimization for learning theory*. In: *Advances in neural information processing systems*. 1992, pp. 831–838.

[Vap99] Vladimir N Vapnik. *An overview of statistical learning theory*. In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999. DOI: 10.1109/72.788640.

[Vaz01] Vijay V Vazirani. *Approximation algorithms*. Vol. 1. Springer, 2001.

[Von07] Jan Vondrák. *Shortest-path metric approximation for random subgraphs*. In: *Random Struct. Algorithms* 30.1-2 (2007), pp. 95–104. DOI: 10.1002/rsa.20150.

[Weg67] G. Wegner. *Eigenschaften der Nerven homologisch–einfacher Familien im R^n*. PhD thesis. Universität Göttingen, 1967.

[Wei20] Alexander Wei. *Better and Simpler Learning-Augmented Online Caching*. In: *APPROX/RANDOM*. Vol. 176. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 60:1–60:17. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2020.60.

[Wei79] Martin Weitzman. *Optimal Search for the Best Alternative*. In: *Econometrica* 47.3 (1979), pp. 641–54.

[WGW22] Weina Wang, Anupam Gupta, and Jalani Williams. *Probing to Minimize*. In: *ITCS*. Vol. 215. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 120:1–120:23. DOI: 10.4230/LIPIcs.ITCS.2022.120.

[Wol82] Laurence A. Wolsey. *An analysis of the greedy algorithm for the submodular set covering problem*. In: *Comb.* 2.4 (1982), pp. 385–393. DOI: 10.1007/BF02579435.

[WZ20] Alexander Wei and Fred Zhang. *Optimal Robustness-Consistency tradeoffs for Learning-Augmented Online Algorithms*. In: *NeurIPS*. 2020.

[YG80] Mihalis Yannakakis and Fanica Gavril. *Edge Dominating Sets in Graphs*. In: *SIAM Journal on Applied Mathematics* 38.3 (1980), pp. 364–372. DOI: 10.1137/0138030.

Zusammenfassung

Bei der Lösung von Optimierungsproblemen in realen Anwendungen sind Unsicherheit in den Eingabedaten und unvollständige Informationen eine große Herausforderung. Man denke beispielsweise an schwankende Transportzeiten, unbekannte Ausführungszeiten von zu planenden Aufgaben, variable Parameter wie Bandbreite und Nachfrage, sich dynamisch ändernde Standorte von mobilen Agenten oder dezentralisierte Daten, die nur unregelmäßig aktualisiert werden.

Da Unsicherheit in einer Vielzahl von realen Anwendungen allgegenwärtig ist, wurden eine Reihe von mathematischen Methoden für die Modellierung von Unsicherheit in Optimierungsproblemen entwickelt. Eine erste wichtige Methode ist die *Online-Optimierung*, bei der Teile der Eingaben zunächst unbekannt sind und die fehlenden Daten im Laufe der Zeit nach und nach aufgedeckt werden. Wann immer neue Daten eintreffen, muss ein Optimierungsalgorithmus unwiderrufliche Entscheidungen darüber treffen, wie er mit den neuen Informationen umgehen soll (vgl. z.B. [BE98]). *Stochastische Optimierung* bezieht sich auf Probleme, in denen unsichere Eingangsdaten durch (bekannte oder unbekannte) Wahrscheinlichkeitsverteilungen modelliert werden. Ziel ist es, Lösungen zu finden, die entweder im Erwartungswert oder mit hoher Wahrscheinlichkeit eine hohe Qualität haben. Die tatsächlichen Realisierungen der unsicheren Teile der Eingabe werden in der Regel stufenweise nacheinander aufgedeckt (vgl. z.B. [BL11]). In der dritten weit verbreiteten Methode, der *robusten Optimierung*, wird die Unsicherheit in der Regel durch eine explizit oder implizit vorgegebene Menge von Szenarien modelliert, die möglicherweise eintreten könnten. Üblicherweise ist dann das Ziel, eine einzige Lösung zu finden, die für jedes mögliche Szenario eine hohe Qualität hat (vgl. z.B. [BGN09]).

