Randomization Helps Computing a Minimum Spanning Tree under Uncertainty*

Nicole Megow[1], Julie Meißner[2], and Martin Skutella[2]

[1] Center for Mathematics, Technische Universität München, Germany.
Email: nicole.megow@tum.de
[2] Department of Mathematics, Technische Universität Berlin, Germany.
Email: {jmeiss,skutella}@math.tu-berlin.de

Abstract. We consider the problem of finding a minimum spanning tree (MST) in a graph with uncertain edge weights given by open intervals on the edges. The exact weight of an edge in the corresponding uncertainty interval can be queried at a given cost. The task is to determine a possibly adaptive query sequence of minimum total cost for finding an MST. For uniform query cost, a deterministic algorithm with best possible competitive ratio 2 is known [7].
We solve a long-standing open problem by showing that randomized query strategies can beat the best possible competitive ratio 2 of deterministic algorithms. Our randomized algorithm achieves expected competitive ratio $1 + 1/\sqrt{2} \approx 1.707$. This result is based on novel structural insights to the problem enabling an interpretation as a generalized online bipartite vertex cover problem. We also consider arbitrary, edge-individual query costs and show how to obtain algorithms matching the best known competitive ratios for uniform query cost. Moreover, we give an optimal algorithm for the related problem of computing the exact weight of an MST at minimum query cost. This algorithm is based on an interesting relation between different algorithmic approaches using the cycle-property and the cut-property characterizing MSTs. Finally, we argue that all our results also hold for the more general setting of matroids. All our algorithms run in polynomial time.

1 Introduction

Uncertainty in the input data is an omnipresent issue in most real world planning processes. The quality of solutions for optimization problems with uncertain input data crucially depends on the amount of uncertainty. More information, or even knowing the exact data, allows for significantly improved solutions (see, e. g., [16]). It is impossible to fully avoid uncertainty. Nevertheless, it is sometimes possible to obtain exact data, but it may involve certain exploration cost in time, money, energy, bandwidth, etc. A classical application are estimated user

* This research was carried out in the framework of MATHEON supported by Einstein Foundation Berlin. The first two authors were additionally supported by the German Science Foundation (DFG) under contract ME 3825/1.

demands that can be specified by undertaking a user survey, but this is an investment in terms of time and/or cost.

In this paper we are concerned with fundamental combinatorial optimization problems with uncertain input data which can be explored at certain cost. We mainly focus on the minimum spanning tree (MST) problem with uncertain edge weights. In a given graph, we know initially for each edge only an interval containing the edge weight. The true value is revealed upon request (we say query) at a given cost. The task is to determine a minimum-cost adaptive sequence of queries to find a minimum spanning tree. In the basic setting, we only need to guarantee that the obtained spanning tree is minimal and do not need to compute its actual weight, i. e., there might be tree edges whose weight we never query, as they appear in an MST independent of their exact weight. We measure the performance of an algorithm by competitive analysis. For any realization of edge weights, we compare the query cost of an algorithm with the optimal query cost. This is the cost for verifying an MST for a given fixed realization.

As our main result we develop a randomized algorithm that improves upon the competitive ratio of any deterministic algorithm. This solves an important open problem in this area (cf. [4]). We also present the first algorithms for non-uniform query costs and generalize the results to uncertainty matroids, in both settings matching the best known competitive ratios for MST with uniform query cost.

Related work. The huge variety of research streams dealing with optimization under uncertainty reflects its importance for theory and practice. The major fields are online optimization [3], stochastic optimization [2], and robust optimization [1], each modeling uncertain information in a different way. Typically these models do not provide the possibility to influence when and how uncertain data is revealed. Kahan [12] was probably the first to study algorithms for explicitly exploring uncertain information in the context of finding the maximum and median of a set of values known to lie in given uncertainty intervals.

The MST problem with uncertain edge weights was introduced by Erlebach et al. [7]. Their deterministic Algorithm U-RED achieves competitive ratio 2 for uniform query cost when all uncertainty intervals are open intervals or trivial (i. e., containing one point only). They also show that this ratio is optimal and can be generalized to the problem of finding a minimum weight basis of a matroid with uncertain weights [6]. According to Erlebach [4] it remained a major open problem whether randomized algorithms can beat competitive ratio 2. The offline problem of finding the optimal query set for given exact edge weights can be solved optimally in polynomial time [5].

