SET SELECTION WITH UNCERTAIN WEIGHTS: NON-ADAPTIVE QUERIES AND THRESHOLDS

CHRISTOPH DÜRR, ARTURO MERINO, JOSÉ A. SOTO, AND JOSÉ VERSCHAE

Abstract. We study set selection problems where the weights are uncertain. Instead of its exact weight, only an uncertainty interval containing its true weight is available for each element. In some cases, some solutions are universally optimal; i.e., they are optimal for every weight that lies within the uncertainty intervals. However, it may be that no universal optimal solution exists, unless we are revealed additional information on the precise values of some elements.

In the *minimum cost admissible query* problem, we are tasked to (non-adaptively) find a minimum-cost subset of elements that, no matter how they are revealed, guarantee the existence of a universally optimal solution. This belongs to the setting of explorable uncertainty and while there is a significant body of work in the adaptive setting, non-adaptive versions, such as the one in this paper, are far-less understood.

We introduce *thresholds under uncertainty* to analyze problems of minimum cost admissible queries. Roughly speaking, for every element e, there is a threshold for its weight, below which e is included in all optimal solutions and a second threshold above which e is excluded from all optimal solutions.

We show that computing thresholds and finding minimum cost admissible queries are essentially equivalent problems. Thus, the analysis of the minimum admissible query problem reduces to the problem of computing thresholds.

We provide efficient algorithms for computing thresholds in the settings of minimum spanning trees, matroids, and matchings in trees; and NP-hardness results in the settings of s-t shortest paths and bipartite matching. By making use of the equivalence between the two problems these results translate into efficient algorithms for minimum cost admissible queries in the settings of minimum spanning trees, matroids, and matchings in trees; and NP-hardness results in the settings of s-t shortest paths and bipartite matching.

1. Introduction

We study set selection problems under uncertain weights. Without uncertainty, an instance of *(min-weight) set selection* consists of a *ground set* E, together with weights $w : E \to \mathbb{R}$, and an (implicitly defined) collection of *feasible sets* $\mathcal{F} \subseteq 2^E$. The goal is to find a feasible set $S \in \mathcal{F}$ which minimizes the total weight $w(S) := \sum_{e \in S} w(e)$.

Set selection encodes many classical problems in combinatorial optimization. In particular, it encodes the problems of finding minimum spanning trees, shortest paths, and minimum weight perfect matching. Maximization problems can be similarly modeled by simply multiplying the weights by -1.

In the uncertain setting, we are again given a ground set E and an implicitly defined collection of feasible sets $\mathcal{F} \subseteq 2^E$. However, we do not have precise weights for each element $e \in E$. Instead, for each element $e \in E$, we know an interval $I_e := [\ell_e, h_e]$ in which the true weight w_e lies; i.e., $w_e \in I_e$ (see Figure 1a). We call such an interval the *uncertainty interval* of e and denote the collection of intervals by $\mathcal{I} := \{I_e\}_{e \in E}$. Note that it is possible for these intervals to be singletons, in that case we call them *trivial*. Additionally, every weight function w that obeys $\mathcal{I}$ is called a *realization* of $\mathcal{I}$; i.e., $w \in \prod_{e \in E} I_e$ (see Figure 1b and 1c). Naturally, realizations of $\mathcal{I}$ are the possible *true weights* of an uncertain instance.

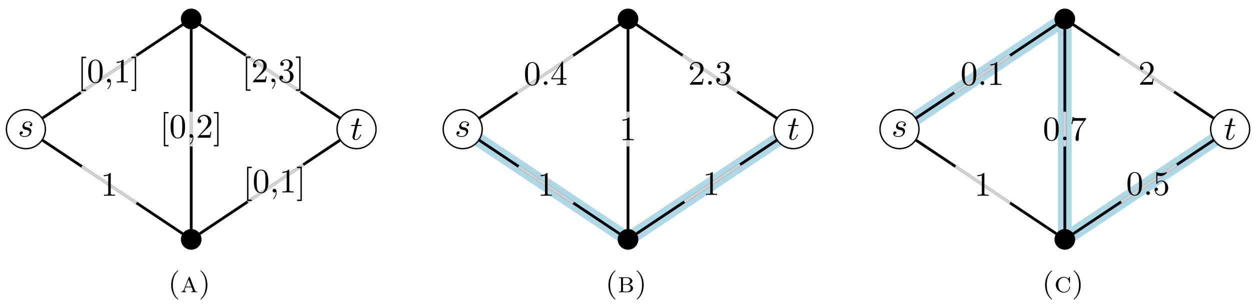

FIGURE 1. (a) An instance of s-t paths with uncertain weights. (b)+(c) Two realizations of the instance in (a) with different optimal solutions marked in blue.

Under uncertainty, we again are interested in minimizing the weight of the chosen solution. Note, however, that this could be realization dependent (see Figures 1b and 1c); i.e., solutions could be optimal for some realizations and not for others. Thus, the best solution we could hope for is a feasible set that has the best weight independently of the realization. More formally, we say that a feasible set $S^* \in \mathcal{F}$ is *universally optimal* if it holds that $w(S^*) \leq w(S)$ for every feasible set $S \in \mathcal{F}$ and every weight realization w of $\mathcal{I}$.

1.1. Minimum cost queries.

Alas, not every uncertain set selection instance has universally optimal solutions (e.g., Figure 1a). To counteract this, we consider the setting where we can obtain more precise information about the weights at a cost. The reader can imagine that the original uncertainty intervals represent a rough estimate of the true weights and that we can spend additional resources to find out precise information about them.

The *(non-adaptive) query problem* models exactly the aforementioned situation: Combinatorial optimization problems where the weights are uncertain, but can be queried at a cost. Querying an element reveals its true weight, replacing the possible weights in can take by a unique element. Here, our objective is to find the least costly set of queries that allows us to compute aa universally optimal solution.

More specifically, we consider the setting where we can make exactly one (non-adaptive) set of queries; i.e., we can select a set $Q \subseteq E$ to be queried, and for every $e \in E$ we will learn its true weight $w_e \in I_e$. A set of elements $Q \subseteq E$ is an *admissible query* if after obtaining any possible answer to the queries in Q, there exists a universally optimal solution (see Figure 2a for an illustration).[1] Note that admissible queries always exist, as querying every element is an admissible query. Querying everything, however, is costly: Querying an element $e \in E$ comes at a cost c_e, which is known in advance. Thus, the objective is to find an admissible query Q that minimizes the total cost $c(Q)$.

We are particularly interested in algorithms that not only compute a minimum cost admissible query, but also compute a universally optimal solution (using the additionally

[1]The standard term in the literature seems to be *feasible queries*. We use the term *admissible queries*, as to make the distinction with the *feasible sets* $S \in \mathcal{F}$.

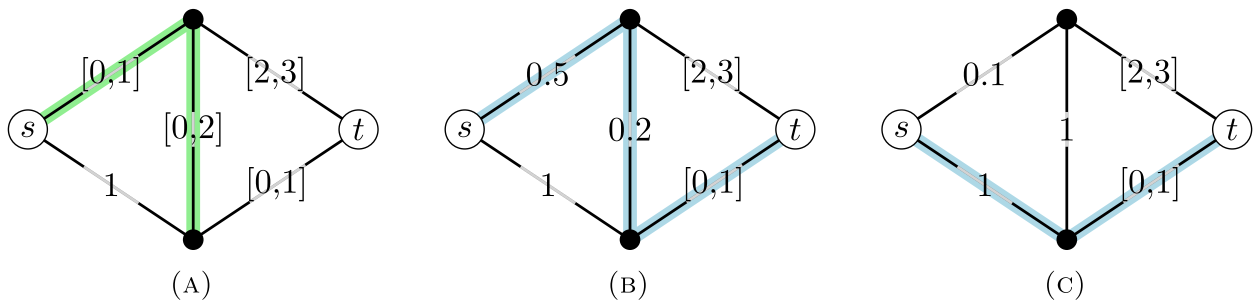

Figure 2. (a) A minimum-sized admissible query marked in green. (b)+(c) Different true weights of the queried edges give rise to different universally optimal solutions (marked in blue).

queried information, see Figure 2b and 2c). More concretely, these algorithms operate in the following stages.

(1) Receive an instance $(E, \mathcal{F}, \mathcal{I})$.
(2) *Query* a set of elements $Q \subseteq E$.
(3) For the queried elements $e \in Q$, we obtain the true weights $w_e \in I_e$.
(4) Return a universally optimal solution $S \subseteq E$ for the intervals $\mathcal{I}'$ given by $\{w_e\}_{e\in Q} \cup \{I_e\}_{e\in E\setminus Q}$; i.e., $w(S^*) \leq w(S)$ for all feasible sets S and for all weight realizations $w \in \prod_{e\in Q}\{w_e\} \times \prod_{e\in E\setminus Q} I_e$.

We highlight that it is *not* required that the algorithm knows the weight of the solution it outputs in the end, only that is *optimal*. In particular, it is not required that the algorithm has queried all elements in the returned solution.

This model is known as the *explorable uncertainty* model and was introduced by Kahan in 1991 [Kah91]. Explorable uncertainty has attracted considerable attention since its introduction; see e.g., the survey of Erlebach and Hoffmann [EH15]. Most of the work has focused on *adaptive* algorithms; i.e., algorithms that can query a few elements, learn their weights, decide whether they query more elements or can compute an optimal solution, and repeat. We usually analyze these algorithms via competitive analysis; i.e., they compare against an adversary that knows the true weights beforehand. Our understanding of the adaptive setting is quite good; almost optimal constant competitive algorithms are known for spanning trees [Erl+08; MMS17; MS19], matroids [EH15], geometric problems [Bru+05], scheduling [LMS19; Dür+20; AE21; Duf+22; Zha23] for slightly different models, and many other settings. Many of the results can even be obtained in a unified manner by using the technique of *witness sets* as introduced by Bruce et al. [Bru+05]. Furthermore, for other natural problems such as maximum weight matching, there are impossibility results showing that no constant-competitive algorithms exist [Mei18, Chapter 3].