Alle diese Modelle haben gemeinsam, dass die unsicheren Informationen entweder passiv im Laufe der Zeit oder überhaupt nicht offenbart werden. Optimierungsalgorithmen für Probleme in diesen Modellen müssen mit dieser Art von passiv offengelegter Unsicherheit umgehen und Entscheidungen auf der Grundlage unvollständiger Informationen treffen. Insbesondere haben sie nicht die Möglichkeit, auch nicht gegen zusätzliche Kosten, aktiv neue Informationen zu ermitteln, die bei der Lösung der Optimierungsaufgabe helfen. In einigen realen Anwendungen ist die Möglichkeit, unsichere Teile der Eingabe zu einem bestimmten Preis abzufragen, jedoch eine naheliegende Annahme. So können unsichere Ausführungszeiten von zu planenden Aufgaben durch weitere Analysen ermittelt werden (vgl. [Sha16] für ein Beispiel in der Instandhaltung mittels Fehleranalyse), variable Parameter wie die Bandbreite einer Netzwerkverbindung können gemessen werden, dynamisch wechselnde Standorte von sich bewegenden Agenten können durch zusätzliche Kommunikation [Kah91] ermittelt werden und dezentrale Daten können durch Abfragen an eine Master-Datenbank [OW00] aktualisiert werden.

Das Gebiet der *erforschbaren Unsicherheit* wurde in einer initialen Arbeit von Kahan [Kah91] eingeführt und betrachtet genau solche Szenarien, in denen unsichere Teile in der Eingabe eines Optimierungsproblems abgefragt werden können, um mehr Informationen zu erhalten. Da solche Abfragen kostspielig sind, besteht das Ziel darin, die Abfragekosten zu minimieren, die notwendig sind, um eine optimale (oder annähernde) Lösung für das zugrunde liegende Optimierungsproblem zu finden. Das Modell berücksichtigt insbesondere die Unsicherheit von numerischen Parametern der Eingabe. Anstatt Zugang zu den genauen

Werten dieser Parameter zu haben, werden uns *Unsicherheitsintervalle* gegeben. Diese Intervalle enthalten die genauen Werte der entsprechenden Parameter, wir haben initial aber keine Information darüber, wo innerhalb des Intervalls der Wert tatsächlich liegt. Da die optimale Lösung des zugrundeliegenden Optimierungsproblems von den unsicheren Parametern abhängen kann, ist es gegebenenfalls unmöglich eine optimale (oder annähernde) Lösung in Bezug auf die unsicheren Werte zu finden, ohne weitere Informationen zu erhalten. Daher besteht das Ziel im Gebiet der erforschbaren Unsicherheit darin, Algorithmen zu entwickeln, die unsichere Parameter so lange abfragen, bis die aufgedeckten Informationen ausreichen, um eine optimale Lösung für das zugrundeliegende Optimierungsproblem zu finden. Dabei sollen die Abfragekosten minimiert werden. Diese Dissertation und die Mehrzahl der bestehenden Forschungsarbeiten auf diesem Gebiet befassen sich mit *adaptiven Algorithmen* für Probleme mit erforschbarer Unsicherheit, die Ergebnisse früherer Abfragen bei der Entscheidung über die nächste Abfrage berücksichtigen können.

Betrachte als Beispiel das klassische *minimale Spannbaum-Problem*, bei dem ein ungerichteter Graph $G = (V, E)$ mit Kantengewichten w_e für alle $e \in E$ gegeben ist. Ziel ist es, einen *minimalen Spannbaum (MST)* zu finden, d.h. eine Teilmenge $T \subseteq E$ mit minimalem Gewicht $w(T) = \sum_{e \in T} w_e$, so dass der Teilgraph $G' = (V, T)$ zusammenhängend ist und keine Kreise enthält. Unter erforschbarer Unsicherheit sind die genauen Kantengewichte w_e zunächst unbekannt. Stattdessen haben wir nur Zugriff auf Unsicherheitsintervalle $I_e = (L_e, U_e)$, für alle $e \in E$, die garantiert die genauen Kantengewichte enthalten, d.h. $w_e \in I_e$. Das Ziel bleibt, einen MST für die unbekannten genauen Kantengewichte zu finden. Um einen solchen MST trotz der fehlenden Information über die Kantengewichte zu finden, können Kanten abgefragt werden, um das genaue Gewicht w_e einer Kante e zu Abfragekosten c_e zu ermitteln. Die Aufgabe besteht darin, Algorithmen zu entwerfen, die adaptiv Kanten abfragen, bis die abgefragten Informationen ausreichen, um einen MST für die genauen Gewichte zu bestimmen und dabei die gesamten Abfragekosten zu minimieren.