Further problems studied in this uncertainty model include finding the k-th smallest value in a set of uncertainty intervals [9, 11, 12] (also with non-uniform query cost [9]), caching problems in distributed databases [15], computing a function value [13], and classical combinatorial optimization problems, such as shortest path [8], finding the median [9], and the knapsack problem [10].

A generalized exploration model was proposed in [11]. The OP-OP model reveals, upon an edge query, a refined open or trivial subinterval and might, thus, require multiple queries per edge. They show that Algorithm U-RED [7] can be

adopted and still achieves competitive ratio 2. The restriction to open intervals is not crucial as slight model adaptions allow to deal with closed intervals [11, 12].

While most works aim for minimal query sets to guarantee exact optimal solutions, Olsten and Widom [15] initiate the study of trade-offs between the number of queries and the precision of the found solution. They are concerned with caching problems. Further work in this vein can be found in [8, 9, 13].

Our contribution. After presenting some structural insights in Section 2, we affirmatively answer the question if randomization helps to minimize query cost in order to find an MST. In Section 3 we present a randomized algorithm with tight competitive ratio 1.707, thus beating the best possible competitive ratio 2 of any deterministic algorithm. On the other hand, one can easily achieve a lower bound of 1.5 for any randomized algorithm by considering a graph consisting of two parallel edges with crossing uncertainty intervals, e.g., $(1, 3)$ and $(2, 4)$.

One key observation is that the minimum spanning tree problem under uncertainty can be interpreted as a generalized online bipartite vertex cover problem. A similar connection for a *given* realization of edge weights was established in [5] for the related MST verification problem. In our case, new structural insights allow for a preprocessing which suggests a unique bipartition of the edges for *all* realizations simultaneously. Our algorithm borrows and refines ideas from a recent water-filling algorithm for the online bipartite vertex cover problem [17].

In Section 4 we consider the more general non-uniform query cost model in which each edge has an individual query cost. We observe that this problem can be reformulated within a different uncertainty model, called OP-OP, presented in [11]. The 2-competitive algorithm in [11] is a pseudo-polynomial 2-competitive algorithm for our problem with non-uniform query cost. We design new direct and polynomial-time algorithms that are 2-competitive and 1.707-competitive in expectation. To that end, we employ a new strategy carefully balancing the query cost of an edge and the number of cycles it occurs in.

In Section 5 we consider the problem of computing the exact value of an MST under uncertain edge weights. While previous algorithms (U-RED [7], our algorithms) aim for removing the largest-weight edge from a cycle, we now attempt to detect minimum-weight edges separating the graph into two components. Interestingly, the latter cut-based algorithm can be shown to solve the original problem (not computing the exact MST value) achieving the same best possible competitive ratio of 2 as the cycle-based algorithm presented in [7].

Finally, in Section 6 we observe in a broader context that these two algorithms can be interpreted as the best-in and worst-out greedy algorithm on matroids.

Due to space constraints some details are omitted and will be presented in a full version.

2 Problem Definition, Notation and Structural Insights

Problem description. Initially we are given a weighted, undirected, connected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$. Each edge $e \in E$ comes with an uncertainty interval A_e and possibly a query cost c_e. The uncertainty interval A_e

gives the only information about e's unknown weight $w_e \in A_e$. We assume that an interval is either trivial, i.e., $A_e = [w_e, w_e]$, or, it is open $A_e = (L_e, U_e)$ with lower limit L_e and upper limit $U_e > L_e$. A realization of edge weights $(w_e)_{e \in E}$ for an uncertainty graph G is *feasible*, if all edge weights w_e, $e \in E$, lie in their corresponding uncertainty intervals, i.e., $w_e \in A_e$.

The task is to find a minimum spanning tree (MST) in the uncertainty graph G for an unknown, feasible realization of edge weights. To that end, we may query any edge $e \in E$ at cost c_e and obtain the exact weight w_e. The goal is to design an algorithm that constructs a sequence of queries that determines an MST at minimum total query cost. A set of queries $Q \subseteq E$ is *feasible*, if an MST can be determined given the exact edge weights for edges in Q; that is, given w_e for $e \in Q$, there is a spanning tree which is minimal for any realization of edge weights $w_e \in A_e$ for $e \in E \setminus Q$. We denote this problem as *MST with edge uncertainty* and say *MST under uncertainty* for short.

