Received: 15 October 2022 | Revised: 13 September 2023 | Accepted: 24 November 2023

DOI: 10.1002/net.22204

SPECIAL ISSUE ARTICLE

WILEY

Special cases of the minimum spanning tree problem under explorable edge and vertex uncertainty

Corinna Mathwieser[1] | **Eranda Çela**[2]

[1]Lehr- und Forschungsgebiet Kombinatorische Optimierung, RWTH Aachen University, Aachen, Germany
[2]Department of Discrete Mathematics, TU Graz, Graz, Austria

Correspondence
Corinna Mathwieser, Lehr- und Forschungsgebiet Kombinatorische Optimierung, RWTH Aachen University, Aachen, 52062, Germany.
Email: mathwieser@combi.rwth-aachen.de

Abstract

This article studies the Minimum Spanning Tree Problem under Explorable Uncertainty as well as a related vertex uncertainty version of the problem. We particularly consider special instance types, including cactus graphs, for which we provide randomized algorithms. We introduce the problem of finding a minimum weight spanning star under uncertainty for which we show that no algorithm can achieve constant competitive ratio.

KEYWORDS

cactus graphs, competitive analysis, explorable uncertainty, online algorithms, randomized algorithms, spanning tree

1 | INTRODUCTION

Many real world problems do not allow to work with precise data as parts of the input are uncertain or only known approximately. Different approaches to deal with uncertainty include stochastic optimization where the input data is known to follow a specific probability distribution, robust optimization which aims to find good solutions for all possible inputs and explorable uncertainty. In the latter setting, it is possible to obtain more precise or even exact data by making queries. However, any query causes exploration cost. In an applied scenario this might be time, money or other resources which are needed for further measurements.

In this paper, we consider the *Minimum Spanning Tree Problem under Explorable Uncertainty* (MST-U). MST-U requires finding a minimum spanning tree (MST) in an edge weighted graph where the weights are initially not known but can be revealed upon request. In an instance of MST-U, each edge is equipped with an uncertainty set and a query cost. The uncertainty set, usually an interval, is guaranteed to contain the edge's weight. An edge query reveals the edge's true weight. The goal is to find a set Q of queries of minimum cost which allows to find a minimum spanning tree with certainty; that is, given the weights of all edges in Q, there exists an edge set T such that T is a minimum spanning tree for all possible realizations of edge weights with respect to the remaining uncertainty sets. The queries may be chosen adaptively which means that we are allowed to choose the next update based on the previous outcomes of edge queries. Note that while MST-U requires the specification of an edge set which corresponds to an MST, it is not necessary to compute the MST weight.

1.1 | Related work

The first work on problems where parts of the input are uncertain and can be queried is due to Kahan (see [13]) who studied the problem of finding the maximum, the median and the minimum of a set of real values, each of which is known to lie in a given interval. Since then, explorable uncertainty has been considered for different combinatorial problems, for example, shortest paths (see [9]), scheduling (see [4]) and the knapsack problem (see [12]). Erlebach et al. were the first to introduce the Minimum

Spanning Tree Problem under Explorable Uncertainty in [8]. They showed that without restrictions made on the uncertainty sets, no algorithm can achieve constant competitive ratio. This is why subsequently it is assumed that uncertainty sets are either singletons or open intervals. In [8], Erlebach et al. also presented the deterministic algorithm U-RED for MST-U with uniform query costs which achieves competitive ratio 2 and proved that no deterministic algorithm can have a smaller competitive ratio. Moreover, they introduced a different version of the problem, the *Minimum Spanning Tree Problem under Vertex Uncertainty* (V-MST-U) where vertices are points with uncertain locations in the plane and the edge weights correspond to the (Euclidean) distances between the respective end vertices. They showed that U-RED can be adapted to work for the vertex uncertainty setting as well if uncertainty sets are (topologically) open and achieves a competitive ratio of 4 which they proved to be a lower bound for the performance of any deterministic algorithm. An important question left open was the effect of randomization in the setting of MST-U. This question was subsequently answered by Megow et al. in [15] where they provided the randomized algorithm RANDOM with competitive ratio $1 + \frac{1}{\sqrt{2}}$. The best known lower bound for the performance of randomized algorithms is 1.5 which holds true even for K_3 and was observed by Erlebach and Hoffmann in [7]. [15] transformed their randomized algorithm into the deterministic algorithm BALANCE which achieves a competitive ratio of 2 on instances with general query cost. In [15] also considered the problem of finding the weight of an MST under uncertainty, the problem of finding an α-approximate MST under uncertainty as well as a version of MST-U where queries return subintervals instead of the precise edge weights. Test results for MST-U are provided by Focke et al. in [10]. Erlebach et al. ([5]) study an extension of the problem that allows for untrusted predictions; that is, there exist possibly erroneous predictions for the outcome of each query, as may be obtained through machine learning methods. For all integral $\gamma \geq 2$, they present algorithms that are γ-competitive if the predictions are arbitrarily wrong and $(1 + \frac{1}{\gamma})$-competitive if the predictions are correct. In [6], Erlebach and Hoffmann deal with the *verification problem* for MST-U which consists in the problem of computing an optimal query set if the uncertainty sets as well as the exact edge weights are given. They show that the verification problem for MST-U is solvable in polynomial time while the verification problem for the vertex uncertainty problem V-MST-U is NP-hard. Merino and Soto ([16]) study the non-adaptive version of the problem where all the queried elements reveal their values at the same time. They provide a polynomial-time algorithm that finds a query set of minimum cost for the minimum matroid base problem under non-adaptive explorable uncertainty.

1.2 | Our contribution

One of the main difficulties in solving MST-U stems from dealing with edges that are part of several cycles. Thus, it is a natural question to consider the special case of cactus graphs where every edge belongs to at most one cycle. However, the algorithm RANDOM presented in [15] reaches its worst case competitive ratio of $1 + \frac{1}{\sqrt{2}} \approx 1.71$ even for instances where the input graph is a cycle. We introduce an algorithm which achieves competitive ratio 1.5 on instances with cactus graphs. V-MST-U on the contrary, has only been considered in the setting of deterministic algorithms so far. We prove that there exists no randomized algorithm for V-MST-U with a competitive ratio better than 2.5 even if the input graph is a cycle. Unfortunately, structural differences between the edge uncertainty setting and the vertex uncertainty setting impede the straight forward adaption of RANDOM to the vertex uncertainty setting. Instead, we consider the special case of cactus-like instances with uniform query costs where no two cycles share a non-trivial vertex and introduce the algorithm V-RANDOM$_C$ for which we prove a competitive ratio of at most 2.5. While deterministic and randomized algorithms with reasonable performance guarantee exist for MST-U, this is not necessarily the case when aiming to find more specific spanning trees. We demonstrate this by introducing the *Minimum Spanning Star Problem under Explorable Uncertainty* (MSS-U). MSS-U is defined analogously to MST-U except that we want to identify a spanning star of minimum weight rather than a general spanning tree. For MSS-U we derive a negative result with respect to competitive analysis; that is, we show that no algorithm for MSS-U can achieve constant competitive ratio.

