Explorable Uncertainty Meets Decision-Making in Logistics

Nicole Megow and Jens Schlöter

Abstract Decision-making under uncertainty is a major challenge in logistics. Mathematical optimization has a long tradition in providing powerful methods for solving logistics problems. While classical optimization models for uncertainty in the input data do not consider the option to actively query the precise value of uncertain input elements, this option is in practice often available at a certain cost. The recent line of research on optimization under explorable uncertainty develops methods with provable performance guarantees for such scenarios. In this chapter, we highlight some recent results from the mathematical optimization perspective and outline the potential power of such model and techniques for solving logistics problems.

1 Uncertainty in Logistics

Uncertainty is a major challenge in effective decision-making in logistics and supply chain management (Wilding 1998). Uncertainty may refer to any lack of information about the supply chain, its environment or conditions, and the unpredictable impact of decisions (van der Vorst and Beulens 2002). There are numerous sources of uncertainty and dynamics in the system parameters; for a classification and survey, we refer to Simangunsong et al. (2012), Sanchez-Rodrigues et al. (2010). As illustrating examples consider varying transportation times depending on traffic conditions, weather, or disruptions; variable demand; deteriorating conditions of delivered goods; changing production capacity or speed due to aging or replacement of production units, etc.

In addition, digitalization and data-driven applications give permanent access to large amounts of data that can be used for planning and decision-making purposes.

N. Megow (✉) · J. Schlöter
Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany
e-mail: nicole.megow@uni-bremen.de; jschloet@uni-bremen.de

M. Freitag et al. (eds.), *Dynamics in Logistics*,
https://doi.org/10.1007/978-3-030-88662-2_2

However, these data need to be analyzed and processed to provide valuable insights, and moreover, they increase the level of dynamics and uncertainty significantly.

Mathematical optimization has a tradition in providing powerful methods for effectively planning logistics processes, such as routing and delivery, replenishment, and scheduling.

Due to the ubiquity of uncertainty (in fact, in all kinds of real-world applications), a plethora of research articles is devoted to decision-making under data uncertainty and optimization methods that are capable of handling large amounts of dynamically changing data. From the mathematical optimization perspective, there are three major frameworks for modeling uncertainty in the input data: stochastic optimization, online optimization, and robust optimization. Below, we sketch their general characteristics without reviewing the numerous individual settings and the respective literature.

Stochastic optimization generally refers to settings in which there is some randomness in the data. Typically, parts of the input data are modeled as random variables that follow a given probability distribution. Such probabilistic information may stem from statistical information, such as, travel time distributions from traffic surveillance or customer statistics. The goal is to find a solution that is (approximately) optimal in expectation. Different models define how adaptively an algorithm may proceed in making decisions and in what manner the true realization of uncertain data is revealed. Typically, we distinguish single-stage and multi-stage problems, but there are more dynamic models; see, e.g., the textbook by Birge and Louveaux (2011).

Online optimization refers to settings in which parts of the input are completely unknown a priori. The data is revealed incrementally one by one or over time, and an algorithm must make irrevocable decisions given only partial information; see, e.g., the book by Borodin and El-Yaniv (1998). As an example, consider the (often) unpredictable arrival of maintenance requests and a service provider who must immediately allocate and coordinate resources for maintenance and possibly replacement. It is typically impossible to give an algorithm that finds an optimal solution for all possible online inputs. Therefore, we evaluate the performance of algorithms by *competitive analysis*, a worst-case analysis that compares (implicitly), for any possible input instance, the cost of an algorithm's solution with the optimal cost.

Robust optimization is a framework in which, again, there are no probabilistic assumptions made. Typically, a set of possible input scenarios is given either explicitly or implicitly, e.g., by giving uncertainty intervals in which the true value of some unknown data lies. In the classical setting, we ask for a single solution that performs well in any possible input scenario. Hence, algorithms are absolutely non-adaptive and make decisions without any chance of adjusting them later. More recently, models have been proposed that allow some recovery actions once the scenario has been realized; see, e.g., the book by Ben-Tal et al. (2009).

In all these traditional frameworks for data uncertainty, optimization methods have to accept the incompleteness of input data. They can rely only on algorithmic intelligence to cope with it since there is *no way to explicitly query for precise*

data without committing to a partial solution. Clearly, more information or even knowing the exact data would allow for significantly better solutions. The possibility of querying is a reasonable modeling assumption as many real-world applications provide the possibility to obtain exact data at a certain exploration cost such as extra time, money, bandwidth, etc.

The framework of **explorable uncertainty** captures problem settings in which, for any uncertain input element, a *query* can be used to obtain the exact value of that element. Queries are costly, and hence the goal is to make as few queries as possible until sufficient information has been obtained to solve the given problem. The major challenge is to balance the resulting exploration–exploitation trade-off.

Outline of the Chapter
The goal of this chapter is to highlight some recent results on explorable uncertainty from the mathematical optimization perspective and to outline the potential power of such framework and techniques for solving logistics problems. In the following sections, we introduce three models within this framework.

1. An *online model*, in which an uncertain input element can attain any value within a given interval.
2. A *learning-based model*, in which an algorithm has access to a prediction, possibly machine-learned, on the exact value of an uncertain element but without any guarantee on its accuracy.
3. A *stochastic model*, in which there is given probabilistic information about the exact value.

We give intuitive insights, state mathematical results, and outline methods to tackle problems under explorable uncertainty. To illustrate these insights, we consider the example problems of selecting a minimum element within a set of uncertain values by querying a minimal number of elements (*minimum problem*) as well as a query-based variant of the *minimum spanning tree (MST) problem*, a most fundamental network design problem that asks for the minimum cost network (a cycle-free connected graph). These are important problems that appear as subproblems in many applications, also in logistics, and they have been studied extensively in the framework of explorable uncertainty, offering positive results in all three models.

2 Power of Exploring Uncertain Data in Logistics

We showcase different logistics scenarios in which methods building on explorable uncertainty promise a substantial improvement of the current state of the art.

Consider the task of designing an effective complex supply chain (or certain parts of it) for a company with various production facilities, sub-contractors, storage facilities, and customers. At several places in the decision-making process, the management has to decide for a best choice out of a number of choices where some

parameters might be uncertain. Such choices could be possible sub-contractors, facility locations, transportation means, etc. Which is the "best" may depend on parameters such as cost, quality, reliability, or customer satisfaction and might not be known, but there is the possibility to query those parameters by, e.g., measurements, lab tests, customer interviews, etc. This is typically costly and shall be used only if the effect on the performance of supply chain processes is worth it. Powerful and well-thought exploration methods may give a substantial improvement over decision-making under uncertainty.