Note that this problem does not necessarily involve computing the actual MST weight. We refer to the problem variant in which the actual MST weight must be computed as *computing the MST weight under uncertainty*.

We evaluate our algorithms by standard competitive analysis. An algorithm is α-*competitive* if, for any realization $(w_e)_{e \in E}$, the solution query cost is at most α times the optimal query cost for this realization. The optimal query cost is the minimum query cost that an offline algorithm must pay when it is given the realization (and thus an MST) and has to verify an MST. The *competitive ratio* of an algorithm ALG is the infimum over all α such that ALG is α-competitive. For randomized algorithms we consider the expected query cost. Competitive analysis addresses the problem complexity evolving from the uncertainty in the input, possibly neglecting any computational complexity. However, we note that all our algorithms run in polynomial time unless explicitly stated otherwise.

Structural insight. We derive a structural property that allows to reduce MST under uncertainty to a set of crucial instances. Given an uncertainty graph $G = (V, E)$, consider the following two MSTs for extreme realizations. Let $T_L \subseteq E$ be an MST for the realization w^L, in which all edge weights of edges with non-trivial uncertainty interval are close to their lower limit, more precisely $w_e = L_e + \varepsilon$ for infinitesimally small $\varepsilon > 0$. Symmetrically, let $T_U \subseteq E$ be an MST when the same edges have weight $w_e = U_e - \varepsilon$.

Theorem 1. *Given an uncertainty graph with trees T_L and T_U, any edge $e \in T_L \setminus T_U$ with $L_e \neq U_e$ is in every feasible query set for any feasible realization.*

Proof. Given an uncertainty graph, let h be an edge in $T_L \setminus T_U$ with non-trivial uncertainty interval. Assume all edges apart from h have been queried and thus have fixed weight w_e. As edge h is in T_L, we can choose its edge weight such that edge h is in any MST. We set $w_h = L_h + \varepsilon$ and choose ε so small, that all edges with at least the same weight in w^L now have a strictly larger edge weight. Symmetrically, if we choose the edge weight w_h sufficiently close to the upper limit U_h, no MST contains edge h. Consequently we cannot decide whether edge h is in an MST without querying it. □

Any edge in the set $T_L \backslash T_U$ with non-trivial uncertainty area is in every feasible query set and thus can be queried in a preprocessing step. Its existence increases the size of every feasible query set and hence decreases the competitive ratio of an instance. Thus we restrict our analysis to instances for which the set $T_L \backslash T_U$ contains only edges with $L_e = U_e$, to find the worst-case competitive ratio of an algorithm.

Assumption 1. *We restrict to uncertainty graphs for which $e \in T_L \setminus T_U$ implies $L_e = U_e$.*

We call an edge in a cycle *maximal*, if any realization has an MST that does not contain this edge. Symmetrically, an edge in a cut is *minimal* if every realization has an MST containing it. Whenever we sort by increasing lower (decreasing upper) limit, we break ties by preferring the smaller upper (greater lower) limit.

3 A Randomized Algorithm for MST under Uncertainty

We give an algorithm that solves MST under uncertainty with competitive ratio $1 + 1/\sqrt{2} \approx 1.707$ in expectation. As U-RED in [7], our algorithm is based on Kruskal's algorithm [14], that iteratively deletes maximal edges from cycles. We decide how to resolve cycles, by maintaining an edge potential for each edge $e \in T_L$ describing the probability to query it. The edge potentials are increased in every cycle we consider throughout the algorithm. To determine the increase, we carefully adapt a water-filling scheme presented in [17] for online bipartite vertex cover. In this section we assume uniform query cost $c_e = 1$, $e \in E$, and explain the generalization to non-uniform query costs in Section 4.

Our algorithm RANDOM is structured as follows: We maintain a tree Γ which initially is set to T_L and sort all remaining edges in $R = E \setminus T_L$ by increasing lower limit. We choose a query bound $b \in [0, 1]$ uniformly at random. Then we iteratively add edges $f_i \in R$ to Γ, closing a unique cycle C_i in each iteration i. Edges in $C_i \cap T_L$ with uncertainty interval overlapping that of edge f_i compose the *neighbor set* $X(f_i)$. In each cycle we query edges until we identify a maximal edge.