The remainder of this paper is organized as follows: In Section 2 we provide the precise definition of MST-U and further basic definitions and concepts used throughout the rest of the paper. Section 3 deals with a randomized algorithm for MST-U on cactus graphs. In Section 4 we prove a lower bound for the performance of randomized algorithms for V-MST-U and introduce an optimal randomized algorithm for V-MST-U on cactus-like graphs with uniform query costs. In Section 5 we study the Minimum Spanning Star Problem under Explorable Uncertainty. We conclude with a brief summary and some open questions in Section 6.

2 | PRELIMINARIES

In this section we will introduce definitions, notation and structural results used throughout this paper.

2.1 | Problem definition MST-U and notation

Definition. An (edge-)uncertainty graph is a tuple $\mathcal{G} = (G, (A_e)_e)$ where $G = (V, E)$ is an undirected, connected graph and for each edge $e \in E$, $A_e \subseteq \mathbb{R}$ is either a singleton set or (topologically) open. The sets A_e with $e \in E$ are called uncertainty sets. A vector $(w_e)_{e\in E}$ of edge weights is called a (feasible) realization of edge weights for the uncertainty graph $\mathcal{G}$ if each weight lies within the corresponding uncertainty set; that is, if $w_e \in A_e$ for all $e \in E$.

An instance of the *Minimum Spanning Tree Problem under Explorable Uncertainty* (MST-U) is specified in terms of an uncertainty graph $\mathcal{G} = (G, (A_e)_e)$ as well as an a priori unknown realization of edge weights $(w_e)_{e\in E}$ and a query cost $q_e > 0$ for each $e \in E$. We refer to $(w_e)_{e\in E}$ as the true (realization of) edge weights. We denote by $n = |V|$ the number of vertices and by $m = |E|$ the number of edges of G. If A_e contains a single element only we say that A_e is *trivial*. An edge e is trivial if A_e is trivial. For an edge $e \in E$, we denote by $L_e := \inf A_e$ the infimum and by $U_e := \sup A_e$ the supremum of the uncertainty set. We will also refer to L_e and U_e as *the lower and the upper limit* of A_e (or of e) respectively. We can query an edge e to determine its true weight w_e. If we query e, the set A_e is updated to a singleton set containing only w_e. The cost of querying an edge e is $q_e > 0$. The goal is to find a minimum cost set of queries needed to identify an edge set T of a minimum spanning tree (MST) in G with respect to the true edge weights w_e. More precisely, a feasible query set is defined as follows:

Definition. Given an uncertainty graph $\mathcal{G}$ with graph $G = (V, E)$ and uncertainty sets A_e, $e \in E$, as well as true edge weights $w_e \in A_e$ for $e \in E$, a set $Q \subseteq E$ is called a feasible query set if there exists a spanning tree T in G such that T has minimum weight with respect to any weight function $\overline{w} : E \to \mathbb{R}$ which fulfills that $\overline{w}(e) = w_e$ if $e \in Q$ and $\overline{w}(e) \in A_e$ if $e \in E - Q$. We say that Q *verifies* T.

MST-U thus consists in finding a feasible query set Q of minimum query cost $\sum_{e\in Q} q_e$. Note that while initially only the uncertainty graph is known, the feasibility and optimality of a query set for an MST-U instance strongly depend on the true realization of edge weights.

Moreover, we will use the following definition: Given an uncertainty graph $\mathcal{G} = (G, (A_e)_e)$ and a cycle C in G, we say that an edge f is *always maximal* in C if $L_f \geq U_e$ for all e in $C - f$.

For the sake of simplicity, we will assume all uncertainty sets to be open intervals or trivial throughout the remainder of the paper. Otherwise the results remain true under the assumption that $\sup A_e$ and $\inf A_e$ can be accessed in $\mathcal{O}(1)$ time for all $e \in E$.

2.2 | Structural aspects of MST-U

In the following we will recall some structural insights into MST-U which were provided by Megow et al. in [15] and will be used throughout the remainder of the paper. Given an instance of MST-U, a *lower limit tree* is the edge set of a minimum spanning tree in G with respect to edge weights w_e^L where $w_e^L := L_e = U_e$ for trivial edges and infinitesimally close to the lower limit of the edge's uncertainty set (i.e., $w_e^L := L_e + \varepsilon$ for infinitesimally small $\varepsilon > 0$) for non-trivial edges. An *upper limit tree* and edge weights w_e^U for its computation are defined analogously.

Lemma 1 ([15]). *Let T_L, T_U be a lower and an upper limit tree respectively. All edges in $T_L - T_U$ with non-trivial uncertainty sets lie in any feasible query set.*

Hence, the instance can be preprocessed by querying all edges in $T_L - T_U$ until we obtain an instance where $T_L - T_U$ contains only trivial edges. Moreover, we can obtain identical upper and lower limit trees $T_L = T_U$ from the preprocessed instance by picking the same ordering for identical trivial edges in the computation of T_L and T_U. We may thus assume that $T_L = T_U$.

Now recall the following well-known properties of minimum spanning trees: Given a cycle C in a weighted graph G, we have that if an edge e is such that $w_e > w_{e'}$ for all $e' \in C - e$ then e is not contained in any MST. If $w_e \geq w_{e'}$ for all $e' \in C$ then there exists an MST of G which does not contain e. In the context of MST-U this means that if we encounter a cycle C with an always maximal edge e then e can be discarded in the search for an MST. Conversely, given a cycle C and a non-trivial edge f which has largest upper limit U_f in C, it is impossible to verify an MST which contains f without querying f because f is a candidate for an edge with strictly largest weight in C. Possible candidates for a largest weight edge in C are f and all edges e in C with possibly larger weight than f, i.e. with $U_e > L_f$. We will refer to these edges as *neighbors* of f in C and denote the set of all neighbors by $X_C(f) := \{e \in C - f | U_e > L_f\}$. We will write $X(f)$ instead of $X_C(f)$ if the intended cycle C is unambiguous. In [15], Megow et al. prove the following with respect to f and its neighbor set:

Lemma 2 ([15]). *Let T_U be an upper limit tree. Let f be an edge in $G - T_U$ with smallest lower limit L_f in $G - T_U$ and let C be the cycle in $T_U + f$. If no edge in C is always maximal, then any feasible query set contains f or $X(f)$. Moreover, if there is an edge $g \neq f$ in C such that $L_g \geq L_f$ and $U_g \neq L_f$, then f lies in any feasible query set.*

If $T_U = T_L$ is a lower limit tree too and no edge is known to have maximum weight in C even after querying f or all edges in X(f), then an edge g in C with maximum upper limit U_g lies in any feasible query set.