The next example falls into the area of transport logistics of *perishable goods*, that is, goods with a decay time in the same order of magnitude as the transport time, e.g., fruits, vegetables, meat, or cheese in sea and land transportation. A major challenge in the transportation of perishable goods is the uncertainty in the state of the goods and their environment, e.g., in a container. The ripeness degree and shelf life of fresh fruit are hard to predict as they may change drastically and very fast. During a typical transport, e.g., maritime container transport or transportation via truck or train, there is no feedback on the current state of the goods and no timely action to avoid food loss can be taken. With an *intelligent container* (a keyword coined by Lang and Jedermann (2016)), i.e., a container equipped with sensors and infrastructure for monitoring, communication, and possibly even further actions, such an early feedback and even active queries for more precise information are possible and could be integrated into an adaptive logistics planning framework.

More concretely, consider a truck routing problem for delivering fresh food to several possibly widespread destinations, e.g., raspberries from Turkey delivered by truck to several destinations in Germany. Clearly, frequent sensor measurement and resulting updates on the remaining shelf time cost energy and decrease the life time of a battery. Replacing and disposing batteries costs time, money, and it requires the expertise (of the truck driver) to do so. This may not be relevant for a single 3-day trip, but it might play a significant role as trucks drive tours repeatedly. Dynamic routing algorithms that incorporate data exploration may minimize the food loss while not exceeding a given budget of exploration cost measured, e.g., in energy.

Another serious problem in container transport is pest insects and dangerous gases, which are detected only when unloading at a port where ad hoc protective measures are costly and inefficient. Consider the scheduling and resource allocation for handling containers that require special treatment. The operations for opening dangerous containers in a secure area (chemicals and gas) or rerouting a container (pest insects) are time- and resource-expensive. Suppose sensor measurements can be inquired, and the relevant information can be made available on time for efficiently planning the necessary port operations. It might be more efficient to open several dangerous containers in a secure area at the same time than blocking resources any time a single such container arrives. This is only possible if the information is queried at the right point in time while avoiding unnecessary query cost, e.g., energy.

Besides the technical possibility to query uncertain data, it is a major challenge to algorithmically balance the cost for data exploration (e.g., extensive tests in a lab,

energy consumption for queries to the intelligent container) with the benefit for the quality of a solution (e.g., cost for establishing a network, amount of food loss).

3 Optimization Under Explorable Uncertainty

In explorable uncertainty, we are given, instead of precise data points, only rough information in form of *uncertainty intervals*. Precise data points can be revealed by using *queries*. Since querying data points comes at a cost, the goal is to extract sufficient information to solve the problem while minimizing the total query cost. In the following, we formally introduce the model of explorable uncertainty and highlight important concepts for solving problems under explorable uncertainty.

3.1 The Model

We are given a ground set $\mathcal{I}$ of uncertainty intervals. Each interval $I_i \in \mathcal{I}$ is associated with a precise value $w_i \in I_i$ that is initially unknown. The precise value of an uncertainty interval I_i can be extracted by using a query. Intuitively, querying the interval $I_i = (L_i, U_i)$ replaces the open interval (L_i, U_i) with the singleton $[w_i]$. We call L_i and U_i the lower and upper limits of I_i. How to obtain the upper and lower limits of the uncertainty intervals is problem specific and depends on the application. As an example consider the distances between mobile agents. While the agents change their positions and the precise distance between two agents might not always be known, last known locations as well as maximum movement speeds can be used to compute an uncertainty interval that is guaranteed to contain the precise distance. In the following, we abstract from the process of obtaining the uncertainty intervals and assume they are part of the input. If $I_i = [w_i]$, we define $L_i = U_i = w_i$. A query to an interval I_i comes with a *query cost* of c_i. For the remainder of this chapter, we only consider uniform query costs, i.e., $c_i = 1$ for all $I_i \in \mathcal{I}$.

We can define various optimization problems based on the ground set of uncertainty intervals. For each problem, the goal is to extract sufficient information to solve the problem for a fixed but initially unknown realization of precise values, while minimizing the total query costs. In the case of uniform query costs, the total cost is just the number of queried intervals. A *query set* $Q \subseteq \mathcal{I}$ is *feasible* if querying Q extracts sufficient information to optimally solve the problem at hand. Thus, a query set Q is only feasible if querying Q allows us to compute a solution for the underlying optimization problem that is guaranteed to be optimal for all possible precise values of intervals in $\mathcal{I} \setminus Q$. We further discuss this assumption at the end of the chapter. We analyze the performance of algorithms in terms of their *competitive ratio*. Let $\mathcal{J}$ denote the set of all instances for a problem under explorable uncertainty, let $\text{ALG}(J)$ for $J \in \mathcal{J}$ denote the query cost needed by an

algorithm ALG to solve instance J, and let OPT(J) denote the optimal query cost for solving J. That is, for a fixed instance J with fixed precise values, OPT(J) denotes the minimum query cost necessary to solve J. Then, the competitive ratio of ALG is defined as

$$\max_{J \in \mathcal{J}} \frac{\text{ALG}(J)}{\text{OPT}(J)}.$$

In the following, we introduce two example problems under explorable uncertainty.

3.1.1 Example: Minimum and Selection Problems

In the *minimum problem*, the goal is to determine, for a given set of uncertainty intervals $\mathcal{I}$, an interval $I_i \in \mathcal{I}$ with minimum precise value, i.e., $I_i = \arg\min_{I_i \in \mathcal{I}} w_i$. Note that this problem does not necessarily involve computing the actual precise value of that interval.

As an example recall the scenario given in Sect. 2, where a company has to select the "best" out of a pool of possible sub-contractors, facility locations, transportation means, etc. without having all the information to determine it. This scenario can be modeled as a minimum problem: the possible choices can be modeled by the index set $\{1, \ldots, n\}$. For each possible choice $i \in \{1, \ldots, n\}$, we have an initial estimation of its quality (based, e.g., on publicly available information, past experiences, and already known basic conditions) that can be modeled by the uncertainty interval I_i. A precise estimation for a possible choice can be obtained, e.g., by executing measurements, lab tests, or customer interviews. Then, the process of obtaining a precise estimation for a possible choice can be modeled by a query. As the described operations come typically at a high cost, the goal is to make the best possible choice while minimizing this extra cost. This corresponds to the minimum problem.

Since the precise values are initially unknown, it might not be possible to find the interval of minimum precise value without executing queries. For example, in Fig. 1, we are given a set of two uncertainty intervals with the task to determine the interval with minimum precise value. Since those intervals overlap, both of them could possibly be of minimum precise value. To solve the problem, an algorithm has to execute at least one query.