To decide which edge to query in cycle C_i, we consider the potentials y_e of edges $e \in C_i \cap T_L$. In each iteration we evenly increase the potential of all neighbors $X(f_i)$ of edge f_i. We choose a threshold $t(f_i)$ as large as possible, such that when we increase y_e to $\max\{t(f_i), y_e\}$ for all neighbors $e \in X(f_i)$, the total increase in the potential sums up to no more than a fixed parameter α whose optimal value is determined later in the analysis.

Now we compare the edge potentials to the query bound b and decide if we query the edges in $X(f_i)$ or edge f_i. If these queries do not suffice to identify a maximal edge, we repeatedly query the edge with the largest upper limit in the cycle. A formal description of our algorithm is given in Algorithm 1.

RANDOM computes a feasible query set, since it deletes in each cycle a maximal edge. It terminates, as in each iteration of the while loop one edge is queried. When all edges in a cycle have been queried, we always find a maximal edge.

For the analysis of RANDOM we combine the Kruskal-MST structure of our algorithm with Assumption 1. Similar to the analysis in [7] we derive two lemmas.

Algorithm 1 RANDOM

Input: An uncertainty graph $G = (V, E)$ and a parameter $\alpha \in [0, 1]$.
Output: A feasible query set Q.

1: Determine tree T_L, set the temporary graph Γ to T_L, and initialize $Q = \emptyset$.
2: Index the edges in $R = E \backslash T_L$ by increasing lower limit $f_1, \ldots, f_{m-n+1}$.
3: For all edges $e \in T_L$ set the potential y_e to 0.
4: Choose the query bound $b \in [0, 1]$ uniformly at random.
5: **for** $i = 1$ to $m - n + 1$ **do**
6: Add edge f_i to the temporary graph Γ and let C_i be the unique cycle closed.
7: Let $X(f_i)$ be the set of edges $g \in T_L \cap C_i$ with $U_g > L_{f_i}$.
8: **if** no edge in the cycle C_i is maximal **then**
9: Maximize the threshold $t(f_i) \leq 1$ s.t. $\sum_{e \in X(f_i)} \max\{0, t(f_i) - y_e\} \leq \alpha$.
10: Increase the edge potential $y_e = \max\{t(f_i), y_e\}$ for all edges $e \in X(f_i)$.
11: **if** $t(f_i) \leq b$ **then**
12: Add edge f_i to the query set Q and query it.
13: **else**
14: Add all edges in $X(f_i)$ to query set Q and query them.
15: **while** no edge in the cycle C_i is maximal **do**
16: Query the edge $e \in C_i$ with maximum U_e and add it to the query set Q.
17: Delete a maximal edge from Γ.
18: **return** The query set Q.

Lemma 1. *Any feasible query set contains for every cycle-closing edge f_i either edge f_i or its neighborhood $X(f_i)$.*

Lemma 2. *Any edge queried in Line 16 of* RANDOM *is contained in any feasible query set.*

This concludes the preliminaries to prove the algorithm's competitive ratio.

Theorem 2. *For $\alpha = \frac{1}{\sqrt{2}}$,* RANDOM *has competitive ratio $1 + \frac{1}{\sqrt{2}} (\approx 1.707)$.*

Proof. Consider a fixed realization and an optimal query set Q^*. We denote the potential of an edge $e \in T_L$ at the start of iteration i by y_e^i and use y_e to denote the edge potential after the last iteration of the algorithm. The increase of potentials in the algorithm depends on the cycles that are closed and thus on the realization, but not the queried edges. This means the edge potentials are chosen independently of the query bound b in the algorithm. Edges queried in Line 16 are in Q^* by Lemma 2, therefore an edge $e \in T_L \backslash Q^*$ is queried with probability $P(y_e > b) = y_e$ and an edge $f_i \in R \backslash Q^*$ is queried with probability $P(t(f_i) \leq b) = 1 - t(f_i)$. Hence, we can bound the total expected query cost by

$$\mathbb{E}[|Q|] \leq |Q^*| + \sum_{e \in T_L \backslash Q^*} y_e + \sum_{i: f_i \in R \backslash Q^*} (1 - t(f_i)) \,. \tag{1}$$