2.3 | Vertex uncertainty problem

In an instance of the *Minimum Spanning Tree Problem under Explorable Vertex Uncertainty* (V-MST-U) we are given an undirected, connected graph G where each vertex corresponds to a point in the Euclidean plane. The weight of an edge is equal to the Euclidean distance between its end vertices. Instead of the precise location of a vertex v we are given an uncertainty set $A_v \subseteq \mathbb{R}^2$. A_v can either be a singleton set (in which case we refer to the vertex and the uncertainty set as trivial) or an open subset of $\mathbb{R}^2$. An algorithm can query a vertex v at query cost q_v to reveal its exact location. We define a *(vertex-)uncertainty graph* as well as a *feasible (vertex) query set* in the same way as in the setting of MST-U. Then V-MST-U is defined analogously to MST-U; that is, we want to identify a feasible vertex query set of minimum query cost.

In order to apply methods developed for MST-U to V-MST-U it is common to transform vertex uncertainty sets into edge uncertainty sets by computing all possible distances between vertices. The resulting instance is referred to as associated edge instance:

Definition. Given an instance $\mathcal{I}$ of V-MST-U with graph $G = (V, E)$ and uncertainty sets $A_v, v \in V$, the associated edge instance $\mathcal{I}'$ is an instance of MST-U with graph G and uncertainty sets $A_{\{u,v\}} = \{d(u', v') | u' \in A_u, v' \in A_v\}$.

Note that by standard properties of Euclidean topology, uncertainty sets of the associated edge instance of an V-MST-U instance with open or trivial uncertainty sets are open or trivial. For the remainder of the paper we will assume V-MST-U instances to be such that uncertainty sets in the associated edge instances are open intervals or trivial. However, results will remain true for the more general case under the assumption that we can access infima and suprema of uncertain vertex distances.

In the following, we define a feasible realization of vertex positions and a feasible realization of edge lengths in the setting of V-MST-U.

Definition. Let $\mathcal{G} = (G, (A_v)_v)$ be a vertex uncertainty graph. We call $(\overline{v})_{v \in V}$ with $\overline{v} \in \mathbb{R}^2$ for all $v \in V$ a feasible realization of vertex positions if $\overline{v} \in A_v$ for all $v \in V$. A vector $(w_e)_{e \in E}$ is called a feasible realization of edge lengths if there is a feasible realization $(\overline{v})_{v \in V}$ of vertex positions such that $\|\bar{u} - \overline{v}\|_2 = w_e$ for all $e = \{u, v\} \in E$ where $\|\cdot\|_2$ denotes the Euclidean distance.

Note that while every feasible realization of edge lengths for a V-MST-U instance is also a feasible realization of edge weights for the associated edge instance, not every feasible realization of edge weights for the associated edge instance is feasible for the original V-MST-U instance due to dependecies between edge lengths of adjacent edges.

In the following we also provide a definition for the terms “always maximal” and “neighbor set” for V-MST-U.

Definition. Let $\mathcal{G} = (G, (A_v)_v)$ be a vertex uncertainty graph and C a cycle in G. Let f be an edge in C with largest upper limit in the associated edge instance. We denote by $X_C(f)$ the set of edges $e \in C$ such that $w_e > w_f$ for some feasible realization $(w_h)_{h \in E}$ of edge lengths and call $X_C(f)$ the neighbor set of f in C. The edge f is called always maximal in C if $X_C(f) = \emptyset$.

We again write $X(f)$ instead of $X_C(f)$ if the cycle C is unambigous. Note that the neighbor set in the associated edge instance is not necessarily identical to the neighbor set of an edge f in a V-MST-U instance as is displayed in Figure 1.

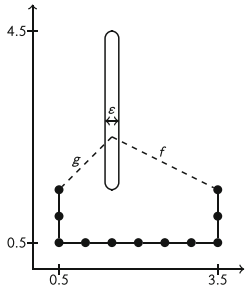

FIGURE 1 A vertex uncertainty graph with cycle C where black dots indicate trivial vertices and g and f share the only non-trivial vertex. Clearly $U_f = \max\{U_e : e \in C\}$ and $U_g > L_f$ but f is always maximal in C.

The edge f in Figure 1 is always strictly larger than the length of edge g (even though their uncertainty intervals in the associated edge instance overlap.) Given an instance of V-MST-U, we say that an edge f *dominates* an edge g if $w_f > w_g$ for any feasible realization $(w_e)_e$ of edge lengths. Note the following:

> *Observation* 3. For any two edges f and g in an instance of V-MST-U we have that either (1) f dominates g, (2) g dominates f, (3) $w_f = w_g$ for any feasible realization $(w_e)_e$ of edge lengths or (4) $w_f < w_g$ and $\overline{w}_f > \overline{w}_g$ for two feasible realizations $(w_e)_e, (\overline{w}_e)_e$ of edge lengths. In case (3) where edge lengths coincide for all realizations we say that f and g have always identical edge lengths. Note that f and g can only have always identical edge lengths if either f and g both have two trivial end vertices or in case that f and g share a common non-trivial end vertex while the vertices $v \in f - g$ and $w \in g - f$ are both trivial and have identical positions $\overline{v} = \overline{w}$.

The correctness of Observation 3 is easy to see as we cannot have that $w_f \leq w_g$ for any feasible realization $(w_e)_e$ of edge lengths if equality is attained for some but not all realizations, due to uncertainty sets being open or trivial.
Finally, we define lower and upper limit trees for V-MST-U. A lower (upper) limit tree T_L (T_U) for an instance of V-MST-U is the edge set of a lower (upper) limit tree for the associated edge instance with the following restrictions imposed on the ordering of edges during the computation of T_L and T_U (e.g., with Kruskal):

- If an edge f dominates an edge g then g is added to T_L (T_U) prior to f. Note that the restriction does not interfere with the well-definedness of T_L (T_U) as lower (upper) limit tree for the associated edge instance as $L_f \geq L_g$ and $U_f \geq U_g$ if f dominates g.
- If two edges f and g have always identical edge lengths then the ordering is arbitrary but fixed, in the sense that we choose the same ordering for the computation of T_L and T_U.