The example of Fig. 1 also shows that no algorithm is better than 2-competitive for the minimum problem, as Kahan (1991) observed already in his seminal paper. By definition, for an algorithm to be better than 2-competitive, the ratio between ALG(J) and OPT(J) has to be strictly smaller than 2 for *every* instance J. In the example, we consider two instances with the same intervals that differ only

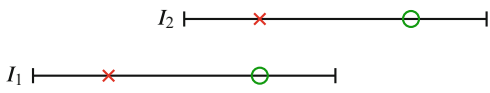

Fig. 1 Lower bound example for the minimum problem in a single set

in the precise values (crosses vs. circles). Since an algorithm has no knowledge of the precise values, both instances look the same to the algorithm, and thus, a deterministic algorithm will make the same first query for both instances. We argue that each possible first query will lead to a ratio of 2 for at least one of the instances, which implies that no deterministic algorithm is better than 2-competitive. In such a worst-case analysis, we may assume that different precise values are revealed for different algorithms. (In general, the precise values are independent of the query order.) If an algorithm queries I_1 first, then, in the worst case, the green circle is revealed as the precise value of I_1. After querying I_1, it is still unknown which interval has minimum value, which forces the algorithm to also query I_2. If the query to I_2 again reveals the green circle as the precise value of I_2, an optimal query set could determine that I_1 has minimum precise value by only querying I_2. Thus, the cost of the algorithm is twice the cost of the optimal query set. Vice versa, if an algorithm queries I_2 first, then, in the worst case, the red crosses are revealed as precise values, and the algorithm queries $\{I_1, I_2\}$, while the optimal query set queries only I_1. Hence, for any algorithm's choice on this instance (either query I_1 first or I_2), there is a realization of precise values on which the algorithm requires two queries, whereas an optimal query set with one query exists. This implies that no deterministic algorithm (an algorithm that makes the same decisions when given the same input) can be better than 2-competitive.

In a more general variant of the minimum problem, we are given a family $\mathcal{S}$ of (possibly non-disjoint) subsets of $\mathcal{I}$, and the goal is to determine the member of minimum precise value for each subset $S_j \in \mathcal{S}$. Consider the example given in Bampis et al. (2021) concerning a multi-national medical company. The company relies on certain products for its operation in each country, e.g., a chemical ingredient or a medicine. However, due to different approval mechanisms, the concrete products that are accessible differ for each country. The task is to find the best approved product for each country. The product quality can be determined by extensive tests in a lab (queries) and, since the quality is independent of the country, each product has to be tested at most once. The set of products available in one country corresponds to a set in $\mathcal{S}$, and the problem of identifying the best product in each country is the minimum problem in multiple sets.

In a similar way, we can model other selection problems, e.g., finding the kth smallest element and sorting.

3.1.2 Example: Minimum Spanning Tree Problem

In the *minimum spanning tree (MST) problem*, we are given a weighted, undirected, connected graph $G = (V, E)$, with nodes V and edges E, where each edge $e \in E$ has associated a weight $w_e \geq 0$. The task is to find a spanning tree of minimum total weight. A spanning tree is a connected acyclic graph whose edges span all the vertices. See Fig. 2 for an example graph. The MST problem has various applications, e.g., in the design of distribution networks: nodes can be used to model storage facilities, manufacturers, and transportation systems, while the edges and

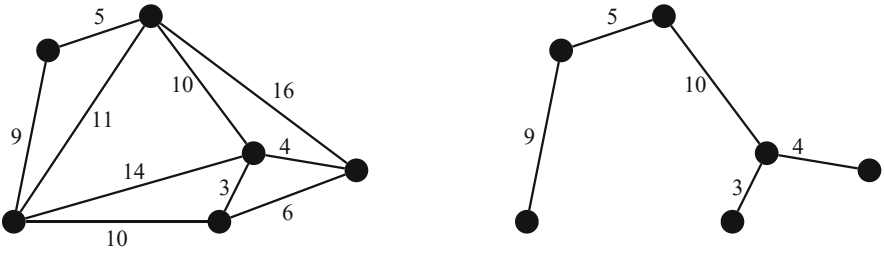

Fig. 2 Graph *without* uncertainty in the edge weights (left) and the corresponding minimum spanning tree (right)

their weights can model the cost of establishing a direct connection between two such points of interest, where a direct connection could, for example, be a road, a pipeline, or an Ethernet connection. To establish connections between all points of interest in a cost-minimal way, we have to compute a minimum spanning tree.

In the MST problem with uncertainty, the precise edge weights w_e are unknown. Each edge $e \in E$ is associated with an uncertainty interval $I_e \in \mathcal{I}$, and w_e is guaranteed to be in the given interval I_e. The task is to find an MST in the uncertainty graph G for an a priori unknown realization of edge weights. Note that this problem does not necessarily involve computing the actual MST weight. In the application given above, uncertainty could arise from an unknown existing infrastructure or unclear environmental and political factors. For example, the exact existing underground infrastructure might be unknown and potentially decrease the cost of building a connection, and the building of a pipeline could lead to conflicts with environmental protection groups or nearby residents that might increase the cost. These dynamic changes in the cost can be modeled by uncertainty intervals. Such uncertainties can then be resolved by inspecting the existing infrastructure or surveying residents and other potential stakeholder; both actions can be modeled by queries. Since the described actions to resolve the uncertainty can be cost extensive, the goal is to find an MST while minimizing the query cost.

It is well known that edges that have unique minimum weight in a cut of the graph are part of any MST. Furthermore, edges that have unique maximum weight on a cycle are part of no MST. Thus, to solve the MST problem under explorable uncertainty, we have to analyze the behavior of intervals and queries in terms of their interplay on cycles and in cuts. A simple cycle with three edges (triangle) already gives both lower bound examples and insights about the structure of a feasible query set. Consider the example of Fig. 3. It is clear that edge h is part of every MST, but we cannot decide which of the two edges f and g is in the MST without querying at least one of them. Similar to the lower bound example for the minimum problem, querying f first, in the worst case, reveals the green circles as precise weights, while querying g first reveals the red crosses. This forces any deterministic algorithm to query two elements, while the optimal query set contains just one. Thus, as was

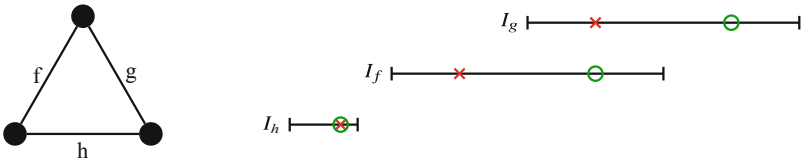

Fig. 3 Lower bound example for the minimum spanning tree problem

observed in Erlebach et al. (2008), no such algorithm can achieve a competitive ratio smaller than 2.

3.2 Mandatory Queries

A key aspect of several algorithms for problems under explorable uncertainty is the identification of *mandatory queries*. An interval $I_i \in \mathcal{I}$ is mandatory for the problem instance if each feasible query set has to query I_i, i.e., $I_i \in Q$ for all feasible query sets Q. The identification of mandatory queries is important since an algorithm can query such intervals without ever worsening its competitive ratio. In a sense, mandatory queries allow an algorithm to extract new information "for free." The revealed precise values might in turn allow the identification of further mandatory queries and, thus, lead to chains of mandatory queries. While it is possible to achieve theoretical worst-case guarantees without exploiting mandatory elements, empirical results indicate that the performance of algorithms significantly improves when the algorithm prioritizes the identification and querying of mandatory intervals (Erlebach et al. 2020; Focke et al. 2020).