Surprisingly, we know very little about non-adaptive algorithms. They are used in settings with a huge fixed setup cost for any number of queries. Merino and Soto [MS19] studied the problem on matroids with uncertainty sets; i.e., not only intervals. They provide polynomial time algorithms for finding minimum cost admissible queries. In particular, for minimum spanning trees, they find minimum cost admissible queries in $\mathcal{O}(m^2\alpha(m,n))$ time; here, n is the number of vertices of the graph, m the number of edges, and α is the inverse Ackermann function. Despite that, their techniques do not seem to easily generalize to other problems.

In particular, they leave open whether one can find minimum cost admissible queries for s-t shortest paths or minimum cost perfect matchings in polynomial time.

1.2. **Thresholds of inclusion and exclusion.** To better understand minimum admissible queries, we study a related problem on set selection under uncertainty. Note that even if an instance has no universally optimal solution, there are elements which belong to *all* optimal solutions and some others which are in *none* for all realizations (see Figure 3a). Identifying these key elements has proven useful to deal with uncertainty; see e.g., [Erl+08; MMS17; MS19].

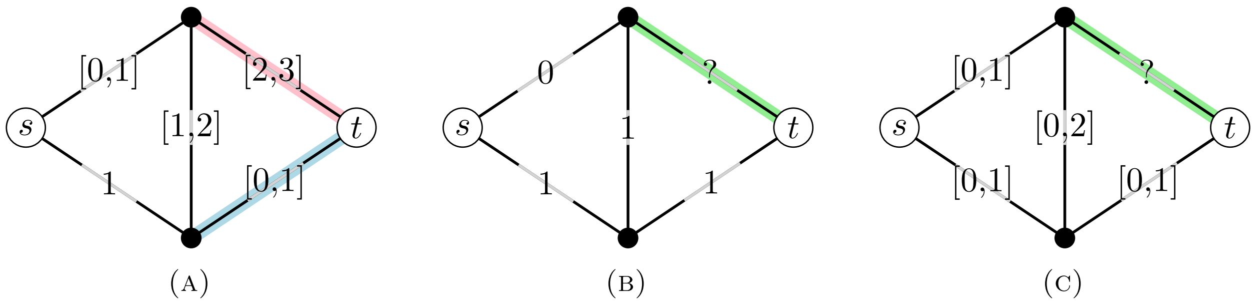

Figure 3. (a) This instance has no universally optimal solution. (b) If the edge marked in green has weight less (resp. more) than 2, then it is contained in *every* (resp. no) s-t shortest path. (c) If the green edge has weight smaller than 1 (resp. larger than 2), then it is included in all (resp. *no*) shortest s-t paths, regardless of the true weights.

In the setting without uncertainty, these ideas are captured by the concept of *thresholds*. Here, we consider a fixed element $e \in E$ and are given all weights except the weight of e. Suppose that there are feasible sets with and without e: When is e contained in an optimal solution? If the weight of e is very small, say $w_e \approx -\infty$, then e must be included in all optimal solutions. Similarly, if the weight of e is very large, say $w_e \approx +\infty$, then every optimal solution must avoid e. The point where this behavior changes is called the *threshold* of e, denoted by T_e (see Figure 3b). More formally, T_e is the unique number such that

- if $w_e > T_e$, then no optimal solution contains e,
- if $w_e < T_e$, then e belongs to every optimal solution,
- if $w_e = T_e$, there are optimal solutions with and without e.

Naturally, the threshold T_e is a function of the weights of all other elements w_{-e}. Furthermore, it is not hard to see that T_e is a piecewise linear function in each coordinate and can be easily computed whenever you can solve the corresponding min-weight set selection problem (see Section 2 for details).

In the uncertain setting, things are a bit more complicated, as we do not have a unique threshold that marks the two regimes (see Figure 3c). For every item $e \in E$, we define the *threshold of inclusion* T_e^+ and the *threshold of exclusion* T_e^- as follows

$$T_e^+ := \inf\{T_e(w_{-e}) \mid w_{-e} \text{ realization of } \mathcal{I} \setminus \{I_e\}\}$$
$$T_e^- := \sup\{T_e(w_{-e}) \mid w_{-e} \text{ realization of } \mathcal{I} \setminus \{I_e\}\}.$$

In other words, T_e^+ is the minimum possible threshold, and T_e^- is the largest possible threshold. Note that $T_e^+ = T_e^- = \infty$ whenever e is in *every* feasible set; similarly, $T_e^+ = T_e^- = -\infty$ whenever e is in *no* feasible set. The interval $[T_e^+, T_e^-]$ indicates the possible behavior of e with respect to optimal solutions.

- If $h_e < T_e^+$, then e belongs to any optimal solution for every realization of $\mathcal{I} \setminus \{I_e\}$.
- If $\ell_e > T_e^-$, then e does not belong to any optimal solution for every realization of $\mathcal{I} \setminus \{I_e\}$.
- Otherwise, there are optimal solutions with and without e for specific realizations of $\mathcal{I} \setminus \{I_e\}$.

Also, the union of $T_e(w_{-e})$ over all realizations w_{-e} of $\mathcal{I} \setminus \{I_e\}$ is exactly the interval $[T_e^+, T_e^-]$.

The existence of thresholds is central to the isolation lemma, a technique from the 1980s to guarantee uniqueness of optimal solutions; see, e.g., [Ta-15] and references therein. Despite its notoriety, to the best of our knowledge, the problem of computing thresholds under uncertain weights has not been studied prior to this work.

1.3. **Our contribution.** Our main contribution is the introduction of thresholds under uncertainty and using them to analyze minimum cost admissible queries. As a first result, we show that computing thresholds and finding minimum cost admissible queries are essentially equivalent.

- Given the thresholds of inclusion and exclusion for every element, we can compute a minimum cost admissible query in linear time. We can compute a universally optimal solution by solving only one extra set selection problem. (See Theorem 10.)
- Computing thresholds up to a constant additive error reduces to solving a logarithmic number of minimum cost admissible query problems (see Theorem 12).

After showing this equivalence, we turn towards computing thresholds in various settings.

- For the minimum spanning tree problem, we show a $\mathcal{O}(m\alpha(m, n) + n)$ algorithm for computing all $2m$ thresholds (see Theorem 19).
- For the maximum weight matching problem on trees, we show a linear time algorithm that given element $e \in E$ it computes its thresholds (see Theorem 25).

In other settings, we are able to show hardness. In particular, we show that computing the thresholds is NP-complete for shortest paths on DAGs (see Theorem 20) and for bipartite minimum cost perfect matching (see Theorem 23).

As a consequence of the equivalence, we can translate the aforementioned results into the setting of minimum cost admissible queries, obtaining the following results.

- An $\mathcal{O}(m\alpha(m, n) + n)$ time algorithm for finding minimum cost admissible queries in the setting of minimum spanning trees, improving on the $\mathcal{O}(m^2\alpha(m, n))$ algorithm of Merino and Soto [MS19].
- A quadratic time algorithm for minimum cost admissible queries for maximum matching on trees.
- Deciding whether a given set is an admissible query for s-t shortest paths is NP-complete (see Observation 21)
- Deciding whether a given set is an admissible query for bipartite minimum cost perfect matching is NP-complete (see Observation 24).

These last two points imply that there is no polynomial time algorithms for finding minimum cost admissible queries in the setting of shortest paths and bipartite minimum cost perfect matching (unless P = NP).

1.4. **Further related work.**

Besides what was mentioned in Subsection 1.1, there are a couple of other explorable uncertainty settings related to this work.

Stochastic uncertainty. In this model instead of working only with uncertainty intervals as prior information on the weights, they are given additional stochastic information. Again, the objective is to minimize the cost of querying while being able to compute an optimal solution. Particularly relevant to us is the work of Megow and Schlöter that explores adaptive algorithms for set selection problems [MS23].

Value-based models. There are models with the additional requirement that the produced solution must belong to the set of queried element; see e.g., the work of Singla [Sin18]. In other words, the algorithm not only needs to produce an optimal solution, but also the value of the solution.

The optimality condition is sometimes relaxed, allowing to output a solution that is additively suboptimal by some pre-specified constant; see e.g., the work of Feder et. al. [Fed+00; Fed+07]. We highlight that these last works are non-adaptive, however since their setting requires finding the value, it has a very different flavor.

2. Threshold preliminaries

We begin by analyzing basic properties of the threshold functions in both without and with uncertainty.

2.1. **Thresholds without uncertainty.**

Note that for every $e \in E$ can easily compute T_e as follows. Let $\mathrm{OPT}^{-e}(w_{-e})$ be the weight of an optimal feasible set S under the constraint $e \notin S$, and let $\mathrm{OPT}^{+e}(w_{-e})$ be the similar weight under the constraint $e \in S$ and imposing $w_e = 0$. (See Figure 4a.) We have

$$T_e(w_{-e}) = \mathrm{OPT}^{-e}(w_{-e}) - \mathrm{OPT}^{+e}(w_{-e}).$$

How does each of the terms $\mathrm{OPT}^{-e}(w_{-e})$ and $\mathrm{OPT}^{+e}(w_{-e})$ depend on the weight of another element $f \in E$? Consider the optimal feasible set without e, f, and the optimal feasible set without e but with f. The weight of the first is constant in w_f, while the second is linear (with slope 1) in w_f. Thus, $\mathrm{OPT}^{-e}(w_{-e})$ is a constant function of w_f up to some point, from which it has a unit slope. Similarly, $\mathrm{OPT}^{+e}(w_{-e})$ is a similar function in w_f, but with a different constant part and different threshold. Hence, their difference $T_e(w_{-e})$ is a piecewise linear function in w_f, consisting of three parts. The first and third parts are constant, while the second part has a slope of either $+1$ or -1, depending on how the thresholds compare to each other (see Figure 4b). As a consequence, we have that T_e is continuous and monotone in each coordinate.

The next lemma states that OPT^{+e} and OPT^{-e} are structurally very similar. Intuitively, this holds as we are optimizing a linear problem along the same direction and only making a small change in the restrictions.