For any edge $e \in T_L \backslash Q^*$, Lemma 1 states that all edges $f \in R$ with $e \in X(f)$ must be in the optimal query set Q^*. The potential y_e is the sum of the potential

increases caused by edges $f \in R$ with $e \in X(f)$. As in each iteration of the algorithm the total increase of potential is bounded by α, we have

$$\sum_{e \in T_L \setminus Q^*} y_e = \sum_{e \in T_L \setminus Q^*} \sum_{\substack{i: f_i \in R \cap Q^*, \\ e \in X(f_i)}} \max\left\{t(f_i) - y_e^i, 0\right\}$$
$$\leq \sum_{i: f_i \in R \cap Q^*} \sum_{e \in X(f_i)} \max\left\{t(f_i) - y_e^i, 0\right\} \leq \sum_{i: f_i \in R \cap Q^*} \alpha = \alpha \cdot |R \cap Q^*|. \quad (2)$$

For an edge $f_i \in R \setminus Q^*$ with $t(f_i) < 1$ we distribute exactly potential α among its neighbors $X(f_i)$ in Lines 9 and 10 of the algorithm. By Lemma 1, the neighbor set $X(f_i)$ is part of the optimal query set Q^*. We consider the share of the total potential increase each neighbor receives and distribute the term $1 - t(f_i)$ (see (1)) according to these shares. Hence,

$$\sum_{i: f_i \in R \setminus Q^*} (1 - t(f_i)) = \sum_{i: f_i \in R \setminus Q^*} \frac{1 - t(f_i)}{\alpha} \sum_{e \in X(f_i)} \max\{t(f_i) - y_e^i, 0\}$$
$$= \sum_{e \in T_L \cap Q^*} \sum_{\substack{i: f_i \in R \setminus Q^*, \\ e \in X(f_i)}} \frac{1 - t(f_i)}{\alpha} \left(y_e^{i+1} - y_e^i\right). \quad (3)$$

In the last equation we have used $y_e^{i+1} = \max\{t(f_i), y_e^i\}$. We consider the inner sum in (3) and bound the summation term from above by an integral from y_e^i to y_e^{i+1} of the function $\frac{1-z}{\alpha}$. This yields a valid upper bound as the function is decreasing in z and $t(f_i) = y_e^{i+1}$, unless $y_e^{i+1} - y_e^i = 0$. This yields

$$\sum_{\substack{i: f_i \in R \setminus Q^*, \\ e \in X(f_i)}} \frac{1 - t(f_i)}{\alpha} \left(y_e^{i+1} - y_e^i\right) \leq \sum_{\substack{i: f_i \in R \setminus Q^*, \\ e \in X(f_i)}} \int_{y_e^i}^{y_e^{i+1}} \frac{1 - z}{\alpha} dz \leq \int_0^1 \frac{1 - z}{\alpha} dz = \frac{1}{2\alpha}.$$

Now we use this bound in Equation (3) and conclude

$$\sum_{i: f_i \in R \setminus Q^*} (1 - t(f_i)) \leq \frac{1}{2\alpha} \cdot |T_L \cap Q^*|.$$

Plugging this bound and (2) into (1) yields total query cost

$$\mathbb{E}\left[|Q|\right] \leq |Q^*| + \alpha \cdot |R \cap Q^*| + \frac{1}{2\alpha} \cdot |T_L \cap Q^*|.$$

Choosing $\alpha = 1/\sqrt{2}$ gives the desired competitive ratio $1 + 1/\sqrt{2}$ for RANDOM.

A simple example shows that this analysis is tight. Consider two parallel edges f and g with overlapping uncertainty intervals. Let f be the edge with larger upper limit. In RANDOM we distribute potential α to g and potential $1-\alpha$ to edge f. However, the realization with $L_f < w_g < U_g < w_f$ has optimal query set $\{f\}$, while $\{g\}$ is not a feasible query set. The algorithm queries edge g first with probability α and has query cost 2 in this case. Thus the algorithm has expected competitive ratio $1 + \alpha$ for this instance. □

4 Non-uniform Query Cost

Consider the problem MST under uncertainty in which each edge $e \in E$ has associated an individual query cost c_e. W.l.o.g. we assume $c_e > 0$, for all $e \in E$, since querying all other edges only decreases the total query cost. We give a new polynomial-time 2-competitive algorithm, which is deterministically best possible. Furthermore, we adapt our algorithm RANDOM (Sec. 3) to handle non-uniform query costs achieving the same competitive ratio $1 + 1/\sqrt{2}$.

Before showing the main results, we remark that the problem can be transformed into the OP-OP model [11]. This model allows multiple queries per edge and each query returns an open or trivial subinterval. Given an uncertainty graph, we model the query cost $c_e, e \in E$, in the OP-OP model as follows: querying an edge e returns the same interval for $c_e - 1$ queries and then the exact edge weight. The 2-competitive algorithm for the OP-OP model [11] has a running time depending on the query cost of our original problem.