When characterizing mandatory queries, we distinguish between characterizations based on the unknown precise values and characterizations that are only based on the uncertainty intervals. While the latter only uses information that can be accessed by an algorithm and, therefore, can actually be used to identify mandatory queries, the former is still helpful to analyze algorithms and will be useful in the following sections. We continue by characterizing mandatory queries for the two example problems.

3.2.1 Identifying Mandatory Queries for the Minimum Problem

Consider the minimum problem in multiple sets as introduced in the previous section. For a set $S \in \mathcal{S}$, we call an interval $I_i \in S$ the *precise minimum* of S if I_i has minimum precise value over all elements of S. The following lemma allows us to identify mandatory queries based on the precise values of the intervals.

Lemma 1 (Erlebach et al. (2020)) *An interval I_i is mandatory for the minimum problem if and only if (a) I_i is a precise minimum of a set S and contains w_j of another interval $I_j \in S \setminus \{I_i\}$ (in particular, if $I_j \subseteq I_i$), or (b) I_i is not a precise minimum of a set S with $I_i \in S$ but contains the value of the precise minimum of S.*

A common proof technique to show that an interval I_i is mandatory is to consider the query set $\mathcal{I} \setminus \{I_i\}$. Showing that querying every element except I_i does not solve the problem implies that I_i is mandatory. Vice versa, if querying $\mathcal{I} \setminus \{I_i\}$ solves the problem, then I_i is not mandatory. The following proof, which was given in Erlebach et al. (2020), uses this proof technique to show Lemma 1.

Proof If I_i is the precise minimum of S and contains w_j of another interval $I_j \in S$, then S cannot be solved even if we query all intervals in $S \setminus \{I_i\}$, as we cannot prove $w_i \leq w_j$ or $w_j \leq w_i$. If I_i is not a precise minimum of set S with $I_i \in S$ and contains the precise minimum value w^*, then S cannot be solved even if we query all intervals in $S \setminus \{I_i\}$, as we cannot prove that $w^* \leq w_i$.

If I_i is the precise minimum of a set S, but $w_j \notin I_i$ for every $I_j \in S \setminus \{I_i\}$, then $S \setminus \{I_i\}$ is a feasible query set for S. If I_i is not a precise minimum of a set S and does not contain the precise minimum value of S, then again $S \setminus \{I_i\}$ is a feasible query set for S. If every set S that contains I_i falls into one of these two cases, then querying all intervals except I_i is a feasible query set for the whole instance. □

Explicitly, Lemma 1 only enables us to identify mandatory intervals given full knowledge of the precise values, but it also implies criteria to identify *known mandatory* intervals, i.e., intervals that are known to be mandatory given only the intervals, and precise values revealed by previous queries. We call an interval *leftmost* in a set S if it is an interval with minimum lower limit in S. The following corollary follows from Lemma 1 and gives a characterization of known mandatory intervals.

Corollary 1 (Erlebach et al. (2020)) *If the leftmost interval I_l in a set S contains the precise value of another interval in S, then I_l is mandatory. In particular, if I_l is leftmost in S and $I_j \subseteq I_l$ for some $I_j \in S \setminus \{I_l\}$, then I_l is mandatory.*

3.2.2 Identifying Mandatory Queries for the Minimum Spanning Tree Problem

Mandatory queries for the MST problem can be characterized by using a structural property given by Megow et al. (2017). Let the *lower limit tree* $T_L \subseteq E$ be an MST for values w^L with $w_e^L = L_e + \epsilon$ for an infinitesimally small $\epsilon > 0$. Similarly, let the *upper limit tree* T_U be an MST for values w^U with $w_e^U = U_e - \epsilon$. Using the lower and upper limit trees, the following lemma allows us to identify mandatory queries based only on the intervals.

Lemma 2 (Megow et al. (2017)) *Any edge in $T_L \setminus T_U$ is mandatory.*

Thus, we may repeatedly query edges in $T_L \setminus T_U$ until $T_L = T_U$, and this will not worsen the competitive ratio. By this preprocessing, we may assume $T_L = T_U$. A characterization of the mandatory queries based on the full knowledge of the precise values is given by Erlebach and Hoffmann (2014).

3.3 Methods and Results

While the identification and querying of mandatory elements improve the performance of algorithms empirically and will be a key ingredient in the following sections, it is not sufficient to solve our two example problems. Therefore, we consider the *witness set algorithm*, one of the most important frameworks in explorable uncertainty. The witness set algorithm was introduced by Bruce et al. (2005) and relies on the identification of *witness sets*. A set $W \subseteq \mathcal{I}$ is a witness set if each feasible query set has to query at least one member of W, i.e., if $W \cap Q \neq \emptyset$ for all feasible query sets Q. Note that witness sets W with $|W| = 1$ are exactly the mandatory queries. Algorithm 1 formulates the witness set algorithm in a problem independent way. The algorithm essentially just queries witness sets until the problem is solved. Similar to mandatory queries, we distinguish between witness sets that can be identified based on the uncertainty intervals alone and witness sets that can only be identified based on knowledge of the precise values. The algorithm can only use the former kind.

Algorithm 1: Abstract formulation of the witness set algorithm

Input: Problem under explorable uncertainty with uncertainty intervals $\mathcal{I}$
while *The problem is not solved yet* **do**
  Query all elements of a witness set W.

The competitive ratio of the witness set algorithm depends on the size of the queried witness sets as formulated in the following lemma.

Lemma 3 (Bruce et al. (2005)) *If $|W| \leq \rho$ holds for all witness sets W that are queried by the witness set algorithm, then the algorithm is ρ-competitive.*

Proof Since querying elements multiple times does not reveal additional information, we can assume that all queried witness sets are pairwise disjoint. Let $W_1, \ldots, W_k$ denote those witness sets. Then, by definition of witness sets and since the sets are pairwise disjoints, the optimal query set contains at least k elements. By assumption, $|W_j| \leq \rho$ holds for all $j \in \{1, \ldots, k\}$. Thus, the algorithm queries at most $\rho \cdot k$ elements, and the competitive ratio is at most $\frac{\rho \cdot k}{k} = \rho$. □

In order to apply (and analyze) the witness set algorithm to a concrete problem, one has to characterize witness sets, bound the size of the witness sets, and show

that the problem is solved once the characterization does not admit any more witness sets. In the following, we apply the algorithm to the two example problems.

3.3.1 Witness Set Algorithm for the Minimum Problem

For the minimum problem, we can identify witness sets of size one, i.e., mandatory queries, by using Corollary 1. Furthermore, we can identify witness sets of size two using the following lemma that was first (implicitly) shown by Kahan (1991).