We formalize this in the context of polytopes. Given a set of vectors $X \subseteq \mathbb{R}^d$ its *convex hull* $\mathrm{conv}(X)$ consists of all convex combinations of vectors in X. For $F \subseteq E$ we denote by

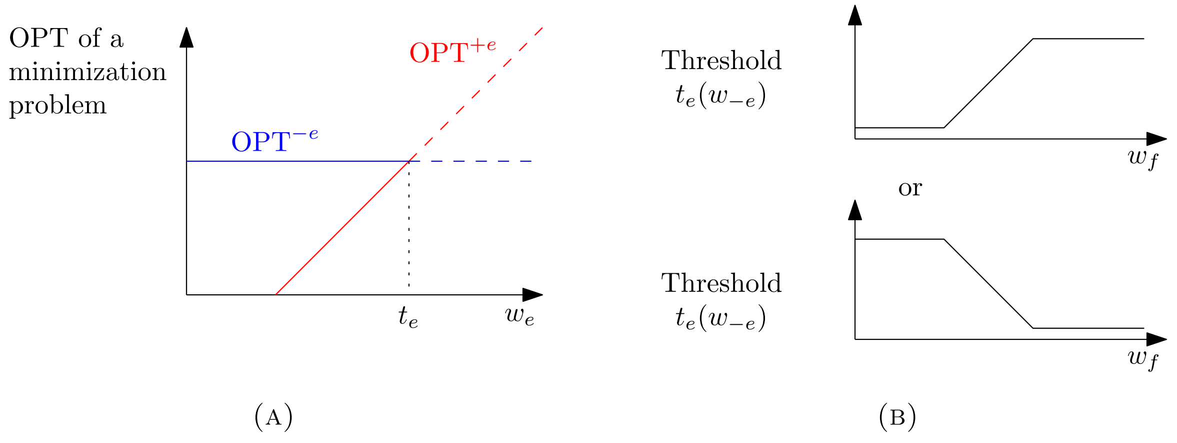

FIGURE 4. (a) Behaviour of OPT^{-e} and OPT^{+e} when we change the weight of e. The threshold T_e is exactly the point at which these two values are the same. (b) Schematic view of this threshold as a function of the weight of some other element f.

$\chi_F \in \{0,1\}^E$ the *characteristic vector* of F. For $x, y \in \{0,1\}^n$ their *Hamming distance* is $d(x,y) = \sum_{i=1}^n |x_i - y_i|$.

Lemma 1. *Let $P := \text{conv}(\{\chi_F \mid F \in \mathcal{F}\})$, $w \in \mathbb{R}^E$, and $e \in E$. There exists $x', y' \in \{0,1\}^E$ such that $w \cdot x' = \min\{w \cdot x \mid x \in P,\ x_e = 0\}$, $w \cdot y' = \min\{w \cdot x \mid x \in P,\ x_e = 1\}$, and xy is an edge of the polytope P.*

Proof. Choose two vertices $x', y' \in \{0,1\}^E$ of P such that $x' \in \text{argmin}\{w \cdot x \mid x \in P,\ x_e = 0\}$, $y' \in \text{argmin}\{w \cdot x \mid x \in P,\ x_e = 1\}$, and the Hamming distance between x' and y' is minimized. Suppose that x' and y' are not adjacent in P. Then, there exists $\mu \in (0,1)$ such that $z := \mu x' + (1-\mu)y'$ is a convex combination of the vertices of P except x' and y'; i.e., $z = \sum_{i=1}^k \lambda_i z^i$ where for every $i \in [k]$ we have $\lambda_i \in (0,1)$ and that $z^i \in \{0,1\}^E$ is a vertex of P different from x' or y'.

Let us define $I_0 = \{i \in [k] \mid z^i_e = 0\}$ and $I_1 = \{i \in [k] \mid z^i_e = 1\}$. Note that $\sum_{i \in I_0} \lambda_i = \mu$ and $\sum_{i \in I_1} \lambda_i = (1-\mu)$. Furthermore, for every $i \in I_0$ we have that $w \cdot z^i = w \cdot x'$ and for every $i \in I_1$ we have that $w \cdot z^i = w \cdot y'$, as otherwise we have that

$$w \cdot z = w \cdot \left(\sum_{i \in I_0} \lambda_i z^i + \sum_{i \in I_1} \lambda_i z^i \right) > \mu(w \cdot x) + (1-\mu)(w \cdot y) = w \cdot z,$$

which is not possible. Thus, we conclude that $z^i \in \text{argmin}\{w \cdot x \mid x \in P, x_e = 1\}$ for $i \in I_1$ and that $z^i \in \text{argmin}\{w \cdot x \mid x \in P, x_e = 0\}$ for $i \in I_0$.

Let us define A as the coordinates in which x' and y' agree; i.e., $A = \{f \in E \mid x'_f = y'_f\}$. Note that for $f \in A$ we have that $z^i_f = x'_f = y'_f$ for every $i \in [k]$. Furthermore, $z^i = x'$ whenever $i \in I_0$; otherwise, there exists $i^* \in I_0$ such that z^{i^*} agrees in more than $|A|$ coordinates with y', hence $d(z^{i^*}, y') < |E| - |A| = d(x', y')$ and contradicting the minimality of $d(x', y')$. A similar argument shows that $z^i = y'$ whenever $i \in I_1$. This contradicts the fact that z^i was not x' nor y' for every $i \in [k]$. $\square$

Lemma 1 has interesting consequences when looking at specific problems. In these problems edges of the corresponding polytope are well-understood. For example, for minimum spanning trees, a result due to Hausmann and Korte implies that two trees are adjacent if and only if they differ in the exchange of two edges. Combining this with Lemma 1 we obtain that.

Corollary 2. *Let $G = (V, E)$ be a graph with weights. There exists an edge $f \in E - e$ and a spanning tree T^+ such that T^+ is a spanning tree that contains $e \in E$ of minimum weight, and $T^- := T^+ - e + f$ is a spanning tree that avoids $e \in E$ of minimum weight.*

Analogous statements also hold for other set selection problems; see e.g., matchings and perfect matchings [Chv75], *s*-*t* paths and flows [GS78].

2.2. **Thresholds with uncertainty.** A weight realization w such that $w_f \in \{\ell_f, h_f\}$ for every $f \in E$ is said to be *extreme*. Observe that by monotonicity of T_e — which can be non-increasing or non-decreasing, see Figure 4 — we have $T_e^+ = T_e(w_{-e})$ for some extreme realization w_{-e}. The same holds for T_e^-.

This means that the threshold of inclusion can be computed by minimizing over $2^{|E|}$ different extreme weights, which implies the following observation.

Observation 3. *Suppose that the underlying optimization problem can be solved in polynomial time for a fixed weight realization. The decision version of computing the threshold of inclusion asks if T_e^+ is at most some given value. The decision version of computing the threshold of exclusion asks if T_e^- is at least some given value. Both problems are in NP, and the certificates are the corresponding extremal realizations.*

We can obtain an alternative characterization of the threshold of inclusion. For a given realization w_{-e}, let $S^- \in \mathcal{F}$ be the feasible solution with optimal value $\mathrm{OPT}^{-e}(w_{-e})$, and $S^+ \in \mathcal{F}$ be the feasible solution with optimal value $\mathrm{OPT}^{+e}(w_{-e})$. Then we can assume without loss of generality that the realization sets w_f to its lower bound for all $f \in S^-$, and to its upper bound for all $f \notin S^-$. In other words T_e^+ is the minimum among all $S \in \mathcal{F}$ with $e \notin S$ of $\ell(S) - \mathrm{OPT}^{+e}(w)$, where $w_e = 0$, and $w_f = \ell_f$ for all $f \in S$ and $w_f = h_f$ otherwise, conditioned on S being an optimal solution with respect to these weights w and the constraint $e \notin S$.

Similarly, the threshold of exclusion can be expressed as the maximum over all $T \in \mathcal{F}$ with $e \in T$ of the difference $\mathrm{OPT}^{-e}(w) - \ell(T)$, where $w_f = \ell_f$ for all $f \in T$ and $w_f = h_f$ otherwise, conditioned on T being an optimal solution with respect to these weights w and the constraint $e \in T$.

3. From thresholds to minimizing queries, and back

In this section, we prove that the problem of computing thresholds and finding minimum cost admissible queries are effectively equivalent.

3.1. **The query problem reduces to threshold computation.** We begin by classifying elements according to their relation to thresholds. We say that $e \in E$ is *blue* if I_e is non-trivial and $h_e \leq T_e^+$. Similarly, we say that $e \in E$ is *red* whenever I_e is non-trivial and $T_e^- \leq \ell_e$. Intuitively, blue elements are *safe* to include in an optimal solution, and red elements are *safe* to delete from optimal solutions.

We collect some simple observations.

Lemma 4. *No element $e \in E$ is blue and red.*

Proof. Let $e \in E$ be an element that is blue and red. Thus, $h_e \leq T_e^+ \leq T_e^- \leq \ell_e$, which implies that I_e is trivial, a contradiction. $\square$

Therefore, the elements are partitioned into blue, red, trivial, and uncolored non-trivial. We also note that colors are preserved under queries.

Lemma 5. *Let $Q \subseteq E$ be a set to be queried. If $e \notin Q$ was blue (resp. red) before querying Q, then it is blue (resp. red) after querying Q.*

Proof. We consider an instance with uncertainty intervals $\mathcal{I} = \{I_e\}_{e\in E}$. After querying a set $F \subseteq E$ and obtaining the true weights $w_e \in I_e$ for $e \in E$, we have a new instance with uncertainty intervals $\mathcal{I}' := \{\{w_e\}\}_{e\in F} \cup \{I_e\}_{e\in E\setminus F}$. For every $e \in E$, let T_e^+, T_e^- (resp. $T_{e,Q}^+, T_{e,Q}^-$) be the thresholds for e with intervals $\mathcal{I}$ (resp. $\mathcal{I}'$). Note that every realization of $\mathcal{I}'$ is a realization of $\mathcal{I}$. Thus, $T_e^+ \leq T_{e,Q}^+$ and $T_{e,Q}^- \leq T_e^-$ and the lemma follows. $\square$

Furthermore, as already noted by Merino and Soto [MS19], in the context of matroids, supersets of admissible queries are admissible.