Theorem 3. *There is a pseudo-polynomial 2-competitive algorithm for MST under uncertainty and non-uniform query cost.*

4.1 Balancing Algorithm

Our polynomial-time algorithm BALANCE relies on the property that an MST is cycle-free, similar to previous algorithms for uniform query cost. The key idea is as follows: To decide which edge to query in a cycle, we use a value function $v : E \to \mathbb{R}_{\geq 0}$ that represents for an edge $e \in E$ the cost difference between a local solution containing e and one that does not. Initially we are locally only aware of each edge individually and thus initialize its value at c_e. We design a balancing scheme that queries among two well-chosen edges the one with smaller value and charges the value of the queried edge to the non-queried alternative.

More formally, in BALANCE (cf. Algorithm 2) we choose a tree T_L and iteratively add the other edges in increasing order of lower limit to T_L. In an occurring cycle, we consider an edge f with maximal upper limit and an edge g with overlapping uncertainty interval. We query the edge $e \in \{f, g\}$ with the smaller value $v(e)$ and decrease the value of the non-queried edge by $v(e)$. We repeat this until we identify a maximal edge in the cycle.

BALANCE computes a feasible query set, since it deletes in each cycle a maximal edge. It terminates, as in each iteration of the while loop one edge is deleted or queried. When all edges in a cycle are queried, we always find a maximal edge.

Now we consider a query set computed by BALANCE. For each edge $e \in E$, let the set of its *children* $C(e) \subseteq E$ be the set of edges that decreased $v(e)$ in the algorithm (cf. Line 14 and 16). Furthermore we define recursively the set of *related edges* $S_e \subseteq E$ to be the union of edge e and the sets S_h of all children $h \in C(e)$.

Every edge is the child of at most one edge, because when it contributes to some value in Line 14 or 16, it is queried. Thus we can interpret a set of related edges S_e and its children-relation as a tree. Slightly abusing notation we speak of a *vertex cover* $VC(S_e)$ of the set of related edges S_e and mean a minimum weight

Algorithm 2 BALANCE

Input: An uncertainty Graph $U = (V, E)$ and a query cost function $c : E \to \mathbb{R}_{\geq 0}$.
Output: A feasible query set Q.

```
1: Choose a tree T_L and let the temporary graph Γ = T_L.
2: Index the edges in E\T_L by increasing lower limit e_1, e_2, ..., e_{m-n+1}.
3: Set a value function v : E → R_{≥0} to c_e for all edges.
4: for i = 1 to m − n + 1 do
5:    Add e_i to Γ.
6:    while Γ has a cycle C do
7:       if C contains a maximal edge e then
8:          Delete e from Γ.
9:       else
10:         Choose f ∈ C such that U_f = max{U_e | e ∈ C} and g ∈ C\f with U_g > L_f.
11:         if A_g is trivial then
12:            Query edge f and add f to Q.
13:         else if v(f) ≥ v(g) then
14:            Query edge g, add g to Q, and subtract v(g) from v(f).
15:         else
16:            Query edge f, add f to Q, and subtract v(f) from v(g).
17: return The query set Q.
```

vertex cover in the corresponding tree. We use VC_e for a vertex cover containing edge e and $VC_{\setminus e}$ for one not containing e. Similar to Lemmas 1 and 2 we have:

Lemma 3. *For every feasible query set Q and every set of related edges S_e in* BALANCE*, the set Q contains a vertex cover of S_e.*

Lemma 4. *Every edge queried in Line 12 is in every feasible query set.*

The following two lemmas establish a relation between the value $v(e)$ of an edge $e \in E$ and the cost of its related edges S_e. The proof is by induction.

Lemma 5. *The value function after an execution of* BALANCE *fulfills for every edge $e \in E$ and its set of related edges S_e:*

$$\sum_{f \in VC_e(S_e)} c_f = v(e) + \sum_{f \in VC_{\setminus e}(S_e)} c_f.$$

Lemma 6. *The value function after an execution of* BALANCE *fulfills for every edge $e \in E$ and its set of related edges S_e:*

$$2 \cdot \sum_{f \in VC_{\setminus e}(S_e)} c_f = -v(e) + \sum_{f \in S_e} c_f.$$

This concludes all preliminaries we need to prove the main theorem.