2.4 | Performance analysis

To analyze the quality of a solution found by an algorithm we compute the competitive ratio between the query cost of the algorithm's solution and the cost of an optimal query set. An optimal query set is an optimal solution to the offline problem where all edge weights (or vertex positions) are known a priori.

> **Definition.** Let $\mathcal{I}$ be an instance of MST-U (or V-MST-U). By $OPT(\mathcal{I})$ we denote the cost of an optimal query set for instance $\mathcal{I}$. For an algorithm ALG we denote by $ALG(\mathcal{I})$ the cost of the query set which the algorithm outputs when applied to $\mathcal{I}$. We say that ALG achieves competitive ratio $c \geq 1$ or is c-competitive if
>
> $$\frac{ALG(\mathcal{I})}{OPT(\mathcal{I})} \leq c$$
>
> for all instances $\mathcal{I}$. A randomized algorithm is said to achieve competitive ratio $c \geq 1$ if the ratio between the expected query cost of the algorithm's solution and the cost of an optimal solution is at most c; that is, if
>
> $$\frac{\mathbb{E}(ALG(\mathcal{I}))}{OPT(\mathcal{I})} \leq c$$
>
> for all instances $\mathcal{I}$.

3 | A RANDOMIZED ALGORITHM FOR MST-U ON CACTUS GRAPHS

In this section we consider a randomized algorithm for MST-U on cactus graphs. A *cactus graph* is a connected graph in which any two cycles share at most one vertex. A feasible query set needs to allow for the detection of a maximum weight edge in each cycle. For work on detecting a data item of minimum value with uncertainty intervals see [1, 3, 13, 14]. It can be easily seen that an optimal query set for a cactus graph consists of the disjoint union of the optimal query sets for each of the graph's cycles. Thus we can first consider MST-U on a cycle and then extend the result to the case of an edge uncertainty graph which is a cactus. Once we are able to treat cycles separately, it is possible to achieve an optimal competitive ratio of 1.5 using the following observation: Assume that we have preprocessed the instance such that $T_L = T_U$ and let f be an edge in $G - T_L$ and let C be the cycle in $T_L + f$. Assume moreover that no edge in C is always maximal. Then Lemma 2 guarantees that if we start by querying f and continue to query edges in order of decreasing upper limit until an edge in C is always maximal, then we have queried at most one edge which is not in every feasible query set.

Theorem 4. *For cactus graphs there exists an algorithm Random$_C$ with competitive ratio at most 1.5, which is best possible. Moreover, if G is a cycle, Random$_C$ achieves competitive ratio $1 + \frac{q_{X(f)} \cdot q_f}{q_{X(f)}^2 + q_f^2}$ where $q_{X(f)}$ denotes the cumulative query cost of edges in $X(f)$.*

Proof. We first consider a graph C which consists of a single cycle. Assume again that we have preprocessed the instance such that $T_L = T_U$ and that no always maximal edge in C is known. Let f be the edge in $E\text{-}T_L$ and let $q_{X(f)} := \sum_{e \in X(f_i)} q_e$ be the query cost of all neighbors of f in C. With probability p, our algorithm starts by querying all edges in $X(f)$. With probability $1 - p$, its first step is to query f. Once it has queried $X(f)$ or f, it queries edges in order of decreasing upper limit until an always maximal edge in C can be identified.

We distinguish two cases: either an optimal solution queries f or it does not. If an optimal solution does not query f, it must by Lemma 2 query all of the neighbors in $X(f)$ and thus makes queries at cost $q_{X(f)}$. In this case with probability p, we query the same edges as the optimal solution and achieve competitive ratio 1. With probability $1 - p$, RANDOM$_C$ queries f and possibly all edges in the neighbor set $X(f)$ such that the competitive ratio is at most $\frac{q_{X(f)} + q_f}{q_{X(f)}}$. Thus, in this case the overall competitive ratio is at most $1 + (1 - p)\frac{q_f}{q_{X(f)}}$.

If an optimal solution queries f, RANDOM$_C$ queries the same edges as the optimal solution if it starts by querying f; that is, with probability $1 - p$. With probability p, RANDOM$_C$ starts by querying all neighbors in $X(f)$ and might have to query f too, while an optimal query set might contain f only. Summing up, the competitive ratio is bounded by $1 + p \cdot \frac{q_{X(f)}}{q_f}$ in this case. By setting $p = \frac{q_f^2}{q_f^2 + q_{X(f)}^2}$, the obtained bounds for both cases coincide and equal

$$1 + \frac{q_{X(f)} q_f}{q_f^2 + q_{X(f)}^2}.$$

Note that the following equivalences hold:

$$\begin{aligned} \frac{q_{X(f)} \cdot q_f}{q_f^2 + q_{X(f)}^2} \leq \frac{1}{2} \quad &\Leftrightarrow \\ 2 \cdot q_{X(f)} \cdot q_f \leq q_f^2 + q_{X(f)}^2 \quad &\Leftrightarrow \\ 0 \leq (q_f - q_{X(f)})^2. \end{aligned}$$

Thus, RANDOM$_C$ is 1.5-competitive on instances where the uncertainty graph is a cycle if p is chosen as above.

For an instance $\mathcal{I}$ where the uncertainty graph is a general cactus G, we again preprocess the instance such that $T_L = T_U$ is a lower and an upper limit tree. Denote the edges in $G - T_L$ by $f_1, \ldots, f_{m-n+1}$, the order is arbitrary. Let C_i denote the cycle in $T_L + f_i$. We apply RANDOM$_C$ for cycles as described above to each of the cycles separately. Let Q_i^* denote an optimal query set for an instance where the uncertainty graph consists only of C_i and the uncertainty sets and weights are as in $\mathcal{I}$. We set $OPT_i := \sum_{e \in Q_i^*} q_e$, $i = 1, \ldots, m - n + 1$. As any two cycles in G do not share common edges and edges that lie in no cycle need to be part of any spanning tree, the disjoint union $Q^* := \dot{\bigcup}_{i=1}^{m-n+1} Q_i^*$ is an optimal solution for $\mathcal{I}$ and thus $OPT(\mathcal{I}) = \sum_{i=1}^{m-n+1} OPT_i$. Moreover, the structure of cactus graphs guarantees that for any $i = 1, \ldots, m - n + 1$, $X(f_i)$ is independent of the choice of queries that RANDOM$_C$ makes when applied to C_j with $j \neq i$ as well as of the queries' outcome. Let q_{prep} denote the cost of queries made in the preprocessing. We denote by ALG_i the cost of the queries RANDOM$_C$ makes when applied to C_i, $i \in \{1, 2, \ldots, m - n + 1\}$. As we expect at most $1.5 \cdot OPT_i$ queries when applying RANDOM$_C$ to C_i, we obtain that

$$\frac{\mathbb{E}[\text{RANDOM}_C(\mathcal{I})]}{OPT(\mathcal{I})} = \frac{q_{prep} + \sum_{i=1}^{m-n+1} \mathbb{E}[ALG_i(\mathcal{I})]}{OPT(\mathcal{I})} \leq \frac{q_{prep} + \sum_{i=1}^{m-n+1} 1.5 \cdot OPT_i}{q_{prep} + \sum_{i=1}^{m-n+1} OPT_i} \leq 1.5.$$