Lemma 6. *If $F \subseteq E$ is an admissible query, all its supersets $F \subseteq F'$ are admissible queries.*

The key observation is that uncolored non-trivial elements *must* be queried.

Lemma 7. *If $F \subseteq E$ is an admissible query, then it contains the set*

$$Q = \{e \in E \mid e \textit{ is uncolored and } I_e \textit{ is non-trivial}\}.$$

Proof. Suppose, aiming at a contradiction, that there is an admissible query F and an element $e^* \in Q \setminus F$. Let $A_{e^*} = (\ell_{e^*}, h_{e^*}) \cap [T_{e^*}^+, T_{e^*}^-]$. Since $h_{e^*} > T_{e^*}^+$ and $\ell_{e^*} < T_{e^*}^-$, the set A_{e^*} is nonempty. Choose $w^* \in A_{e^*}$ and and $\epsilon > 0$ such that $(w^* - \epsilon, w^* + \epsilon) \subseteq I_{e^*}$. Since $T_{e^*}^+ \leq w^* \leq T_{e^*}^-$, we have that there is a realization $w \in \mathbb{R}^E$ such that e^* is in an optimal solution S^+, e^* avoids an optimal solution S^-, and $w_{e^*} = w^*$. Furthermore, we define $w^- \in \prod_{e\in E} I_e$ as w except that in the coordinate e^* we have that $w_{e^*}^- = w^* + \epsilon$. Similarly, we define $w^+ \in \prod_{e\in E} I_e$ as w except that in the coordinate e^* we have that $w_{e^*}^+ = w^* - \epsilon$.

Suppose now we query $E - e^*$. Since $F \subseteq E - e^*$ was an admissible query, by Lemma 6 we obtain that $E - e^*$ is also admissible. Thus, even if we reveal the weights on $E - e^*$ to be w_{-e}, there exists a feasible set $S \in \mathcal{F}$ that is optimal independently of the value of e^*. If the true value of e^* is w^*, we have that $w(S) = w(S^-) = w(S^+)$, as they are all optimal solutions. Now, if the true value of e^* is $w^* - \epsilon$, we have that $e^* \in S$ (otherwise $w^+(S^+) < w^+(S)$). Finally, if the true value of e^* is $w^* + \epsilon$, we have that $e^* \notin S$ (otherwise $w^-(S^-) < w^-(S)$). Thus, we have that $e^* \in S$ and $e^* \notin S$, a contradiction. $\square$

In view of Lemma 7, we only need to figure out what to do with blue and red elements. The next lemma shows that it is always safe to pick blue elements in optimal solutions and avoid red elements.

Lemma 8. *For every realization $w \in \prod_{e\in E} I_e$, there is a w-optimal solution S^* such that*

$$\{e \in E \mid e \textit{ is blue}\} \subseteq S^* \subseteq \{e \in E \mid e \textit{ is not red}\}.$$

Proof. Let $B := \{e \in E \mid e \text{ is blue}\}$ and $R := \{e \in E \mid e \text{ is red}\}$. Let S be a w-optimal solution. We also define

$$g := \min\{w(S') \mid S' \subseteq E \text{ and } B \subseteq S' \subseteq E \setminus R\} - w(S) \geq 0.$$

Assume, for the sake of contradiction, that $g > 0$.

Let $\epsilon := \frac{g}{4|E|}$. For $e \in B$, choose $\delta_e \in (-\epsilon, \epsilon)$ such that $w_e + \delta_e < T_e^+$ and $w_e + \delta_e \in I_e$. Similarly, for $e \in R$, we choose $\delta_e \in (-\epsilon, \epsilon)$ such that $w_e + \delta_e > T_e^-$ and $w_e + \delta_e \in I_e$. For the remaining elements $e \in E \setminus (B \cup R)$, we set $\delta_e = 0$.

We now define $\hat{w} \in \prod_{e \in E} I_e$ as $\hat{w} := w + \delta$. Let $\hat{S}$ be a $\hat{w}$-optimal solution. Since for $e \in B$ we have that $\hat{w}_e < T_e^+$, this implies that $B \subseteq \hat{S}$. Similarly, for $e \in R$ we have that $\hat{w}_e > T_e^-$, which implies that $\hat{S} \subseteq E \setminus R$. Therefore,

$$w(\hat{S}) - w(S) \geq \min\{w(S) \mid S \subseteq E \text{ and } B \subseteq S \subseteq E \setminus R\} - w(S) = g. \tag{1}$$

Since $\hat{S}$ is $\hat{w}$-optimal, we have that $\hat{w}(\hat{S}) \leq \hat{w}(S)$. As a consequence, we have that $w(\hat{S}) + \delta(\hat{S}) \leq w(S) + \delta(S)$. Thus,

$$w(\hat{S}) - w(S) \leq \delta(S) - \delta(\hat{S}) \leq \epsilon(|S| + |\hat{S}|) \leq 2\epsilon|E| = \frac{g}{2}. \tag{2}$$

Combining (1) with (2) we obtain that $\frac{g}{2} \geq g$, a contradiction. □

We can combine Lemma 7 and Lemma 8 to show that the uncolored non-trivial elements are the *unique* minimum-sized admissible query. Therefore, there is a natural algorithm for finding minimum cost admissible queries: Query the uncolored non-trivial elements and every element of negative cost. We can extend this to an algorithm that also computes universally optimal solutions as follows.

Algorithm 9 *(Minimum cost admissible queries and universally optimal solutions).*

```
(1) For every $e \in E$, compute the thresholds $T_e^+, T_e^-$.
(2) Compute $B \leftarrow \{e \in E \mid e \text{ is blue}\}$, $R \leftarrow \{e \in E \mid e \text{ is blue}\}$,
        $T \leftarrow \{e \in E \mid I_e \text{ is trivial}\}$ and $Q \leftarrow E \setminus (B \cup R \cup T)$.
(3) Query $Q$ and every element of negative cost.
        Obtain true weights $w_e \in I_e$ for $e \in Q$.
(4) Let $M \leftarrow \sum_{e \in T \cup Q} |w_e| + \sum_{e \in B \cup R} |\ell_e| + |h_e|$.
(5) For every $e \in B$ assign $w_e \leftarrow -M$ and for every $e \in R$ assign $w_e \leftarrow M$.
(6) Return an element of $\operatorname{argmin}\{w(S) \mid S \in \mathcal{F}\}$ as a universally optimal solution.
```

Theorem 10. *Suppose all thresholds for $(E, \mathcal{F}, \mathcal{I})$ can be computed in time T_{THR} and that the optimization problem on $(E, \mathcal{F})$ can be solved in time T_{OPT}. We can find a minimum cost admissible query problem in time $\mathcal{O}(T_{THR} + |E|)$, and after the weights of the queries are revealed we can find a universally optimal solution in time $\mathcal{O}(T_{OPT} + |E|)$*

Proof. We show that Algorithm 9 solves the minimum cost admissible query problem, since it is clear that Algorithm 9 runs in the desired running time.

We now show that Q is a admissible query. After Algorithm 9 queries Q, it obtains true values $w_e \in I_e$. We aim to show that the algorithm returns an optimal solution for every realization w of the intervals after querying $\mathcal{I}'$; that is $\mathcal{I}' := \{I_e\}_{e \in E \setminus Q} \cup \{\{w_e\}\}_{e \in Q}$. Let $M := \sum_{e \in T \cup Q} |w_e| + \sum_{e \in B \cup R} |\ell_e| + |h_e|$. We also define $w_e = -M$ for blue e and $w_e = M$ for red e as in Algorithm 9. Let S' be a w-optimal solution. It is easy to see that S' contains every blue element, and it avoids every red element. Choose any $w^* \in \mathcal{I}'$. By Lemma 8, we have that there is a w^*-optimal solution S^* such that

$$\{e \in E \mid e \text{ is blue}\} \subseteq S^* \subseteq \{e \in E \mid e \text{ is not red}\}.$$

Note that since $w|_Q = w^*|_Q$ and $w|_T = w^*|_T$, we have that

$$-M|B| + w(S^* \cap (Q \cup T)) = w(S^*) \geq w(S') = -M|B| + w(S' \cap (Q \cup T)).$$

We conclude that $w^*(S' \cap (Q \cup T)) \leq w^*(S^* \cap (Q \cup T))$. Using that we obtain

$$w^*(S') = w^*(B) + w^*(S' \cap (Q \cup T)) \leq w^*(B) + w(S^* \cap (Q \cup T)) = w^*(S^*).$$

Hence, S' is w^*-optimal, and in turn Q is an admissible query.

The fact that Q is of minimum cost follows directly from Lemma 7. □

3.2. **Threshold computation reduces to the query problem.** We show now how to additively approximate thresholds by using minimum cost admissible queries. In particular, we only use a weaker oracle that tests whether $E - e$ is an admissible query. Naturally, this oracle can be simulated with a minimum cost admissible query oracle by using the costs $c_e = 1$ and $c_f = 0$ for $f \in E - e$.

We reason for T_e^+ in this section, as the arguments for computing T_e^- are dual. The main idea is that for a given $e \in E$ and $\alpha \in \mathbb{R}$ we can check whether $T_e^+ < \alpha$ by using the aforementioned oracle. To this end, given uncertainty intervals I_f for $f \in E - e$, we define $K = -\sum_{f \in E-e} |\ell_f| + |h_f|$, $I_e^\alpha = [K - 1, \alpha]$ and $\mathcal{I}^\alpha = \{I_f\}_{f \in E-e} \cup \{I_e^\alpha\}$. Note that T_e^+ is the same for $\mathcal{I}$ and $\mathcal{I}^\alpha$. First, we show that we can assume that $T^- \neq -\infty$.

Recall that we can test whether $T_e^- \neq -\infty$ by running one optimization problem and setting the weight of e to be very small, namely the negation of the total weight upper bounds. Thus, we assume in the next lemma that $T_e^- \neq -\infty$

Lemma 11. *$T_e^+ \geq \alpha$ for $\mathcal{I}$ if and only if $E - e$ is an admissible query for $\mathcal{I}^\alpha$.*

Proof. If $T_e^+ \geq \alpha = h_e$, then e is blue, and therefore $E - e$ is an admissible query.