Theorem 4. BALANCE *has a competitive ratio of 2 and this is best possible.*

Proof. For some realization, let Q^* denote an optimal query set and Q a query set computed by Balance. Let A be the set of all edges not in Q and let B be the set of all edges queried in Line 12 of Balance. As any edge in Q is either queried in Line 12, or the child of some other edge, Q is the disjoint union of the sets $S_a \backslash \{a\}, a \in A$, and the sets $S_b, b \in B$.

We bound the cost of a set $S_a \backslash \{a\}, a \in A$, by applying Lemmas 6 and then 5 and concluding from Lemma 3 that Q^* contains a vertex cover of S_a. Hence,

$$\sum_{e \in S_a \backslash \{a\}} c_e \leq 2 \cdot \sum_{e \in VC_{\backslash a}(S_a)} c_e = 2 \cdot \sum_{e \in VC(S_a)} c_e \leq 2 \cdot \sum_{e \in Q^* \cap S_a} c_e.$$

By definition, every edge $b \in B$ is queried in Line 12 and is thus by Lemma 4 an element of Q^*. Applying Lemma 3 this means the cost of $Q^* \cap S_b$ is at least the of cost $VC_b(S_b)$. We use this fact after applying Lemmas 5 and 6 and deduce

$$\sum_{e \in S_b} c_e = v(b) + 2 \cdot \sum_{e \in VC_{\backslash b}(S_b)} c_e \leq 2 \cdot \sum_{e \in VC_b(S_b)} c_e \leq 2 \cdot \sum_{e \in Q^* \cap S_b} c_e.$$

As the set Q is a disjoint union of all sets $S_a \backslash \{a\}, a \in A$, and $S_b, b \in B$, this yields the desired competitive ratio of 2. This factor is best possible for deterministic algorithms, even in the special case of uniform query costs [7]. □

4.2 Randomization for Non-uniform Query Cost

We generalize the algorithm Random (Sec. 3) to the non-uniform query cost model. The adaptation is similar to one for the weighted online bipartite vertex cover problem in [17]. For each edge $f_i \in R$ we distribute at most $1/\alpha \cdot c_{f_i}$ new potential to its neighborhood $X(f_i)$. We replace Line 9 of Random by:

$$\text{Maximize } t(f_i) \leq 1 \text{ s.t. } \sum_{e \in X(f_i)} c_e \max\{t(f_i) - y_e), 0\} \leq \frac{c_{f_i}}{\alpha} \text{ holds.} \qquad (4)$$

Using exactly the same analysis as presented in Section 3 this yields:

Theorem 5. *For the non-uniform query cost setting* Random *adapted by* (4) *achieves expected competitive ratio* $1 + \frac{1}{\sqrt{2}}$.

5 Computing the MST Weight under Uncertainty

In this section we give an optimal polynomial-time algorithm for computing the exact MST weight in an uncertainty graph. As a key to our result, we algorithmically utilize the well-known characterization of MSTs through the *cut property* - in contrast to previous algorithms for the MST under uncertainty problem which relied on the *cycle property* (cf. Random, Balance, and U-RED [7]).

In Cut-Weight, we consider a tree T_U and iteratively delete its edges in decreasing order of upper limits. In each iteration, we consider the cut which

Algorithm 3 CUT-WEIGHT

Input: An uncertainty graph $G = (V, E)$.
Output: A feasible query set Q.

Choose a tree T_U and let the temporary graph $\Gamma = T_U$. Initialize $Q = \emptyset$.
Index all edges of T_U by decreasing upper limit $e_1, e_2, ..., e_{n-1}$.
for $i = 1$ to $n - 1$ **do**
  Delete e_i from Γ.
  while Γ has two components **do**
    Let S be the cut containing all edges in G between the two components of Γ.
    if S contains a minimal edge e **then**
      Query edge e and add it to Q.
      Replace edge e_i in Γ with e and contract edge e.
    else
      Choose $g \in S$ such that $L_g = \min\{L_e | e \in S\}$, query it and add it to Q.
return The query set Q.

is defined in the original graph and query edges in increasing order of lower limits until we identify a minimal edge. Then we exchange the tree edge with the minimal edge and contract it. Applying this procedure, we only query edges that are in any feasible query set.