This completes the proof of the competitiveness. Note that the correctness of RANDOM$_C$ is immediate as for each cycle the algorithm proceeds to query edges until an edge becomes always maximal. The claim that the competitive ratio is best possible follows from the fact that no randomized algorithm can achieve a competitive ratio less than 1.5 even on triangles, see [7]. A precise description of RANDOM$_C$ for general query costs is given in Algorithm 1. ∎

4 | V-MST-U AND RANDOMIZATION

We will now turn to the vertex uncertainty version of MST-U which has not been considered in the context of randomized algorithms so far. We start by showing that no randomized algorithm for V-MST-U can achieve a better competitive ratio than 2.5. Our proof is based on the proof of the bound for the deterministic performance guarantee by [8].

Algorithm 1. The algorithm RANDOM$_C$ for MST-U with general query costs in cactus graphs

input : An instance of MST-U with cactus graph $G = (V, E)$, uncertainty sets A_e and query costs q_e, $e \in E$
output: A feasible query set Q

Draw b uniformly at random from $[0, 1]$;
Preprocess the instance such that $T_L = T_U$;
Index the edges $f_1, \dots, f_{m-n+1}$ in E-T_L arbitrarily;
Initialize $Q = \emptyset$;
for $i \leftarrow 1$ **to** $m - n + 1$ **do**
  Add f_i to T_L and let C_i be the unique cycle closed;
  Let $X(f_i)$ be the set of edges $g \in T_L \cap C_i$ with $U_g > L_{f_i}$;
  Let $q_i := \sum_{e \in X(f_i)} q_e$;
  if $X(f_i) \neq \emptyset$ **then**
    if $b \leq \frac{q_{f_i}^2}{q_{f_i}^2 + q_i^2}$ **then**
      add all edges in $X(f_i)$ to Q and query them.
    else
      Add f_i to Q and query f_i.
    while *no edge in the cycle* C_i *is always maximal* **do**
      Query an unqueried edge $e \in C_i$-Q with maximum U_e and add it to Q.

Theorem 5. *No randomized algorithm for V-MST-U can achieve a competitive ratio less than 2.5. This remains true even for cycles and under the assumption of uniform query costs.*

Proof. We will prove the theorem by applying a variant of Yao's Principle (see [2]) which allows to derive a lower bound for the performance of randomized algorithms from the best possible expected performance of a deterministic algorithm against a finite randomized family of instances. A *finite randomized family of instances* is a pair $(\mathscr{R}, p)$ where $\mathscr{R}$ is a finite set of instances for V-MST-U and p is a probability vector of length $|\mathscr{R}|$ which indicates for each instance $R \in \mathscr{R}$ the probability of its occurrence.

In [6], Erlebach and Hoffmann define a graph G with four non-trivial vertices A, B, C and D. For each such vertex $v \in \{A, B, C, D\}$ they provide an instance R_v that specifies a realization of vertex positions. See Figure 2 for an illustration of G with the uncertainty sets $\mathscr{A}_A$, $\mathscr{A}_B$, $\mathscr{A}_C$ and $\mathscr{A}_D$ alongside illustrations of the instances R_A, R_B, R_C and R_D with the corresponding realizations of vertex positions. Black dots represent trivial vertices. Each distance between two neighboring trivial vertices or between a trivial vertex and an uncertainty set equals 1. The non-trivial uncertainty sets of A, B, C and D are long, thin open areas of length 2 and small positive width $\varepsilon > 0$. The distance between the uncertainty sets of A and B is 7 while the distance between the uncertainty sets of C and D is 4. Note that all edges which have at least one (initially) trivial end vertex are necessarily part of any minimum spanning tree. Depending on the realization of vertex positions, $\{A, B\}$ and $\{C, D\}$ can both be edges in an MST. The realization of vertex positions in R_A, R_B, R_C and R_D can be described as follows: A (D) is located on the far left of $\mathscr{A}_A$ ($\mathscr{A}_D$) when considering R_A (R_D). Otherwise A (D) is located on the far right of $\mathscr{A}_A$ ($\mathscr{A}_D$). B (C) on the contrary, is located on the far right of $\mathscr{A}_B$ ($\mathscr{A}_C$) when considering R_B (R_C) while otherwise B (C) is located on the far left of $\mathscr{A}_B$ ($\mathscr{A}_C$). Note that for an instance R_v with $v \in \{A, B, C, D\}$, vertex positions are such that $\{v\}$ is the only optimal query set: After querying the vertices in $\{A, B, C, D\} - \{v\}$, the uncertainty interval of $\{A, B\}$ is $(7 + \varepsilon, 9 + \varepsilon)$ and the length of $\{C, D\}$ is $8 - 2\varepsilon$ if $v = A, B$ and the uncertainty interval of $\{C, D\}$ is $(6 - \varepsilon, 8 - \varepsilon)$ and the length of $\{A, B\}$ is $7 + 2\varepsilon$ if $v = C, D$. In both cases, the length of the trivial edge is contained in the uncertainty set of the edge incident to v. Querying v on the other side, clearly results in $\{A, B\}$ being an always maximal edge.

Consider now the randomized family of instances $(\mathscr{R}, p)$, where $\mathscr{R} = \{R_A, R_B, R_C, R_D\}$ and $p(R_v) = \mathbb{P}[R = R_v] = 0.25$ for $v \in \{A, B, C, D\}$. Then no deterministic algorithm ALG achieves a better expected competitive ratio $\mathbb{E}_{R \sim_p \mathscr{R}}\left(\frac{ALG(R)}{OPT(R)}\right)$ than the algorithm ALG_1 which queries A, B, C, D (or less if an MST can already be identified) in this order independently from the queries' results. This is due to the fact that querying v only reveals whether or not we are facing instance R_v but if not, it is indistinguishable which of the remaining instances it might be.