For the converse, note that $T_e^+ < \alpha = h_e$, implies that e is not blue. Furthermore, since $T_e^- \neq -\infty$, then $T_e^- \geq \ell(E - e)$, implies that e is not red. Hence, e is uncolored and must be queried, which implies that $E - e$ is not admissible. □

As a consequence, we can simply apply Lemma 11 and obtain an algorithm that computes thresholds with additive error.

Theorem 12. *Suppose we can decide whether $E - e$ is an admissible query in time T, and that we can solve the corresponding optimization problem in time T_{OPT}. For every $e \in E$ and every additive precision $\delta > 0$, we can compute $a, b \in [-K, K]$ such that $T_e^+ \in [a, a + \delta]$ and $T_e^- \in [b, b + \delta]$ (or decide whether $T_e^-, T^+e = \pm\infty$). This algorithm runs in time $\mathcal{O}\left(T \log(\frac{2K}{\delta}) + T_{OPT}\right)$.*

Proof. We only need to check beforehand whether the thresholds are $\pm\infty$ and then run the corresponding binary search. □

4. Minimum spanning trees

In this section we consider the classical minimum spanning tree problem (MST). A *minimum spanning tree* of a graph $G = (V, E)$ with weights $w : E \to \mathbb{R}$ is a spanning tree T of G that minimizes $w(T)$. We show algorithms for computing the thresholds of inclusion and exclusion for MSTs. We assume that the edge $e = uv$ for which we want to compute the weights is not a *bridge*, also known as *cut-edge*, otherwise both thresholds are $+\infty$.

4.1. **Thresholds via greedy.**

We begin by giving a simple greedy-based algorithm for computing thresholds.

First, we recall the classical *red* and *blue* rules for computing MSTs.

- Red rule: edge e is in no MST if and only if $w_e > \min_{C \text{ cycle}, e \in C} \max_{f \in C-e} w_f$.
- Blue rule: edge e belongs to all MST if and only if $w_e < \max_{C \text{ cut}, e \in C} \min_{f \in C-e} w_f$.

Direct application of the red rule implies that T_e^+ is the minimum over all cycles C containing e of $\max \ell_f$ over $f \in C - e$. Similarly, by applying the blue rule, we obtain T_e^- is the maximum over all cuts C containing e of $\min h_f$ over $f \in C - e$. As a result one can compute T_e^+ and T_e^- via Kruskal's algorithm and reverse-Kruskal.

The standard implementation of these procedures achieves a running time of $\mathcal{O}(m \log n)$ and $\mathcal{O}(m \log n (\log \log n)^3)$ for computing T_e^- and T_e^+ respectively [Tho].

We note that the ideas in this subsection generalize straightforwardly to the *matroid* setting, obtaining efficient algorithms for computing the thresholds of inclusion or inclusion.

4.2. **Thresholds via bottleneck paths.**

Given an undirected graph $G = (V, E)$ with weights $w \in \mathbb{R}^E$ and a path P, the *bottleneck of the path* P is given by $\max_{e \in P} w_e$. A *bottleneck path* P between $u, v \in V$ is a path of minimum bottleneck. The *bottleneck between u and v* is the bottleneck of a bottleneck u-v path. A classical result of Hu [Hu61] states that the paths in a minimum spanning tree are bottleneck paths.

Theorem 13 ([Hu61])**.** *Let $G = (V, E)$ be a graph with weights $w \in \mathbb{R}^E$ and let T be an MST of G. For every $u, v \in V$ the unique u-v path in T is a u-v bottleneck path.*

We consider a slight variation of the bottleneck problem. We say that a path is *trivial* if it is of length one, otherwise, we say it is *non-trivial*. The non-trivial bottleneck between u and v, denoted by $b_w(u, v)$, is the minimum bottleneck of a non-trivial u-v path; i.e.,

$$b_w(u, v) = \min_{P \text{ } u\text{-}v \text{ nontrivial path}} \max_{e \in P} w_e.$$

In other words, the non-trivial bottleneck between u and v in G, is the bottleneck between u and v in $G - uv$.

We show that computing non-trivial bottlenecks is equivalent to computing thresholds.

Lemma 14. *Let $G = (V, E)$ be a graph with weights w. For every $uv \in E$, we have that $T_{uv}(w_{-uv}) = b_w(u, v)$.*

Proof. Let T be a minimum spanning tree, we consider two cases.

- **Case 1:** $e \notin T$. Let T' be $\mathrm{OPT}^{+e}(w_{-e})$. By Corollary 2 we may assume that $T' = T + e - f$. Let P be the unique u-v path in T. Furthermore, f must be an edge of maximum weight in P. Otherwise, there is an edge $f' \in P$ such that $w_{f'} > w_f$, which implies that $w(T+e-f') < w(T+e-f)$, a contradiction. Since P was a u-v bottleneck path by Theorem 13, we conclude that

$$T_e(w_{-e}) = \mathrm{OPT}^{-e}(w_{-e}) - \mathrm{OPT}^{+e}(w_{-e}) = w_f = b_w(u,v).$$

- **Case 2:** $e \in T$. Let T' be $\mathrm{OPT}^{-e}(w_{-e})$. By Corollary 2 we may assume that $T' = T - e + f$. Note that T' is an MST of $G - e$. Let P be the unique u-v path in T'. As before, f must be an edge of maximum weight in P. Otherwise, there is an edge $f' \in P$ such that $w_{f'} > w_f$, which implies that $w(T'+e-f') < w(T'+e-f) = w(T)$, a contradiction. Since P was a u-v bottleneck path in $G - e$ by Theorem 13, then it is a non-trivial u-v path of minimum bottleneck in G. Thus, we conclude that

$$T_e(w_{-e}) = \mathrm{OPT}^{-e}(w_{-e}) - \mathrm{OPT}^{+e}(w_{-e}) = w_f = b_w(u,v). \qquad \square$$

Combining Lemma 14 with the fact that $b_w(u,v)$ is increasing with respect to w, we obtain that for every $uv \in E$ it holds that $T^+_{uv} = b_\ell(u,v)$ and $T^-_{uv} = b_h(u,v)$. Thus, the problem of computing the threshold for $e \in E$ is equivalent to computing $b_\ell(u,v)$ and $b_h(u,v)$.

We now provide a fast algorithm for computing all thresholds in the minimum spanning tree setting. Given a minimum spanning tree T and $e \in T$, we say that f is a *replacement* of e in T if $T+f-e$ is a minimum spanning tree of $G-e$. Furthermore, if the replacement of uv is xy and x is connected to u in $T-e$, we say that x is the replacement of u. Replacements can be computed very efficiently by using the technique of path compression, as shown by Tarjan [Tar79].

Lemma 15. *Given a weighted graph and a minimum spanning tree T, we can compute all $n-1$ replacements of T in time $\mathcal{O}(m\alpha(m,n)+n)$.*

We also make use of the computation of bottlenecks in a tree. This is usually a subroutine of minimum spanning tree verification algorithms, originally shown to run in linear time by Dixon, Rauch, and Tarjan [DRT92], with further simplifications by King [Kin97] and Hagerup [Hag10].

Lemma 16. *Given a tree T with weights on the edges and k pairs of vertices $(u_1,v_1),\dots,(u_k,v_k)$, we can compute the bottlenecks in T of the k pairs $b_T(u_1,v_1),\dots,b_T(u_k,v_k)$ in time $\mathcal{O}(n+k)$.*

The last ingredient we need is Chazelle's algorithm for minimum spanning trees [Cha00].

Lemma 17. *Given a weighted graph, its minimum spanning tree can be computed in time $\mathcal{O}(m\alpha(m,n))$.*

We combine the aforementioned algorithms to obtain the following algorithm for non-trivial bottlenecks.

Algorithm 18 *(Non-trivial bottlenecks).*

(1) Compute a minimum spanning tree T via Chazelle's algorithm.
(2) For every edge $e \in T$, compute its replacement $r_T(e)$ via path compression.
(3) Using the tree-bottleneck algorithm, we compute $b_T(u,v)$ for $(u,v) \in E \setminus T$, and for every $uv \in T$ we compute $b_T(x,u)$ where x is the replacement of u and y the replacement of v.
(4) For $uv \notin T$, return $b_T(u,v)$. Otherwise, let x be the replacement of u and y the replacement of v, and return $\max\{b_T(u,x), w(xy), b_T(v,y)\}$.

We can use Algorithm 18 using ℓ (resp. h) as weights to compute all thresholds T^+ (resp. T^-) for every $e \in E$.

Theorem 19. *For the minimum spanning tree problem, we can compute all thresholds of inclusion and exclusion in time $\mathcal{O}(m\alpha(m,n)+n)$.*

Proof. By Lemma 14 it suffices to compute non-trivial bottlenecks for every $e \in E$ with weights h and ℓ. It is clear that Algorithm 18 runs in the desired time, thus we only prove correctness.

Let T be the minimum spanning tree used by Algorithm 18. For $uv \notin T$, it is clear that T is a minimum spanning tree of $G - uv$. Thus, $b_T(u,v)$ is the non-trivial bottleneck for uv by Theorem 13. If $uv \in T$, let x be the replacement of u in T and y the replacement of v in T. Let P^{ux} be the unique ux-path in T, and P^{yv} be the unique yv-path in T. Then, $T' := T + xy - uv$ is a minimum spanning tree of $G - uv$ and the unique uv path in T' is $P' := P^{ux}P^{yv}$. Since the bottleneck of P' is $\max\{b_T(u,x), w(xy), b_T(y,v)\}$ we conclude by Theorem 13. □

5. Shortest paths

We consider the problem of computing a shortest path between two given vertices s, t, with non-negative uncertain edge weights. In [Fed+07] this problem has been studied in a model, where the algorithm needs to compute the length of the shortest path using as few queries as possible. However, in our setting it is enough to produce some path guaranteed to be the shortest one, even though its exact length could be uncertain.

Consider an edge $e = (u,v)$. The threshold of exclusion T_e^- can be stated as the maximum over all weight realizations w of the difference between the length ℓ of a shortest path from s to t without using e and the length of a shortest path P from s to t forced to use e.