Theorem 6. CUT-WEIGHT *determines the optimal query set and the exact MST weight in polynomial time.*

It may seem surprising that CUT-WEIGHT solves the problem optimally whereas cycle-based algorithms do not. However, there is an intuition. CUT-WEIGHT identifies a *minimum weight* edge in each cut which characterizes an MST. Informally speaking, it has a bias to query edges of the MST. In contrast, cycle-based algorithms identify *maximum weight* edges in cycles, which are not in the tree.

6 Matroids under Uncertainty

We briefly consider a natural generalization of MST under uncertainty: given an *uncertainty matroid*, i.e., a matroid with a ground set of elements with unknown weights, find a minimum weight matroid base. Erlebach et al. [6] show that the algorithm U-RED [7] can be applied to uncertainty matroids with uniform query cost and yields again a competitive ratio of 2. Similarly, our algorithms RANDOM and BALANCE can be generalized to matroids with non-uniform cost, and CUT-WEIGHT can determine the total weight of a minimum weight matroid base.

Theorem 7. *There are deterministic resp. randomized online algorithms with competitive ratio* 2 *resp.* 1.707 *for finding a minimum weight matroid base in an uncertainty matroid with non-uniform query cost.*

Theorem 8. *There is an algorithm that determines an optimal query set and the exact weight of a min-weight matroid base in an uncertainty matroid.*

In a matroid with known weights we can find a minimum weight base greedily; we distinguish between *best-in* greedy and *worst-out* greedy algorithms (cf. [14]). They are dual in the sense that both solve the problem on a matroid and take the role of the other on the corresponding dual matroid. The best-in greedy algorithm adds elements in increasing order of weights as long as the system stays independent. Merging ideas from our algorithm RANDOM and U-RED2 in [6] yields a best-in greedy algorithm, CYCLE-ALG, for uncertainty matroids. A worst-out greedy algorithm deletes elements in decreasing order of weights as long as a basis is contained. We can adapt our algorithm CUT-WEIGHT in Section 5 to a worst-out greedy algorithm, CUT-ALG, for uncertainty matroids.

Proposition 1. *The algorithms* CYCLE-ALG *and* CUT-ALG *are dual to each other in the sense that they solve the same problem on a matroid and its dual.*

Acknowledgments. We thank the anonymous referees for numerous helpful comments that improved the presentation of the paper.

References

1. A. Ben-Tal, L. El Ghaoui, and A. S. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.
2. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research. Springer, 1997.
3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
4. T. Erlebach. Computing with uncertainty. Invited lecture, Graduate Program MDS, Berlin, 2013.
5. T. Erlebach and M. Hoffmann. Minimum spanning tree verification under uncertainty. In *Proceedings of WG*, pages 164–175, 2014.
6. T. Erlebach, M. Hoffmann, and F. Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. In *Proc. of MFCS*, pages 263–274, 2014.
7. T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihalák, and R. Raman. Computing minimum spanning trees with uncertainty. In *Proc. STACS*, pages 277–288, 2008.
8. T. Feder, R. Motwani, L. O'Callaghan, C. Olston, and R. Panigrahy. Computing shortest paths with uncertainty. *Journal of Algorithms*, 62:1–18, 2007.
9. T. Feder, R. Motwani, R. Panigrahy, C. Olston, and J. Widom. Computing the median with uncertainty. *SIAM Journal on Computing*, 32:538–547, 2003.
10. M. Goerigk, M. Gupta, J. Ide, A. Schöbel, and S. Sen. The robust knapsack problem with queries. *Computers & OR*, 55:12–22, 2015.
11. M. Gupta, Y. Sabharwal, and S. Sen. The update complexity of selection and related problems. In *Proc. of FSTTCS*, volume 13 of *LIPIcs*, pages 325–338, 2011.
12. S. Kahan. A model for data in motion. In *Proc. of STOC*, pages 267–277, 1991.
13. S. Khanna and W. C. Tan. On computing functions with uncertainty. In *Proceedings of PODS*, pages 171–182, 2001.
14. B. Korte and J. Vygen. *Combinatorial optimization*, volume 21. Springer, 2012.
15. C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *Proceedings of VLDB*, pages 144–155, 2000.
16. P. Patil, A. P. Shrotri, and A. R. Dandekar. Management of uncertainty in supply chain. *Int. J. of Emerging Technology and Advanced Engineering*, 2:303–308, 2012.
17. Y. Wang and S. C.-W. Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *Proc. of ICALP*, pages 1070–1081, 2015.