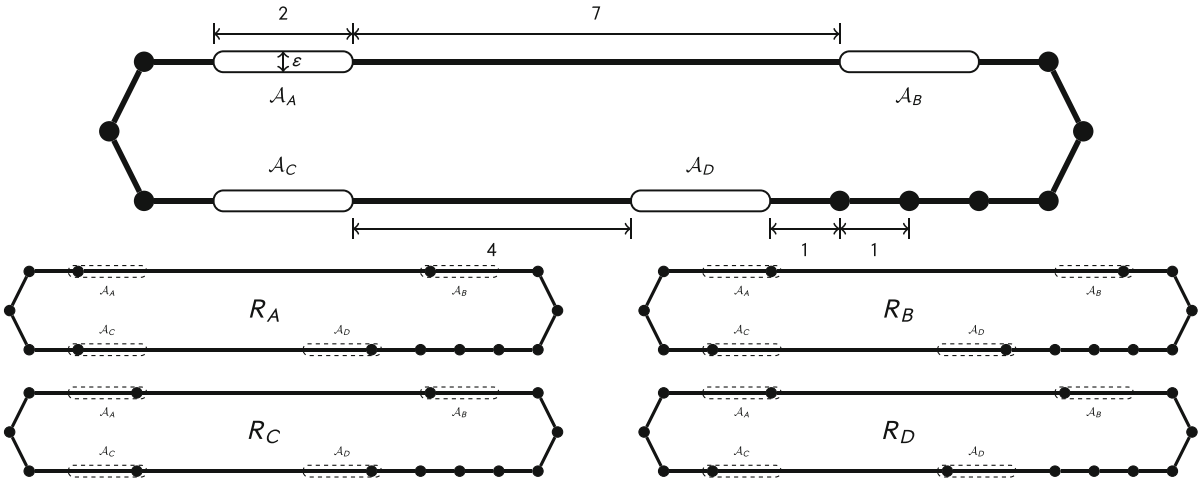

FIGURE 2 Instances R_A, R_B, R_C and R_D for the finite randomized family $(\mathscr{R}, p)$ of instances.

Then by Yao's Principle, no randomized algorithm has a better performance than

$$\min_{ALG\in\mathscr{A}} \mathbb{E}_{R\sim_p\mathscr{R}}\left(\frac{ALG(R)}{OPT(R)}\right) = \mathbb{E}_{R\sim_p\mathscr{R}}\left(\frac{ALG_1(R)}{OPT(R)}\right) = 0.25\cdot(ALG_1(R_A)+ALG_1(R_B)+ALG_1(R_C)+ALG_1(R_D))$$
$$= 0.25\cdot(1+2+3+4) = 2.5,$$

where $\mathscr{A}$ denotes the class of all deterministic algorithms. ∎

4.1 | A randomized algorithm for V-MST-U on cactus-like graphs

V-MST-U structurally differs from MST-U mainly due to the following three aspects:

a) Querying a single end vertex of an edge already yields partial information about the edge's length.
b) A vertex may be incident to several edges. Thus, knowing its precise position impacts on the possible lengths of all adjacent edges.
c) Knowing the ordering of edge lengths of adjacent edges is not necessarily linked to the knowledge of precise vertex positions or edge lengths, even if the uncertainty intervals overlap (see Figure 1).

While for MST-U it is sometimes possible to identify edges which have to lie in any feasible query set, it becomes significantly harder to tell whether a vertex has to be queried due to aspect a). However, we will make use of the weaker statement that for certain edges at least one out of two end vertices needs to be queried.

Similarly to the special case of cactus graphs for MST-U, we will now consider instances of V-MST-U where no two cycles share a non-trivial vertex. This makes it easier to deal with aspect b), as vertex queries may only impact on the possible lengths of at most two adjacent edges. We will refer to vertex-uncertainty graphs where cycles do not intersect in non-trivial vertices as *cactus-like* graphs. For the remainder of the section we will moreover consider uniform query costs. Our algorithm considers cycles separately. For an instance $\mathcal{I}$ with a cycle C, our algorithm V-RANDOM$_C$ works as follows: First, it computes the associated edge instance and preprocesses the instance s.t. $T_L = T_U$ holds. Usually, our algorithm deterministically queries the end vertices of the edge $f \in C - T_U$ and will then query the end vertices of edges in C in order of decreasing upper limit in the associated edge instance until a longest edge in C can be identified. Only if the neighbor set is small (where "small" is yet to be defined) we will make use of randomization. More precisely, the algorithm will make use of subroutines RAND1, RAND2 or RAND3, depending on the size of the neighbor set. In case that the neighbor set is large and the algorithm performs deterministically as described above, we will say that the algorithm follows the *deterministic procedure*.

The remainder of the section is structured as follows: First we prove some structural observations that will be used to provide a pre-processing and prove the competitiveness of the algorithm and its sub-routines. After that we prove an upper bound on the number of queries made by the deterministic procedure in Lemma 9. We proceed by describing RAND3 and proving its competitiveness in Lemma 10. We then describe RAND1 and RAND2 and prove the competitiveness of RAND2 in Lemma 11. Ultimately, we prove the overall performance guarantee for V-RANDOM$_C$ (including the competitiveness of RAND1) in Theorem 5.

4.1.1 | Structural observations

Clearly, an edge f in an instance of MST-U which has largest upper limit U_f in a cycle C can turn out to have strictly largest weight in C even after knowing the precise weights of all edges in $C-f$. For an instance of V-MST-U, there is not necessarily a realization of edge lengths with $w_f > w_e$ for all $e \in C-f$ even after querying all vertices in $C-f$ due to aspect c). We make use of the following, slightly weaker statement instead:

Lemma 6. *Let $\mathcal{G}$ be a vertex uncertainty graph with graph $G=(V,E)$ and let $f \in E$ be an edge in G. Let L_e, U_e with $e \in E$ denote the original infimum or supremum of an edge e prior to making any queries. Let $g \neq f$ be an edge such that $U_g \leq U_f$ ($L_g \geq L_f$), g does not dominate f (f does not dominate g) and f and g do not have always identical edge lengths. Then even after querying all vertices in $E-f$, there exists a feasible realization $(w_e)_{e\in E}$ of edge lengths with $w_f > w_g$ ($w_f < w_g$).*