Bisher wurden Algorithmen für erforschbare Unsicherheit meist mit Hilfe von *Worst-Case Analysen* untersucht, bei denen wir davon ausgehen, dass Abfrageergebnisse in einer Art und Weise zurückgegeben werden, die zur schlechtesten Performanz des Algorithmus führt. Da es in der Regel Probleminstanzen gibt, die nicht gelöst werden können, ohne alle unsicheren Parameter abzufragen, werden Algorithmen für erforschbare Unsicherheit typischerweise auf instanzabhängige Weise mittels *competitive analysis* analysiert. Wir geben später eine formale Definition, aber wir sagen, dass ein Algorithmus *ρ-kompetitiv* für ein Problem unter erforschbarer Unsicherheit ist, wenn für jede Probleminstanz die Abfragekosten des Algorithmus um einen Faktor von höchstens ρ größer sind als die optimalen Abfragekosten für die konkrete Instanz. Das minimale ρ, für das ein Algorithmus ρ-kompetitiv ist, nennt man den *competitive ratio* des Algorithmus.

Die am meisten untersuchten Probleme im Gebiet der erforschbaren Unsicherheit sind Auswahlprobleme, z.B, die Auswahl des Minimums [Kah91], Sortierung [HL21], die Auswahl des k-kleinsten Elements [Kah91; Fed+03], die Auswahl eines minimalen Spannbaums [Hof+08; EH14; MMS17] und geometrische Probleme [Bru+05]. Für all diese Probleme gibt es konstant-kompetitive Algorithmen und untere Schranken an den compeitive ratio, die bessere algorithmen ausschließen. Außerdem haben diese Probleme gemeinsam, dass sie im Wesentlichen (aber auf nicht-triviale Weise) auf den Vergleich einzelner Unsicherheitsintervalle reduziert werden können und dass die unteren Schranken bereits aus sehr einfachen Probleminstanzen bestehen. Sobald zwei Mengen (Summen) von Unsicherheitsintervallen verglichen werden müssen, kann kein deterministischer Algorithmus einen besseren competitive ratio haben als $\Omega(n)$ [EHK16], wobei n die Anzahl der gegebenen Unsicherheitsintervalle ist. Diese untere Schranke von $\Omega(n)$ lässt sich auf kombinatorische Optimierungsprobleme unter erforschbarer Unsicherheit übertragen, wie z.B. die Berechnung des kürzesten Weges in einem

Graphen [Fed+07], das Rucksackproblem [Mei18] und Matchings [Mei18], und verhindert, dass wir nicht-triviale Ergebnisse für diese Probleme erzielen können.

Motiviert durch diese einfachen und starken unteren Schranken stellt diese Dissertation die Frage, ob Worst-Case-Abfrageergebnisse und keine zusätzlichen Informationen über die unsicheren Eingabeparameter (abgesehen von den Unsicherheitsintervallen) zu pessimistisch sind. Betrachten wir dazu erneut die initialen Anwendungsbeispiele. Die Qualität von Netzwerkverbindungen, die anhand von Kennzahlen wie Durchsatz und Zuverlässigkeit in einem drahtlosen Netzwerk gemessen wird, schwankt oft innerhalb eines bestimmten Intervalls. Die tatsächliche Qualität einer Verbindung kann durch eine neue Messung ermittelt werden. Wenn wir einen minimalen Spannbaum mit den Verbindungen aufbauen wollen, die derzeit die höchste Verbindungsqualität haben, und die zusätzlich benötigten Messungen minimieren wollen, dann liegt ein minimales Spannbaum-Problem unter erforschbarer Unsicherheit vor. Die Annahme, dass wir keine zusätzlichen Informationen über die Ergebnisse der Messungen haben, könnte jedoch zu pessimistisch sein, da man zum Beispiel Methoden des maschinellen Lernens verwenden kann, um die genaue Verbindungsqualität auf der Grundlage von früheren Messungen vorherzusagen [Abd+20]. Da diese Vorhersagen nicht immer genau sind, möchten wir trotzdem neue Messungen durchführen, um zu garantieren, dass wir einen minimalen Spannbaum in Bezug auf die aktuelle Verbindungsqualität finden. Die zusätzlichen Informationen in Form der Vorhersagen können dann trotzdem hilfreich dabei sein, die Verbindungen für zusätzliche Messungen auszuwählen. Als zweites Beispiel, betrachte die unsichere Position von mobilen Agenten. Der letzte bekannte frühere Standort und die maximale Bewegungsgeschwindigkeit eines solchen Agenten ergeben ein Unsicherheitsintervall, das garantiert den aktuellen Standort enthält. Mit Hilfe von statistischen Werkzeugen könnte man jedoch in der Lage sein, den wahrscheinlichsten wahren Standort des Agenten auf der Grundlage früherer Bewegungsdaten vorherzusagen.