Lemma 4 (Kahan (1991)) *A set $\{I_i, I_j\} \subseteq \mathcal{I}$ is a witness set if there exists an $S \in \mathcal{S}$ with $\{I_i, I_j\} \subseteq S$, $I_i \cap I_j \neq \emptyset$, and either I_i or I_j leftmost in S.*

Similar to the proof of the mandatory characterization, the lemma can be shown by considering the query set $Q = \mathcal{I} \setminus \{I_i, I_j\}$. After querying Q, both I_i and I_j still could be of minimum precise value in S. Thus, the problem is not solved yet, and at least one of I_i and I_j needs to be queried. This is a common proof strategy for showing that a subset of $\mathcal{I}$ is a witness set.

The witness set algorithm for the minimum problem repeatedly identifies and queries witness sets of size at most two by applying Corollary 1 and Lemma 4 until they cannot be applied anymore. If Lemma 4 cannot be applied anymore, then the leftmost interval I_i of each set S is not overlapped by any $I_j \in S \setminus \{I_i\}$. This implies that the leftmost intervals are the precise minima of the sets. Consequently, the problem then is solved, which implies the following theorem. The theorem was first (implicitly) shown by Kahan (1991) for a single set and translates to multiple sets.

Theorem 1 (Kahan (1991)) *The witness set algorithm is 2-competitive for the minimum problem. This competitive ratio is best possible for deterministic algorithms.*

3.3.2 Witness Set Algorithm for the Minimum Spanning Tree Problem

For the minimum spanning tree problem, we can identify witness sets of size one by using Lemma 2. Furthermore, we can identify witness sets of size two by using the following lemma that was shown in Erlebach et al. (2008), Megow et al. (2017). Recall that T_L and T_U are the lower and upper limit trees of the instance. Let $f_1, \ldots, f_l$ denote the edges in $E \setminus T_L$ ordered by non-decreasing lower limit, and let C_i be the unique cycle in $T_L \cup \{f_i\}$.

Lemma 5 (Erlebach et al. (2008)) *Let $i \in \{1, \ldots, l\}$ be the smallest index such that $I_{f_i} \cap I_e \neq \emptyset$ holds for some $e \in C_i \setminus \{f_i\}$. Then, $\{f_i, e\}$ is a witness set.*

The witness set algorithm for the MST problem repeatedly identifies and queries witness sets of size at most two by applying Lemmas 2 and 5 until they cannot be applied anymore. If Lemma 5 cannot be applied anymore, then each f_i does not overlap with any $e \in C_i \setminus \{f_i\}$. This implies that each f_i is maximal in C_i and

therefore not part of any MST. Thus, T_L is known to be an MST and the problem is solved. This implies the following theorem.

Theorem 2 (Erlebach et al. (2008)) *The witness set algorithm is* 2*-competitive for the MST problem. This competitive ratio is best possible for deterministic algorithms.*

4 Explorable Uncertainty Beyond Worst-Case Analysis

In the previous section, we saw that, for both example problems, there is a 2-competitive algorithm with a matching lower bound. A natural question asks for ways to circumvent those lower bounds. There are different strategies for adjusting the model and performance guarantees beyond worst-case analysis. One common strategy is to allow an algorithm to make decisions randomly and measure the worst-case performance in expectation. The randomized algorithm given by Megow et al. (2017) shows that randomization is indeed powerful and admits improved results for the MST problem.

However, in this section, we follow a different approach and assume access to additional information on the problem instance. We present a *learning-augmented* and a *stochastic* variant of explorable uncertainty, where we are given predictions without any guarantee and probabilistic information, respectively. We outline how to design algorithms that can exploit this extra information by obtaining provable performance guarantees.

4.1 Exploiting Untrusted Predictions

In this section, we review the learning-augmented methods for explorable uncertainty that were introduced very recently by Erlebach et al. (2020). In the learning-augmented setting, we assume that we are given additional information on the problem instance in the form of *predictions* for the precise values of the uncertainty intervals. Those predictions could, for example, be derived by using machine learning (ML) methods. Based on the tremendous progress in artificial intelligence and machine learning, assuming access to such predictions of good accuracy seems reasonable. However, there is no guarantee on the accuracy and the predictions might be arbitrarily wrong. The learnability of predictions for the example problems is discussed in Erlebach et al. (2020).

Formally, we assume that we are given a predicted value $\overline{w}_i$ for each $I_i \in \mathcal{I}$. The predicted values are available *before* we query any elements and predict the results of the queries. Since we do not have any accuracy guarantee on the predictions, we call them *untrusted* to emphasize that the difference between w_i and $\overline{w}_i$ might be arbitrarily large. To compensate for the missing guarantee on the prediction quality,

we aim at designing algorithms that achieve an improved performance for accurate predictions while being robust against arbitrarily bad predictions.

We refine competitive analysis to formulate those two objectives by adding a prediction awareness and adopt the notions of α-consistency and β-robustness (Lykouris and Vassilvitskii 2018; Purohit et al. 2018). An algorithm is α-consistent if it is α-competitive when the predictions are correct, i.e., $w_i = \overline{w}_i$ for all $I_i \in \mathcal{I}$, and it is β-robust if it is β-competitive for arbitrarily wrong predictions. While consistency and robustness only formulate the extreme cases for the prediction quality, we are also interested in guarantees with a smooth transition between consistency and robustness. We aim for performance guarantees that linearly degrade for an increased prediction error. This motivates interesting questions regarding suitable ways of measuring these errors.

4.1.1 Error Measures and Learnability

In the following, we consider the error measures introduced by Erlebach et al. (2020). A first simple and natural prediction error is the number of inaccurate predictions $k_\# = |\{I_i \in \mathcal{I} \mid w_i \neq \overline{w}_i\}|$. However, for the two example problems, a performance guarantee that, in terms of consistency, improves upon the lower bound of two and linearly degrades depending on $k_\#$ is not possible (Erlebach et al. 2020). The reason is that $k_\#$ completely ignores the structure of the intervals. (Similarly, using an error metric that depends on the distances between w_i and $\overline{w}_i$, e.g., $\sum_{I_i \in \mathcal{I}} |w_i - \overline{w}_i|$, would not be meaningful because only the order of the values and the interval end points matters for our problems.) To address this weakness, Erlebach et al. (2020) introduced two refined measures for the prediction quality.

Hop distance. Intuitively, for each interval $I_i \in \mathcal{I}$ with $w_i \neq \overline{w}_i$, this error measure counts the number h_i of lower and upper limits in $\mathcal{I} \setminus \{I_i\}$ that lie in the intervals $[w_i, \overline{w}_i)$ or $(w_i, \overline{w}_i]$. The hop distance of a given instance is then $k_h = \sum_{i=1}^{n} h_i$; see also the left part of Fig. 4. Note that the hop distance value h_i for a single interval I_i only depends on the *number* of interval borders that lie between w_i and $\overline{w}_i$ and, apart from that, is independent of the distance between w_i and $\overline{w}_i$. While the hop distance captures how the relations of the precise values to other intervals change compared to the predicted values, not every such change affects a feasible query set.