Without loss of generality we can assume that all edges of P have their weight at their lower limit, and all edges not in P have their weight at their upper limit. The idea is that if we increase the weight of an edge not in P, then the length of P does not change, while the ℓ might increase, even though the corresponding path might change. Also if we decrease the weight of an edge in P by some amount $\delta > 0$, then the length of P decreases by δ, while ℓ might decrease by at most δ.

Theorem 20. *Computing the threshold of inclusion of an edge for the $s-t$ shortest path problem is NP-complete.*

Proof. By Observation 3 we know that the problem is in NP.

The proof of NP-hardness is a reduction from 3-SAT, and is an adaption of the proof given in [GMO76]. An instance of 3-SAT consists of n boolean variables $X_1, \dots, X_n$, and m clauses. Every clause $\mathcal{C}_j$ contains exactly 3 literals, where a literal is either a variable or its negation. The goal is to find a boolean assignment to the variables which satisfies each clause. A clause is satisfied if it at least one of its literal is True.

Given this 3-SAT instance we construct an instance to the shortest path problem with uncertain edge weights, see Figure 5. There will be a vertex for every literal, and for each occurrence of a literal in a clause. In addition we have the vertices s, u, v, t. The graph has several layers. Layer 0 contains vertex s. Layer $j = 1, \dots, m$ contains 3 vertices u_{1j}, u_{2j}, u_{3j} corresponding to the 3 literals in clause C_j. Layer $m+1$ contains vertex u, layer $m+2$ vertex

v. Then for every $i = 1, \ldots, n$, layer $m + 2 + i$ contains 2 vertices v_{0i} and v_{1i}, corresponding respectively to $\overline{X_i}$ and X_i. Finally layer $m + 3 + n$ contains vertex t. There are two type of edges. Solid edges have lower weight 0 and upper weight 1, and connect all pairs of vertices between two adjacent layers. The other type of edges are dashed and have weight 0 (same lower and upper bound). There is a dashed edge between vertices u_{kj} and v_{bi} if and only if the literal corresponding to u_{kj} is the negation of the literal corresponding to v_{bi}.

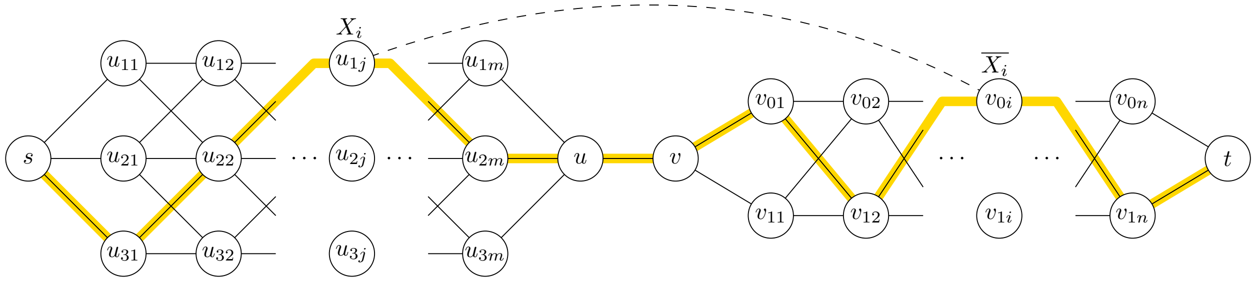

FIGURE 5. Reduction from 3-SAT. Solid edges have uncertainty intervals $[0, 1]$, while dashed edges trivial uncertainty intervals 0.

Now we claim that the 3-SAT formula is satisfiable if and only if the threshold of inclusion T^+_{uv} of the edge uv is at least -1.

For this purpose we introduce the notion of consistency of an $s-t$ path P going through edge uv. We say that P is consistent if it traverses at most one of both endpoints of every dashed edge, and if it traverses every layer exactly once. We can associate a satisfying assignment to every consistent path. Formally, let P be a consistent path. For every $i = 1, \ldots, n$ it traverses exactly one of the vertices v_{0i}, v_{1i}. If it traverses v_{bi}, then we set $X_i = 1 - b$. Now each layer $j = 1, \ldots, m$ is traversed by P in at least one vertex, say u_{kj}, and by consistency this vertex corresponds to a true valued literal.

For one direction of the proof, suppose that the formula is satisfiable. Let P be a consistent $s - t$ path going through edge uv, corresponding to a satisfying assignment. This path has weight 0. Now we consider the shortest $s - t$ path Q not using edge uv. This path has to use at least of the dashed edges, and by consistency of P has to use also at least one solid edge not in P. Hence the weight of Q is at least 1.

The other direction of the proof is straightforward. If there is no $s - t$ path P using edge uv which can enforce weight at least 1, then there is no satisfying assignment. □

Observation 21. *We actually prove that it is NP-hard to distinguish whether the threshold of inclusion is* 0 *or at most* -1. *Combining this with Lemma 11, we obtain that is hard to decide whether a set is an admissible query. As a consequence, we get hardness of approximation. Also, this gap can be made as large a constant as we want by simply scaling the weights.*

For the threshold of exclusion the reduction is much simpler.

Theorem 22. *Computing the threshold of exclusion of an edge for the $s - t$ shortest path problem is NP-complete.*

Proof. The proof is a reduction from the Hamiltonian path problem. Let $H = (V, E)$ be a graph with two vertices $s, t \in V$. The Hamiltonian path problem asks for the existence of a path P from s to v, which visits all vertices of V exactly once. This problem is NP-complete.

Denote by n the number of vertices in H. We construct a graph G from H. All edges in H, which are now also in G, have lower weight $\ell_e = 1$ and upper weight $h_e = n$. We complete the construction by adding two vertices u, v, and edges su, uv, vt, all with zero weight (matching lower and upper weight limit).

Now we claim that such that the threshold of inclusion in G for edge uv is at least $n-1$, if and only if H admits a Hamiltonian path. The shortest $s-t$ path with edge uv has length 0. The shortest $s-t$ path without edge uv has a total length which depends on the edge weights. It can have total length $n-1$ if and only if the set of edges having their weight at the lower limit 1 forms a Hamiltonian path from s to t. This concludes the proof. □

6. Minimum cost perfect matching

We complement the previous NP-hardness proof by showing hardness of computing the thresholds for matching.

Theorem 23. *Computing the threshold of exclusion of an edge in the min cost perfect matching problem is NP-complete already for bipartite graphs.*

Proof. We make a reduction from 3-SAT, similar as in Theorem 20, see Figure 6.

For every variable X_i, there is a *variable gadget* consisting of a complete bipartite graph, between vertices p_i, n_i on one side and vertices s_i, v_i on the other side. Vertex p_i corresponds to the variable X_i and n_i to its negation $\bar{X}_i$.

For every clause $\mathcal{C}_j$ there is a *clause gadget* in form of a bipartite graph between vertices a_j, b_j, c_j on one side and vertices c_{1j}, c_{2j}, c_{3j} on the other side. Vertex c_{kj} corresponds to the k-th literal in $\mathcal{C}_j$.

All edges inside these gadgets have uncertainty intervals of $[0,1]$. In addition, there is a single edge between two new vertices u, v. The construction is completed with additional edges of trivial uncertainty interval 0 (same lower and upper bound), which are detailed below. There is an edge from v to every vertex v_i, and from u to every vertex c_j. All vertices c_{kj} are connected to the corresponding literal vertex. Formally if the k-th literal in $\mathcal{C}_j$ is X_i, then c_{kj} is connected to p_i. And if this literal is $\bar{X}_i$, then c_{kj} is connected to n_i.

We claim that the threshold of exclusion of edge uv is 1 if the formula is satisfiable and 0 if it is not.

Consider some edge weights w, and let M^+ be a min cost perfect matching containing edge uv and M^- a min cost perfect matching not containing edge uv. Since u is matched with v in M^+, in every clause gadget, the vertex c_j is matched with some vertex c_{kj}, and the vertices a_j, b_j are matched to the other two vertices. Similarly, all vertices inside the variable gadgets are matched inside their gadget. These matching edges define a boolean value assignment for the variables, in the sense that $X_i =$ True if s_i is matched to p_i, and $X_i =$ False if s_i is matched to n_i. Also, it selects for every clause one of its literal, by the vertex to which each c_j is matched to. We say that M^+ is *satisfying*, if it selects in every clause a literal assigned to true.

The threshold of exclusion is the difference of the costs $M^- - M^+$, maximized over the edge weights w. Without loss of generality we can assume that $w_e = 0$ for all edges in $e \in M^+$, and $w_e = 1$ for all other edges $e \notin M^+$ and which are inside the gadgets. Remember, edges that are not inside gadgets have weight 0; i.e., a matching lower and upper bound of zero.