Proof. We prove that there is a realization with $w_f > w_g$ for edges g of the specified type. The proof for a realization with $w_f < w_g$ works analogously. Let g first be an edge with $U_g < U_f$. As the queries have not altered the uncertainty set of f, there is a feasible realization such that f has length $w_f \in (U_g, U_f)$. Now assume that $U_g = U_f$ and that $g-f$ contains a non-trivial vertex. Let U_g^{new} denote the upper limit of g after querying all vertices in $E-f$. As $g-f$ contains a non-trivial vertex, the uncertainty set of g is altered by the queries. Thus, $U_g^{new} < U_g$ and there is a feasible realization such that f has length $w_f \in (U_g^{new}, U_f)$. Finally, assume that $g-f$ does not contain a non-trivial vertex. Then querying the vertices in $T-f$ does not alter the uncertainty set of g; that is, $U_g^{new} = U_g$ and $L_g^{new} = L_g$. As g neither dominates f nor has always identical edge length with f, there is a realization $(\overline{w}_e)_{e\in E}$ of edge lengths where f is longer than g (see Observation 3). Then $(w_e)_{e\in E}$ where w_e is the true length of any edge $e \neq f, g$ and $w_f = \overline{w}_f > \overline{w}_g = w_g$ is a feasible realization of edge lengths after querying all vertices in $E-f$. ∎

We will now provide a preprocessing for V-MST-U on cycles that allows for the initial assumption that $T_L = T_U$. Our preprocessing consists in repeatedly computing T_L and T_U and querying the end vertices of the edge in $T_L - T_U$. However, the property $T_L = T_U$ does not necessarily remain true throughout the algorithm V-RANDOM$_C$ as vertex queries have an impact on the uncertainty sets of all adjacent edges. We will now prove that at least half of the vertices queried during the preprocessing lie in any feasible query set.

Lemma 7. *Let $\mathcal{I}$ be a V-MST-U instance on a cycle C. Let T_L, T_U be a lower and an upper limit tree and $f \in C-T_U$, $g \in C-T_L$ the edges which are not included in the respective trees. If $g \neq f$, then f intersects every feasible query set Q for $\mathcal{I}$.*

Proof. Let Q be a feasible query set, let T be an MST that is verified by Q and let $(w_e)_e$ be the underlying realization of edge lengths in $\mathcal{I}$. First note that if the edge lengths of f and g are always identical then the identical ordering of edges with always identical edge lengths imposes $f = g$. Moreover, observe that f cannot dominate g otherwise $L_f \geq L_g$ and f would have been added to T_L after g, a contradiction. Assume now that $f \neq g$. Then f is not trivial otherwise $L_f = U_f \geq U_g$ implies that f dominates g. We first consider the case where $f \notin T$. As $f \in T_L$ and $g \in C - T_L$ we have that $L_g \geq L_f$. Then Lemma 6 guarantees that f intersects Q as otherwise f can realize to be strictly shorter than g even if all vertex positions of vertices in Q are known.

Now assume that $f \in T$. Let h denote the edge in $C - T$. As $h \in T_U$ and $f \in C - T_U$ we have that $U_f \geq U_h$. Observe that h cannot dominate f otherwise h would have been added to T_U after f, a contradiction. First consider the case where h and f do not have always identical edge lengths. Then by Lemma 6, f can realize to be strictly longer than h even if all vertex positions of vertices in $C-f$ are known. Thus f needs to intersect Q to verify that $w_f \leq w_h$. Consider now the case where f and h have always identical edge lengths. Then $g \neq h$ and g and h do not have always identical edge lengths. As f is non-trivial, h has a single non-trivial vertex which lies in f. Moreover, h does not dominate g and $g \in C - T_L$ implies that $L_g \geq L_h$. Then again Lemma 6 guarantees that h can realize to be strictly shorter than g even if all vertex positions of vertices in Q are known. ∎

Finally, we provide a lemma which is an analogue to Lemma 2 for V-MST-U on cycles.

Lemma 8. *Consider an instance $\mathcal{I}$ of V-MST-U on a cycle C with edge lengths $(w_e)_e$ and upper limit tree T_U. Let f denote the edge in $C - T_U$ and assume that no edge in C is always maximal. Then any feasible query set for $\mathcal{I}$ contains a non-trivial vertex in f or a vertex cover of $X(f)$ (which consists of non-trivial vertices only). Moreover, if there is a trivial edge e in C such that $w_e \in A_f$ then any feasible query set of $\mathcal{I}$ contains a non-trivial vertex in f.*

Proof. Let Q be a feasible query set for $\mathcal{I}$ and T a minimum spanning tree verified by Q. Assume by contradiction that Q neither contains an end vertex of f nor a vertex cover of $X(f)$ which consists of non-trivial vertices only. First assume that $f \notin T$. Let e be an edge in $X(f)$ such that no end vertex of e lies in Q. Then even after querying all vertices in Q, there is a feasible realization $(\overline{w}_h)_h$ of edge lengths such that $\overline{w}_e > \overline{w}_f$ by definition of the neighbor set and the fact that no end vertex of e or f is contained in Q. Thus, Q cannot verify T with $f \notin T$. Now assume that $f \in T$. Let $g \neq f$ be the edge in $C - T$. Given that no edge dominates f we either have a feasible realization $(\overline{w}_h)_h$ of edge lengths with $\overline{w}_g < \overline{w}_f$ or we have that f and g have always identical edge lengths. Assume first that f and g have always identical edge lengths. Note that f is not trivial, otherwise f is always maximal. Hence, g and f share a common end vertex v while the other two end vertices in $(g \cup f) - v$ are trivial (see Observation 3). Let e be an edge in $X(f)$ such that no end-vertex of e lies in Q. Then even after querying all vertices in Q, there is a feasible realization $(\overline{w}_h)_h$ of edge lengths such that $\overline{w}_e > \overline{w}_f = \overline{w}_g$ by definition of the neighbor set and the fact that neither v nor an end vertex of e is contained in Q. Hence, Q cannot verify T with $g \notin T$ in the case where f and g have always identical edge lengths. Now assume that there exists a feasible realization $(\overline{w}_h)_h$ of edge lengths with $\overline{w}_g < \overline{w}_f$. Then Lemma 6 guarantees that querying Q with $f \cap Q = \emptyset$ allows for a realization of edge lengths where g is strictly shorter than f, contradicting again that Q verifies T with $g \notin T$.

Finally assume that there is a trivial edge e in C such that $w_e \in A_f$. As f is non-trival, we have that $f \neq e$ and f and e do not have always identical edge lengths. Clearly, f does not dominate e. Hence, $e \in X(f)$ and the second statement in Lemma 8 follows directly from the first statement in Lemma 8 as there exists no vertex cover of $X(f)$ which consists of non-trivial vertices only. ∎

4.1.2 | Deterministic procedure

We will now argue why the deterministic procedure which repeatedly queries the end vertices of an undominated edge with largest upper limit makes at most $2 \cdot OPT(\mathcal{I}) + 2$ queries.