Mandatory query distance. To compensate for this fact, Erlebach et al. (2020) introduce another error measure based on the set $\mathcal{I}_R$ of mandatory intervals and the set $\mathcal{I}_P$ of prediction mandatory intervals, i.e., intervals that are mandatory under the assumption that all predictions are correct. The *mandatory query distance* is the size of the symmetric difference of $\mathcal{I}_P$ and $\mathcal{I}_R$, i.e., $k_M = |\mathcal{I}_P \Delta \mathcal{I}_R| = |(\mathcal{I}_P \cup \mathcal{I}_R) \setminus (\mathcal{I}_P \cap \mathcal{I}_R)| = |(\mathcal{I}_P \setminus \mathcal{I}_R) \cup (\mathcal{I}_R \setminus \mathcal{I}_P)|$. The right part of Fig. 4 shows an example for the mandatory query distance with $k_M = 1$. With respect to the precise values, both $\{I_1\}$ and $\{I_2, I_3, I_4\}$ are feasible query sets, and therefore, no interval is part of every feasible query set. This implies $\mathcal{I}_R = \emptyset$. Under the assumption that the

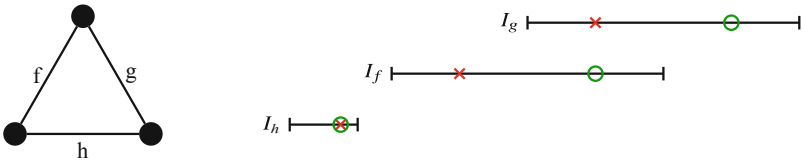

Fig. 4 Examples for the minimum problem with a single set $S = \{I_1, I_2, I_3, I_4\}$ as given in Erlebach et al. (2020). Circles and crosses illustrate precise values and predictions, respectively. Left: predictions and true values with a total hop distance of $k_h = 5$. Right: instance with a mandatory query distance of $k_M = 1$

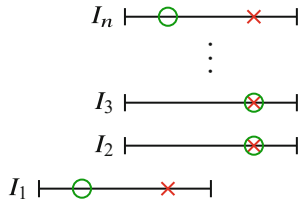

Fig. 5 Example for the minimum problem in a single set as given in Erlebach et al. (2020). The crosses and circles denote the predicted and precise values, respectively

predicted values are correct, Lemma 1 implies that I_1 is part of every feasible query set, and therefore, $\mathcal{I}_P = \{I_1\}$. It follows $k_M = |\mathcal{I}_P \Delta \mathcal{I}_R| = 1$.

In Erlebach et al. (2020), it is shown that, for both example problems, it is indeed possible to learn predictions with respect to k_h and k_M.

4.1.2 Methods and Results

A key aspect in the design of learning-augmented algorithms for explorable uncertainty is the identification of *prediction mandatory* intervals, i.e., intervals that are mandatory under the assumption that the predicted values are correct. For the two example problems, we can apply the characterizations of mandatory queries (cf. Lemma 1 and Erlebach and Hoffmann (2014)) that are based on the precise values under the assumption that the predicted values match the precise values. This allows us to identify all prediction mandatory queries. If we completely trust the predictions, we should just query all prediction mandatory elements. However, the example in Fig. 5 shows that completely trusting the predictions can lead to an arbitrarily bad robustness. In the example, the intervals $\{I_2, \ldots, I_n\}$ are prediction mandatory, but, if the predicted value of I_1 is wrong, it may be the case that the optimal query set contains only I_1.

This example implies that algorithms should balance the querying of prediction mandatory intervals with additional queries. In fact, Erlebach et al. (2020) show the following lower bound on the optimal trade-off between consistency and robustness for the two example problems.

Theorem 3 (Erlebach et al. (2020)) *Let $\beta \geq 2$ be a fixed integer. For the minimum and MST problems under explorable uncertainty, there is no deterministic β-robust algorithm that is α-consistent for $\alpha < 1 + \frac{1}{\beta}$. And vice versa, no deterministic α-consistent algorithm, with $\alpha > 1$, is β-robust for $\beta < \max\{\frac{1}{\alpha-1}, 2\}$.*

As a consequence of Theorem 3, no 2-robust algorithm can be better than 1.5-consistent. Thus, if we want to match the lower bound of 2 in terms of robustness, we should aim for 1.5-consistency. To achieve a good trade-off between consistency and robustness for the two example problems, we can use the two-phase framework as presented in Erlebach et al. (2020). Here, we describe on a high level the framework that can be used to achieve 1.5-consistency and 2-robustness for the two problems. The framework starts by querying all prediction mandatory elements. In a second stage, when no elements are prediction mandatory anymore, the algorithm has to decide which non-prediction mandatory elements to query. For the example problems, the algorithm has to decide which element of each witness pair it queries.

During the first phase, it is not sufficient to just query prediction mandatory elements since we already saw that this can lead to an arbitrarily bad robustness. Thus, each query to a prediction mandatory element is complemented with queries to further elements in such a way that they form witness sets. As the framework aims at breaking the lower bound of 2 in terms of consistency, it is not enough to form and query a size-2 witness set. To guarantee a consistency of 1.5, the framework instead identifies and queries sets of three elements for which we can guarantee that at least 2 of them must be contained in any feasible query set. Finding such elements based on the interval structure alone is not always possible as this would imply a 1.5-competitive algorithm for the non-learning augmented setting, which would contradict the lower bounds for the two example problems. Therefore, the framework identifies such sets under the assumption that the predictions are correct. After identifying such a set, its elements have to be queried adaptively, and, in case of wrong predictions, the algorithm has to compensate for the error by not querying all three elements. Querying all three elements in parallel might, in the case of wrong predictions, lead to a violation of the 2-robustness. The first framework phase repeatedly identifies such elements and queries them in a careful order while adjusting for potential errors, until no prediction mandatory elements remain. The characterization and identification of sets that satisfy the mentioned guarantee is problem specific and a challenging key aspect when applying this strategy to the concrete problems.

In the second phase of the framework, there are no more prediction mandatory elements, and the algorithm cannot identify any more "safe" queries. For the two example problems, this means that the algorithm has to decide, for each witness pair as characterized by Lemmas 4 and 5, which of the two elements to query. These decisions come down to finding a minimum vertex cover in an auxiliary graph representing the structure of the witness sets. In particular, the second phase for the minimum problem consists of finding and querying a minimum vertex cover. If the predictions are correct, querying the vertex cover solves the remaining problem. Otherwise, additional queries might be necessary, but those queries can

be shown to be mandatory. Since both the size of the minimum vertex cover and the number of queried mandatory elements are lower bounds on the optimal query cost, the second framework phase is even 1-consistent and 2-robust. In the MST problem, wrong predictions can change the witness sets dynamically. Therefore, the second framework phase must be executed in a more adaptive and very careful way, requiring substantial additional work.

By applying (a generalized version of) the described framework, the following results for the two example problems can be achieved. In both theorems, OPT denotes the cost of an optimal query set, and the parameter γ can be used to configure the degree to which we trust the predictions. For increasing γ, the consistency improves but the robustness gets worse. To model smooth transitions between consistency and robustness, the theorems state the performance guarantees as the minimum of the error-dependent consistency and the robustness.