We observe that M^- has to match u to some c_j and v to some v_i. Also, these have to be the only vertices of the type $c_{j'}$ and $v_{i'}$ which are matched outside of their gadgets. Since

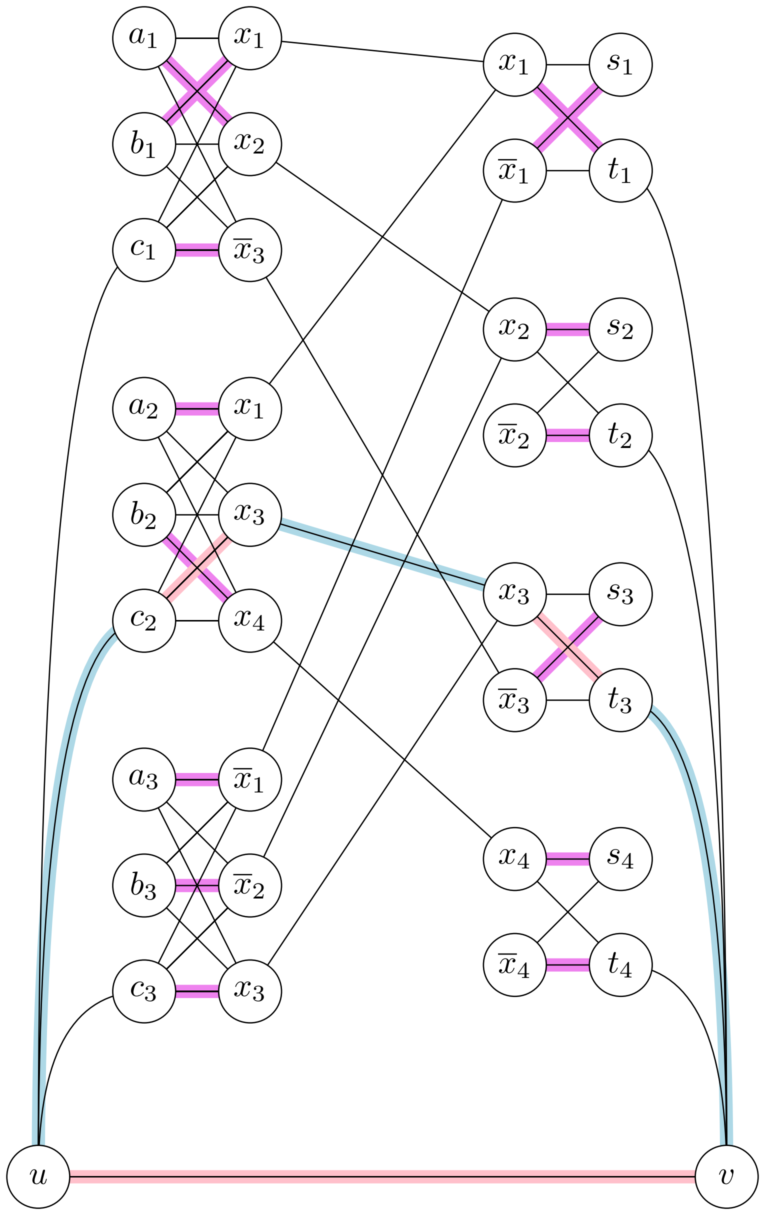

FIGURE 6. The graph obtained by the reduction from the 3-SAT formula consisting of the clauses $\mathcal{C}_1 = X_1 \vee X_2 \vee \overline{X_3}, \mathcal{C}_2 = X_1 \vee X_3 \vee X_4, \mathcal{C}_3 = \overline{X_1} \vee \overline{X_2} \vee X_3$. Edges in $M \setminus M'$ are represented by red lines, edges in $M' \setminus M$ by blue lines, and edges in $M \cap M'$ in purple lines. For convenience we replaced vertices of the form c_{kj}, n_i, p_i by their corresponding literal.

M^- is min cost, by the choice of w, its matching has to coincide with M^+ inside every clause gadget but the j-th and inside every variable gadget but the i-th.

Combining all those observations shows that M^- has zero cost if and only if in M^+ one clause selected a literal that it assigned to False. Say for example that M^+ matches c_j to a vertex c_{kj} corresponding to a literal X_i, and matches s_i to n_i. Then M^- can match a_j, b_j, s_i as in M^+ all at cost 0. However if M^+ is satisfying, the cost of M^- is 1. Indeed it can match

a_j, b_j is in M^+. Assume (c_{kj}, p_i) is matched in M^-. The case (c_{kj}, n_i) is similar. Then s_i must be matched to n_i. However, since M^+ is satisfying by assumption, the cost of this edge is 1. This concludes the proof. □

Observation 24. *We actually prove that it is NP-hard to distinguish whether the threshold of exclusion is* 1 *or* 0*. Combining this with Lemma 11 we obtain that it is NP-hard to decide whether a set is an admissible query. Furthermore, this gap can be made as large a constant as we want by simply scaling the weights.*

The NP-hardness of computing the threshold of inclusion can be shown with a similar construction. Just replace the edge (u, v) by the path $u - u' - v' - v$, where edges (u, u') and (v', v) have both lower and upper weight zero. Now every perfect matching with edge (u, v) in the original graph corresponds to a perfect matching without edge (u', v') in the new graph and vice-versa. Hence computing the threshold of inclusion of edge (u', v') is equivalent to computing the threshold of exclusion of the edge (u, v) in the original graph.

7. Max weight matching in trees

We contrast the previous hardness result with a special matching problem for which the thresholds can be computed efficiently.

Theorem 25. *The thresholds of inclusion/exclusion can be computed in linear time for a fixed edge e for the maximum weight matching problem on trees.*

Proof. We only describe how to compute the threshold of inclusion, as computing the threshold of exclusion follows the same approach.

Fix an edge e in tree. The edge separates the tree into say a *left* and a *right* subtree. By T_u we denote the subtree rooted in some vertex u, where u is not necessarily an endpoint of e. The tree T_u consists of all vertices v, such that the path from v to e contains vertex u.

The threshold of inclusion of edge e is the difference $w(M^-) - w(M^+)$ minimized over weight realizations w, with $w(e) = 0$. We can assume without loss of generality $w(e') = h(e')$ for all $e' \in M^+ \setminus \{e\}$ and $w(e') = \ell(e')$ for all edges $e' \notin M^+$.

For an arbitrary weight realization w, let M^+ be a matching containing edge e and maximizing total weight. Similarly let M^- be a matching not containing edge e and maximizing total weight. By breaking ties consistently between M^+ and M^- we obtain that the symmetric difference $M^+ \Delta M^-$ consists of a single path P containing edge e. This path P connects two vertices s, t where s is in the left subtree and t in the right subtree. By *neighboring edges* of P we mean edges intersecting P at exactly one endpoint. The other endpoint is called a *neighboring vertex* of P.

The vertices s and t determine in a unique manner the matchings M^+ and M^-. Here is how: First the edges along the $s - t$ path belong alternatively to M^+ and to M^-. The alternation is defined with respect to the distance from the edge e in this path. Namely edges e' at odd distance to e belong to M^+ and have weight $h(e')$, while edges at even distance to e belong to M^- and have weight $\ell(e')$. Neighboring edges of the $s - t$ path have low weight and do not belong to M^+ nor M^-. Finally for every neighboring vertex u, we consider a maximum weight matching O_u in T_u were edges in O_u have high weight and edges in $T_u \setminus O_u$ have low weight. This sounds as an intricate definition, because the matching depends on the weight and vice versa, but both can be computing by bottom up dynamic programming.

Since M^+ and M^- coincide outside of the $s-t$ path, the intersection of T_u with M^+ is exactly O_u, and so is the intersection of T_u with M^-.

This observation leads to a simple polynomial time procedure for computing the threshold of inclusion for e. Simply loop over all vertices s in the left subtree and over all vertices t in the right subtree. For each $s-t$ path compute the matchings M^+ and M^- as described in the previous paragraph, and set the weights of all edges in M^+ to their highest value and the weights of all other edges to their lowest value. At this point we can check if M^+ (respectively M^-) is indeed a maximum weight matching under the condition that it contains e (resp. does not contain e). In this case we call the $s-t$ path a *valid* path. The minimum difference $w(M^-) - w(M^+)$ over all valid $s-t$ paths is the threshold of inclusion of e.

However it is possible to compute the threshold in linear time using dynamical programming, as it is often the case for problems defined on trees.

It is well known from matching theory, that matching M^+ has maximum weight, if there is no augmenting path with respect to M^+. Such a path does not contain edge e, by the requirement on e. Also such a path must intersect the $s-t$ path by construction of M^+ which is *locally optimal* outside of the $s-t$ path. Hence we can verify the validity of the $s-t$ path independently on its portions in the left subtree and in the right subtree. The same observation holds for M^-. In that sense, an $s-t$ path is valid of the portion from s to e is valid and the portion from e to t is valid. These properties are independent. If they hold we say that s and t are *valid vertices*. Now we describe how to enumerate in linear time all valid vertices s in one of the subtrees.

For two vertices u, v, such that v lays on the path from u to e, we define A_{uv} to be the following alternating edge weight sum over the $u-v$ path.

- If e' is at even distance from e and at even distance from u, it counts with weight $-\ell(e')$.
- If e' is at even distance from e and at odd distance from u, it counts with weight $+\ell(e')$.
- If e' is at odd distance from e and at even distance from u, it counts with weight $-h(e')$.
- If e' is at odd distance from e and at odd distance from u, it counts with weight $+h(e')$.

These quantities become handy when stating inequalities about paths alternating with respect to M^+ or M^-. Note that if u, v are at odd distance we have $A_{uv} = A_{vu}$ while if they are at even distance we have $A_{uv} = -A_{vu}$.

First we focus on O_u, the maximum weight matching in the subtree rooted at vertex u. In addition we define O_u^- the weight of the maximum weight matching in this subtree, with the restriction that u is unmatched. It could be that later in the whole tree u is matched, but not inside the subtree.

At some moment we need to consider the tree $T_u \setminus T_v$ for some direct descendant v of u. We denote by $O_{u\setminus v}^-$ the maximum weight matching in $T_u \setminus T_v$, with the restriction that u is unmatched. It is simply the sum of $O_{v'}$ over all direct descendants v' of u but different from v. These values can be computed in linear time using dynamic programming, in leaf to root order of the tree. For example we have $O_u = \max\{O_u^-, \max_v h(uv) + O_v^- - O_v\}$.

In addition we define $B_{u\setminus v}$ to be the maximum weight of an alternating $u - v'$ path in $T_u \setminus T_v$ maximized over v'. Edges e' have weight $-h(e')$ if they belong to the maximum weight matching in $T_u \setminus T_v$ and have weight $+\ell(e')$ otherwise.

Before describing the actual algorithm, let's better understand what it means that s is valid. Consider the path P from s to e, and consider a path Q intersection P in a single portion, namely between two vertices u and v. To fix the notation, assume that v is closer to e than u. If Q is meant to be alternating with respect to M^+ or M^-, then u and v must be at odd distance. Let v' be the neighbor of v which is closes to u. Q is not an augmenting path if the following condition hold. It depends on two cases.

- If $u = s$, we must have $B_s + A_{sv} + B_{v\setminus v'} \leqslant 0$.
- If $u \neq s$, let u' the neighbor of u which is closes to s. Then we must have $B_{u\setminus u'} + A_{uv} + B_{v\setminus v'} \leqslant 0$.