In dieser Dissertation betrachten wir verschiedene Probleme unter erforschbarer Unsicherheit und analysieren sie mit Hilfe von Methoden, die über den Worst-Case hinausgehen. Mehrere Methoden für die Analyse von Algorithmen jenseits des Worst-Case wurden zum Beispiel in [Rou20] diskutiert. In dieser Arbeit betrachten wir Algorithmen mit Zugriff auf unzuverlässige Vorhersagen oder stochastische Informationen für Probleme unter erforschbarer Unsicherheit.

Im ersten Fall gehen wir davon aus, dass wir Zugang zu unzuverlässigen Vorhersagen über die unsicheren Werte haben. Angesichts des Aufschwungs der künstlichen Intelligenz und des maschinellen Lernens (ML) in den letzten Jahrzehnten scheint es naheliegend zu sein, Vorhersagen von guter Genauigkeit zu erwarten. Allerdings gibt es dafür keine Garantie, und die Vorhersagen könnten in einigen Fällen auch beliebig falsch sein. Unser Ziel ist es daher, Algorithmen zu entwickeln, die einen verbesserten competitive ratio erzielen wenn die Vorhersagen von guter Genauigkeit sind, und gleichzeitig die Leistungsgarantien von Algorithmen ohne Zugang zu Vorhersagen erfüllen, selbst wenn die Vorhersagen komplett falsch sind. Dies entspricht einem neueren Forschungstrend, der die Verwendung von nicht vertrauenswürdigen Vorhersagen für Online-Algorithmen in Betracht zieht; wir geben später einen Überblick über verwandte Arbeiten auf diesem Gebiet. Wie in [LV21; PSK18] eingeführt, ist ein Algorithmus α-*konsistent*, wenn er im Falle von korrekten Vorhersagen α-kompetitiv ist, und er ist β-*robust*, wenn er auch für beliebig schlechte Vorhersagen β-kompetitiv ist. Idealerweise wollen wir einen fließenden Übergang zwischen Konsistenz und Robustheit in Abhängigkeit von einem Fehlermaß, das die Genauigkeit der Vorhersagen quantifiziert, garantieren. In den Kapiteln 4 und 5 entwerfen wir Algorithmen mit Zugriff auf unzuverlässige Vorhersagen für mehrere Probleme unter erforschbarer Unsicherheit und analysieren sie anhand von Konsistenz, Robustheit und Fehlerabhängigkeit.

Ein weiteres Modell für die Analyse von Problemen unter erforschbarer Unsicherheit ist das stochastische Modell. Anstatt davon auszugehen, dass die Abfrageergebnisse so zurückgegeben werden, wie es für die Performanz des Algorithmus am schlimmsten ist, nehmen wir an, dass sie aus (bekannten oder unbekannten) Wahrscheinlichkeitsverteilungen über die entsprechenden Unsicherheitsintervalle gezogen werden. Im Gegensatz zu den unzuverlässigen Vorhersagen sind die stochastischen Informationen zuverlässig, und die Algorithmen werden im Hinblick auf das Verhältnis zwischen den *erwarteten* Abfragekosten eines Algorithmus und den erwarteten Kosten für die optimale Lösung analysiert. In den Kapiteln 3 und 6 betrachten wir Probleme unter erforschbarer Ungewissheit in verschiedenen stochastischen Modellen und entwerfen Algorithmen, die, im Erwartungswert, eine bessere Performanz haben als die unteren Schranken für Worst-Case Analysen.

Insgesamt zeigen die Ergebnisse dieser Arbeit, dass wir die unteren Schranken für mehrere Probleme unter erforschbarer Unsicherheit verbessern können, wenn wir diese Probleme über den Worst-Case hinaus analysieren. Um diese verbesserten Ergebnisse zu erreichen, entwerfen wir mehrere neue Algorithmen mit Hilfe von algorithmischen Techniken und Analysen, die bisher noch nicht für Probleme in diesem Bereich verwendet wurden. Wir hoffen, dass unsere Ergebnisse und technischen Beiträge den Grundstein für weitere Forschung zu Problemen unter erforschbarer Unsicherheit jenseits des Worst-Case legen.