Theorem 4 (Erlebach et al. (2020)) *There is an algorithm for the minimum problem under uncertainty that, given an integer parameter* $\gamma \geq 2$*, achieves a competitive ratio of* $\min\{(1+\frac{1}{\gamma})(1+\frac{k_h}{OPT}), \gamma\}$*. If* $\gamma = 2$*, the algorithm is* 1.5*-consistent and* 2*-robust. Furthermore, there is an algorithm for the minimum problem under uncertainty that, given an integer parameter* $\gamma \geq 2$*, achieves a competitive ratio of* $\min\{(1+\frac{1}{\gamma-1}) \cdot (1+\frac{k_M}{OPT}), \gamma\}$.

Theorem 5 (Erlebach et al. (2020)) *There is a* 1.5*-consistent and* 2*-robust algorithm for the MST problem under uncertainty. Furthermore, there is an algorithm with competitive ratio* $\min\{1+\frac{1}{\gamma}+\left(5+\frac{1}{\gamma}\right) \cdot \frac{k_h}{OPT}, \max\{3, \gamma+\frac{1}{OPT}\}\}$*, for any* $\gamma \in \mathbb{Z}_{\geq 2}$.

These results show that learning augmentation can be successfully applied to problems under explorable uncertainty and circumvents known lower bounds for good predictions, while at the same time providing strong bounds on the worst-case performance even when the predictions are completely wrong. This eases the integration of machine learning into a system since it allows improved results while protecting users from occasional failures of the ML algorithms.

4.2 Exploiting Stochastic Information

Recently, the setting of *stochastic explorable uncertainty* has received some attention in the context of sorting (Chaplick et al. 2020) and the minimum problem (Bampis et al. 2021), which Bampis et al. (2021) phrases as a (hyper-)graph orientation problem.

In the stochastic setting, we are given a continuous probability distribution d_i over the interval $I_i = (L_i, U_i)$ for each $I_i \in \mathcal{I}$. The precise value w_i of an interval I_i

is drawn independently from d_i. Bampis et al. (2021) again analyze an algorithm ALG in terms of its competitive ratio

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

where $\mathbb{E}[\text{ALG}(J)]$ denotes the expected query cost of ALG when solving instance J, $\mathbb{E}[\text{OPT}(J)]$ denotes the expected optimal query cost for J, and $\mathcal{J}$ is the set of all instances. As a main result, Bampis et al. (2021) give the following theorem.

Theorem 6 (Bampis et al. (2021)) *For any $\epsilon > 0$, there exists a $f(\alpha)$-competitive algorithm for the minimum problem in multiple sets, where $f(\alpha) \in [1.618 + \epsilon, 2]$ depends on the approximation ratio α for solving a vertex cover problem in an auxiliary graph.*

The algorithm relies on computing the probability that an interval is mandatory using the characterization of Lemma 1. It queries all vertices that have a mandatory probability exceeding a certain threshold. Afterward, the algorithm solves a vertex cover problem on an auxiliary graph by first preprocessing the instance via a classical linear programming relaxation and, afterward, executing the α-approximation. The algorithm queries the computed vertex cover and, thereafter, only mandatory intervals that remain. In addition to this general algorithm, Bampis et al. (2021) give several lower bounds and improved algorithms for special cases that also prioritize queries to intervals with a high probability to be mandatory.

5 Concluding Remarks

This chapter discusses the model of explorable uncertainty and its potential use for decision-making under uncertainty in logistics. We illustrate classical techniques to design algorithms with worst-case guarantees using the two example problems of finding the minima of multiple sets and determining a minimum spanning tree. With the learning-augmented and stochastic setting, we also present models and techniques for algorithm design with guarantees beyond the worst case and show that known limitations of worst-case analysis can be circumvented by such settings.

In this chapter, we require algorithms to determine an optimal solution for the underlying optimization problem. One could relax this restriction and ask for an α-approximation. Unfortunately, for the example problems, the lower bounds translate to this relaxed setting (Megow et al. 2017, Section 10). Another interesting variation of the model refers to the objective function. While we consider settings in which the query cost is significant, optimization with explorable uncertainty seems relevant also in settings where query cost and objective value of the underlying problem are comparable. In this case, it would be interesting to consider a combined objective, e.g., to minimize the (weighted) sum of both values.

We illustrated techniques using two example problems, a selection and a network design problem. These appear as sub-problems in classical logistics questions and our techniques would be directly applicable. Admittedly, a rigorous worst-case guarantee such as the competitive ratio may be relevant only in rare applications; otherwise, a good empirical performance on practical input instances is sufficient. In more complex problem settings, e.g., involving additionally routing, packing, and scheduling aspects, we may not be able to prove worst-case guarantees on the performance of our algorithms and they may not even exist. Here, performance measures beyond the worst case are particularly relevant. Overall, we expect that the model and techniques presented here give insights on the power, tractability, and applicability of explorable uncertainty, and we hope that they can serve as a first step toward tackling more complex logistics problems.

6 Bibliographical Notes

We conclude with further pointers to previous work on optimization with explorable uncertainty.

The line of research on explorable uncertainty has been initiated by Kahan (1991) in the context of selection problems. Subsequent work addressed caching problems (Olston and Widom 2000), problems such as computing a function value (Khanna and Tan 2001), finding the kth smallest value in a set of uncertainty intervals (Feder et al. 2003; Gupta et al. 2016), also with non-uniform query cost (Feder et al. 2003), and sorting (Halldórsson and de Lima 2019).

Interestingly, the sorting problem is a special case of the minimum problem for multiple sets. Given an instance of the sorting problem, we can create a set for each pair of elements that are in the same set of the sorting instance and obtain a minimum problem whose feasible query sets also solve the sorting problem. Halldórsson and de Lima (2019) showed directly that the witness set algorithm for sorting a single set is 2-competitive and is best possible. They also show that the competitive ratio can be improved to 1.5 using randomization. Furthermore, Chaplick et al. (2020) introduce an algorithm for sorting a single set of elements under stochastic uncertainty that is optimal in terms of the expected cost $\mathbb{E}[\text{ALG}(J)]$. The competitive ratio of this algorithm is unknown.

Only more recently also optimization problems have been studied. A key role plays the fundamental MST problem with uncertainty. The 2-competitive deterministic witness set algorithm was presented and shown to be best possible by Erlebach et al. (2008). The randomized algorithm by Megow et al. (2017) has an improved competitive ratio of 1.707. 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 (Erlebach et al. 2016; Megow et al. 2017), even for the case with non-uniform query costs (Megow et al. 2017). Other works on the MST problem (and matroids) study a non-adaptive variant (Merino and Soto 2019) and the offline verification problem (Erlebach and

Hoffmann 2014; Megow et al. 2017) and conduct an experimental study (Focke et al. 2020).