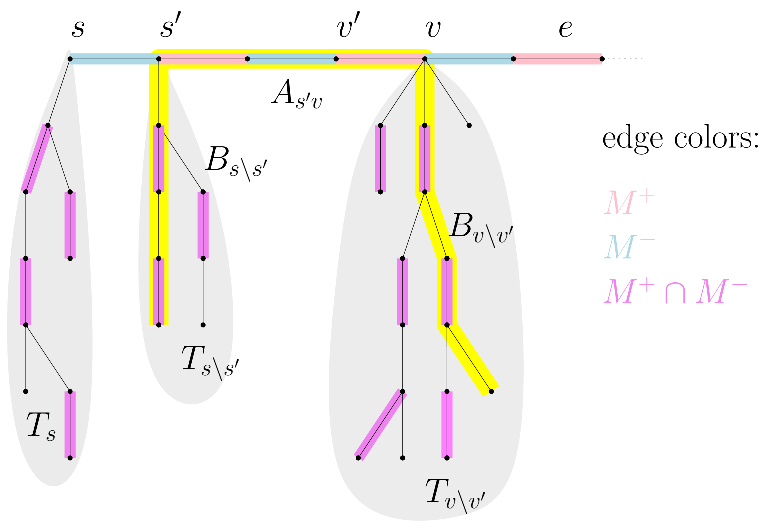

FIGURE 7. Technical aspects of verifying the validity of a path

While we are exploring in depth first manner the left subtree, we verify these conditions along the way in the following manner. Let s be the current vertex of the tree exploration, and s' its ancestor.

If $B_{s'\setminus s} + C_{s'} > 0$ we abort the exploration of this edge (s', s) as no path containing it is valid. Otherwise we continue the tree exploration, but before mark s as being a valid vertex in case if $B_s + C_s > 0$.

Let uu' be the endpoints of e. Let v be a valid vertex which minimizes A_{vu} and v' be a valid vertex which minimizes $A_{v'u'}$. The threshold of inclusion of e is precisely $A_{vu} + A_{v'u'}$. □

8. OPEN QUESTIONS

We finish the paper by pointing out some open questions.

Further exploration of the non-adaptive explorable uncertain setting seems an interesting direction. In particular, to better identify the boundary between polynomial time tractability and NP-hardness of computing thresholds and finding minimum-cost admissible queries. Interesting settings not covered in the paper are: Matching in planar graphs, and computing the edit distance between two strings under uncertain edit costs. Another intriguing question is whether the tractability of the threshold of inclusion is related to the one of the threshold of exclusion; in the sense that both can be solved in polynomial time or none.

Outside of the setting we study in the paper, it would be interesting to study models that interpolate between being fully adaptive and no adaptivity at all; e.g., models with a fixed number of rounds in which to perform the queries.

9. Acknowledgments

We are very grateful to Shahin Kamali, Marcos Kiwi and Claire Mathieu for helpful discussions.

This work is partially supported by the grants ANR-19-CE48-0016 and ANR-23-CE48-0010 from the French National Research Agency, the Proyecto Fondo Basal FB210005 CMM and Fondecyt 1221460 and 1231669, ANID.

References

[Hu61] T. C. Hu. "The Maximum Capacity Route Problem". In: *Operations Research* 9.6 (1961), pp. 898–900.

[Chv75] V. Chvátal. "On certain polytopes associated with graphs". In: *J. Combinatorial Theory Ser. B* 18 (1975), pp. 138–154. DOI: 10.1016/0095-8956(75)90041-6.

[GMO76] H.N. Gabow, S.N. Maheshwari, and L.J. Osterweil. "On Two Problems in the Generation of Program Test Paths". In: *IEEE Transactions on Software Engineering* SE-2.3 (Sept. 1976), pp. 227–231. DOI: 10.1109/TSE.1976.233819.

[GS78] G. Gallo and C. Sodini. "Extreme points and adjacency relationship in the flow polytope". In: *Calcolo* 15.3 (1978), pp. 277–288. DOI: 10.1007/BF02575918.

[Tar79] Robert Endre Tarjan. "Applications of path compression on balanced trees". In: *J. Assoc. Comput. Mach.* 26.4 (1979), pp. 690–715. DOI: 10.1145/322154.322161.

[Kah91] Simon Kahan. "A model for data in motion". In: *Proceedings of the twenty-third annual ACM symposium on Theory of computing - STOC '91*. the twenty-third annual ACM symposium. New Orleans, Louisiana, United States: ACM Press, 1991, pp. 265–277. DOI: 10.1145/103418.103449.

[DRT92] Brandon Dixon, Monika Rauch, and Robert E. Tarjan. "Verification and sensitivity analysis of minimum spanning trees in linear time". In: *SIAM J. Comput.* 21.6 (1992), pp. 1184–1192. DOI: 10.1137/0221070.

[Kin97] V. King. "A simpler minimum spanning tree verification algorithm". In: *Algorithmica* 18.2 (1997), pp. 263–270. DOI: 10.1007/BF02526037.

[Cha00] Bernard Chazelle. "A minimum spanning tree algorithm with inverse-Ackermann type complexity". In: *Journal of the ACM* 47.6 (2000), pp. 1028–1047. DOI: 10.1145/355541.355562.

[Fed+00] Tomas Feder, Rajeev Motwani, Rina Panigrahy, Chris Olston, and Jennifer Widom. "Computing the median with uncertainty". In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. ACM, 2000, pp. 602–607. DOI: 10.1145/335305.335386.

[Bru+05] Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. "Efficient update strategies for geometric computing with uncertainty". In: *Theory of Computing Systems* 38.4 (2005). Publisher: Springer, pp. 411–423. DOI: 10.1007/s00224-004-1180-4.

[Fed+07] Tomás Feder, Rajeev Motwani, Liadan O'Callaghan, Chris Olston, and Rina Panigrahy. "Computing shortest paths with uncertainty". In: *Journal of Algorithms* 62 (Jan. 31, 2007), pp. 1–18. DOI: `10.1016/j.jalgor.2004.07.005`.

[Erl+08] Thomas Erlebach, Michael Hoffmann, Danny Krizanc, Matúš Mihalák, and Rajeev Raman. "Computing Minimum Spanning Trees with Uncertainty". In: *25th International Symposium on Theoretical Aspects of Computer Science*. IBFI Schloss Dagstuhl, 2008, pp. 277–288. DOI: `10.4230/LIPIcs.STACS.2008.1358`.

[Hag10] Torben Hagerup. "An even simpler linear-time algorithm for verifying minimum spanning trees". In: *Graph-theoretic concepts in computer science*. Vol. 5911. Lecture Notes in Comput. Sci. Springer, Berlin, 2010, pp. 178–189. DOI: `10.1007/978-3-642-11409-0_16`.

[EH15] Thomas Erlebach and Michael Hoffmann. "Query-competitive algorithms for computing with uncertainty". In: *Bulletin of EATCS* 2.116 (2015).

[Ta-15] Noam Ta-Shma. "A simple proof of the Isolation Lemma". In: *Electron. Colloquium Comput. Complex.* (2015).

[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). Publisher: SIAM, pp. 1217–1240. DOI: `10.1137/16M1088375`.

[Mei18] Julie Meißner. "Uncertainty exploration (algorithms, competitive analysis, and computational experiments)". PhD thesis. TU Berlin, Germany, 2018. DOI: `10.14279/DEPOSITONCE-7327`.

[Sin18] Sahil Singla. "The price of information in combinatorial optimization". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2018, pp. 2523–2532. DOI: `10.1137/1.9781611975031.161`.

[LMS19] Retsef Levi, Thomas Magnanti, and Yaron Shaposhnik. "Scheduling with testing". In: *Management Science* 65.2 (2019), pp. 776–793. DOI: `10.1287/mnsc.2017.2973`.

[MS19] Arturo I. Merino and José A. Soto. "The Minimum Cost Query Problem on Matroids with Uncertainty Areas". In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Ed. by Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi. Vol. 132. Leibniz International Proceedings in Informatics (LIPIcs). ISSN: 1868-8969. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 83:1–83:14. DOI: `10.4230/LIPIcs.ICALP.2019.83`.

[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`.

[AE21] Susanne Albers and Alexander Eckl. "Explorable uncertainty in scheduling with non-uniform testing times". In: *Approximation and Online Algorithms: 18th International Workshop, WAOA 2020, Virtual Event, September 9–10, 2020, Revised Selected Papers 18*. Springer. 2021, pp. 127–142. DOI: `10.1007/978-3-030-80879-2_9`.

[Duf+22] Fanny Dufossé, Christoph Dürr, Noël Nadal, Denis Trystram, and Óscar C. Vásquez. "Scheduling with a processing time oracle". In: *Applied Mathematical Modelling* 104 (2022), pp. 701–720. DOI: `10.1016/j.apm.2021.12.020`.

[MS23] Nicole Megow and Jens Schlöter. “Set Selection Under Explorable Stochastic Uncertainty via Covering Techniques”. In: *Integer Programming and Combinatorial Optimization*. Ed. by Alberto Del Pia and Volker Kaibel. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2023, pp. 319–333. DOI: 10.1007/978-3-031-32726-1_23.

[Zha23] Xiao Zhang. “Scheduling with explorable uncertainty for minimising the total weighted completion time”. MA thesis. Utrecht University, Department of Information and Computing Sciences, 2023.

[Tho] Mikkel Thorup. “Near-optimal fully-dynamic graph connectivity”. In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. STOC00: The 32nd Annual ACM Symposium on Theory of Computing. Portland Oregon USA: ACM, pp. 343–350. DOI: 10.1145/335305.335345.

(C. Dürr) Sorbonne Université, CNRS, Laboratoire d’informatique de Paris 6, LIP6, Paris, France. Part of the work was done while C.D. was affiliated with Universidad de Chile, CNRS, CMM, Santiago, Chile.

(A. Merino) Saarland University, Computer Science department, Saarbrücken, Germany

(J. A. Soto) Department of Mathematical Engineering, Universidad de Chile, Chile

(J. Verschae) Pontificia Universidad Católica, Institute for Mathematical and Computational Engineering, Faculty of Mathematics and School of Engineering, Chile