Other optimization problems studied in the context of explorable uncertainty include the shortest path problem (Feder et al. 2007), the knapsack problem (Goerigk et al. 2015), and scheduling problems (Albers and Eckl 2020; Arantes et al. 2018; Dürr et al. 2020).

The growth of data-driven applications and machine learning methods in the past years gave rise to a model for learning-augmented online algorithms. The model has been proposed by Medina and Vassilvitskii (2017) in the context of revenue optimization followed by work on online caching by Lykouris and Vassilvitskii (2018). Purohit et al. (2018) studied online scheduling and rent-or-buy problems with respect to consistency and robustness, and they obtained performance guarantees as a function of the prediction error. This work initiated a vast growing line of research, which makes ML predictions without any accuracy guarantee useful in the design of algorithms with hard performance guarantees. Overall, learning-augmented online optimization is a highly topical concept with high potential also for applications in logistics problems.

Acknowledgments The authors were partially funded by the German Science Foundation (DFG).

References

Albers, S., Eckl, A.: Explorable uncertainty in scheduling with non-uniform testing times. In: International Workshop on Approximation and Online Algorithms (2020)

Arantes, L., Bampis, E., Kononov, A.V., Letsios, M., Lucarelli, G., Sens, P.: Scheduling under uncertainty: a query-based approach. In: IJCAI 2018: 27th International Joint Conference on Artificial Intelligence, pp. 4646–4652 (2018)

Bampis, E., Dürr, C., Erlebach, T., de Lima, M.S., Megow, N., Schlöter, J.: Orienting (hyper)graphs under explorable stochastic uncertainty. In: Proceedings of the 29th Annual European Symposium on Algorithms (2021)

Ben-Tal, A., El Ghaoui, L., Nemirovski, A.S.: Robust Optimization. Princeton Series in Applied Mathematics. Princeton University Press, Princeton (2009)

Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming, 2nd edn. Springer, New York (2011)

Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)

Bruce, R., Hoffmann, M., Krizanc, D., Raman, R.: Efficient update strategies for geometric computing with uncertainty. Theory Comput. Syst. **38**(4), 411–423 (2005)

Chaplick, S., Halldórsson, M.M., de Lima, M.S., Tonoyan, T.: Query minimization under stochastic uncertainty. In: Kohayakawa, Y., Miyazawa, F.K. (eds.) LATIN 2020, vol. 12118. Lecture Notes in Computer Science, pp. 181–193 (2020)

Dürr, C., Erlebach, T., Megow, N., Meißner, J.: An adversarial model for scheduling with testing. Algorithmica **82**(12), 3630–3675 (2020)

Erlebach, T., Hoffmann, M.: Minimum spanning tree verification under uncertainty. In: Kratsch, D., Todinca, I. (eds.) International Workshop on Graph-Theoretic Concepts in Computer Science, vol. 8747. Lecture Notes in Computer Science, pp. 164–175 (2014)

Erlebach, T., Hoffmann, M., Krizanc, D., M. Mihalák, Raman, R.: Computing minimum spanning trees with uncertainty. In: Proceedings of the 25th Annual Symposium on the Theoretical Aspects of Computer Science, pp. 277–288 (2008)

Erlebach, T., Hoffmann, M., Kammer, F.: Query-competitive algorithms for cheapest set problems under uncertainty. Theor. Comput. Sci. **613**, 51–64 (2016)

Erlebach, T., Hoffmann, M., de Lima, M.S., Megow, N., Schlöter, J.: Untrusted predictions improve trustable query policies. CoRR, abs/2011.07385 (2020)

Feder, T., Motwani, R., Panigrahy, R., Olston, C., Widom, J.: Computing the median with uncertainty. SIAM J. Comput. **32**(2), 538–547 (2003)

Feder, T., Motwani, R., O'Callaghan, L., Olston, C., Panigrahy, R.: Computing shortest paths with uncertainty. J. Algorithms **62**(1), 1–18 (2007)

Focke, J., Megow, N., Meißner, J.: Minimum spanning tree under explorable uncertainty in theory and experiments. ACM J. Exp. Algorithmics **25**, 1–20 (2020)

Goerigk, M., Gupta, M., Ide, J., Schöbel, A., Sen, S.: The robust knapsack problem with queries. Comput. Operat. Res. **55**, 12–22 (2015)

Gupta, M., Sabharwal, Y., Sen, S.: The update complexity of selection and related problems. Theory Comput. Syst. **59**(1), 112–132 (2016)

Halldórsson, M.M., de Lima, M.S.: Query-competitive sorting with uncertainty. In: 44th International Symposium on Mathematical Foundations of Computer Science, vol. 138. Leibniz International Proceedings in Informatics, pp. 7:1–7:15 (2019)

Kahan, S.: A model for data in motion. In: STOC'91: 23rd Annual ACM Symposium on Theory of Computing, pp. 265–277 (1991)

Khanna, S., Tan, W.-C.: On computing functions with uncertainty. In: PODS '01: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 171–182 (2001)

Lang, W., Jedermann, R.: What can MEMS do for logistics of food? Intelligent container technologies—a review. IEEE Sensors J. **16**(18), 6810–6818 (2016)

Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. In: Proceedings of Machine Learning Research, pp. 3302–3311 (2018)

Medina, A.M., Vassilvitskii, S.: Revenue optimization with approximate bid predictions. In: Proceedings of NIPS, pp. 1856–1864 (2017)

Megow, N., Meißner, J., Skutella, M.: Randomization helps computing a minimum spanning tree under uncertainty. SIAM J. Comput. **46**(4), 1217–1240 (2017)

Merino, A.I., Soto, J.A.: The minimum cost query problem on matroids with uncertainty areas. In: Proceedings of ICALP, vol. 132. Leibniz International Proceedings in Informatics, pp. 83:1–83:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Wadern (2019)

Olston, C., Widom, J.: Offering a precision-performance tradeoff for aggregation queries over replicated data. In: VLDB 2000: 26th International Conference on Very Large Data Bases, pp. 144–155 (2000)

Purohit, M., Svitkina, Z., Kumar, R.: Improving online algorithms via ML predictions. In: Proceedings of Advances in Neural Information Processing Systems, pp. 9661–9670 (2018)

Sanchez-Rodrigues, V., Potter, A., Naim, M.M.: Evaluating the causes of uncertainty in logistics operations. Int. J. Logist. Manag. **21**(1), 45–64 (2010)

Simangunsong, E., Hendry, L.C., Stevenson, M.: Supply-chain uncertainty: a review and theoretical foundation for future research. Int. J. Prod. Res. **50**(16), 4493–4523 (2012)

van der Vorst, J.G., Beulens, A.J.: Identifying sources of uncertainty to generate supply chain redesign strategies. Int. J. Phys. Distrib. Logist. Manag. **32**(6), 409–430 (2002)

Wilding, R.: The supply chain complexity triangle: uncertainty generation in the supply chain. Int. J. Phys. Distrib. Logist. Manag. **28**(8), 599–616 (1998)