Lemma 9. *Let $\mathcal{I}$ be an instance of V-MST-U on a cycle C with uniform query costs. Let ALG be an algorithm which iteratively queries the end vertices of an undominated edge with largest upper limit in the associated edge instance until a longest edge is found. Then ALG makes at most $2 \cdot OPT(\mathcal{I}) + 2$ queries when applied to $\mathcal{I}$. Moreover, if $\mathcal{I}$ is such that $T_L = T_U, f_1 \in C - T_L$ is the first edge with largest upper limit queried by ALG and Q^* is an optimal query set with $f_1 \cap Q^* \neq \emptyset$, then ALG makes at most $2.5 \cdot OPT(\mathcal{I})$ queries when applied to $\mathcal{I}$.*

Proof. We will denote by $\{u_i, v_i\}$ the set of vertices that *ALG* queries during the i'th iteration. (Note that $u_i = v_i$ is possible if the edge considered in the i'th iteration has one trivial end vertex.) Moreover, we will denote by V_i the set of vertices queried during the first i iterations such that $|V_t| = ALG(\mathcal{I})$ where t is the last iteration. Let Q^* be an optimal query set and denote by A_e^i, U_e^i and L_e^i the uncertainty set, upper limit and lower limit of an edge e in the beginning of iteration i. We will argue by induction that for each iteration i it either holds that at least one vertex in $\{u_i, v_i\}$ lies in Q^* or that for each $j < i$ at least one vertex in $\{u_j, v_j\}$ lies in Q^*. This claim trivially holds for the first iteration. Assume now that the claim is known to be true for iterations $1, \dots, i-1$ and consider the i-th iteration. Note that u_i and v_i are non-trivial end vertices of an undominated edge f which has largest upper limit U_f^i in the uncertainty graph considered in iteration i.

We distinguish two cases: (a) f has a trivial neighbor in iteration i and (b) f has no such trivial neighbor. (a) Let $e \in C$ be a trivial neighbor of f; that is, $A_e^i = \{w_e\} \subseteq A_f^i$. Then by Lemma 8 we need to know the position of at least one of f's end vertices in order to be able to identify a minimum spanning tree. (b) Now assume that none of f's neighbors is trivial. By induction hypothesis we know that there is at most one $j < i$ for which we have not yet argued that $\{u_j, v_j\}$ intersects with Q^*. Let h be the edge with largest upper limit in iteration j such that u_j and v_j are end vertices of h. Then w_h can neither lie in A_f^i otherwise $A_h^i = \{w_h\} \subseteq A_f^i$ (h is a trivial neighbor of f) nor can we have that $U_h^i = w_h > U_f^i$ as f has largest upper limit in the i-th iteration. Hence we know that $w_h < w_f$ and thus h lies in any MST. Let g be a longest edge in C. Clearly, g and h do not have always identical edge lengths. By the choice of h we have that $U_h^j \geq U_g^j \geq w_g$ and g does not dominate h. We thus cannot verify that $w_h \leq w_g$ without querying a vertex in h by Lemma 6. Thus without querying neither u_j nor v_j (which are the non-trivial end vertices of h in iteration j) we will not be able to verify an MST which does not contain g.

Observe now that the claim implies the first statement of the lemma: Let j be the last iteration for which we cannot guarantee that v_j or u_j lies in any feasible query set. Then we know that at least half of V_{j-1} lies in Q^* and that for each iteration $i > j$ we know that v_i or u_i lies in Q^*. Hence, $OPT(\mathcal{I}) = |Q^*| \geq \frac{|V_{j-1}|}{2} + \frac{|V_t| - |V_j|}{2} = \frac{|V_t| - |\{u_j, v_j\}|}{2} \geq \frac{ALG(\mathcal{I}) - 2}{2}$.

Finally, we prove the second statement of the lemma. Note that if $j \neq 1$ then $u_j = v_j$: By the choice of j, the weight of any trivial edge in the j'th iteration is not contained in A_e^j where e is the edge with largest upper limit in iteration j and non-trivial end vertices u_j and v_j. Thus $w_{f_1} \leq L_e$ which cannot be the case if both end vertices of e remain unqueried until iteration j due to our preprocessing. Hence, the first iteration is the only iteration where we might query two vertices none of which is in Q^* and thus if $Q^* \cap f_1 \neq \emptyset$ then ALG makes at most $2 \cdot OPT(\mathcal{I}) + 1$ queries; that is, it achieves competitive ratio 2.5. unless $OPT(\mathcal{I}) = 1$. However, if $OPT(\mathcal{I}) = 1$ and Q^* intersects f_1 then Q^* is a subset of f_1 and thus the deterministic procedure finishes after querying only the end vertices of f_1. This proves the second statement of the lemma. ∎

4.1.3 | Randomization

The above lemma guarantees that the deterministic procedure achieves a competitive ratio of $2 + \frac{2}{OPT(\mathcal{I})}$ which is at most 2.5 unless $OPT(\mathcal{I}) \leq 3$. We will first handle the exception where $OPT(\mathcal{I}) = 3$ by making a slight adaption: The algorithm starts with the first three iterations of the deterministic procedure. In the fourth iteration it queries a vertex in $\{u_4, v_4\}$ with probability 0.5 (or probability 1 if $u_4 = v_4$), then queries the other vertex if necessary and finally proceeds with the deterministic procedure if still no longest edge can be identified. See Algorithm 2 for a precise description of the algorithm *RAND*3.

Algorithm 2. The subroutine *RAND*3 of V-RANDOM$_C$

input : An instance $\mathcal{I}$ of V-MST-U with uniform query costs with a cycle $C = (V, E)$ and uncertainty sets A_v for $v \in V$
output: A feasible query set Q

```
Compute the associated edge instance;
Initialize Q = ∅;
for i ← 1 to 3 do
    if no edge in C is always maximal then
        Let g be an undominated edge with largest upper limit in C;
        Query the end vertices of g and add them to Q
if no edge in C is always maximal then
    Let g be an undominated edge with largest upper limit in C and let u, v be the non-trivial end vertices of g;
    Pick b ∈ [0, 1] uniformly at random;
    if b ≤ 0.5 then
        Query u and add it to Q;
        if no edge in C is always maximal then
            Query v and add it to Q
    else
        Query v and add it to Q;
        if no edge in C is always maximal then
            Query u and add it to Q
while no edge in C is always maximal do
    Let g be an undominated edge with largest upper limit in C;
    Query the end vertices of g and add them to Q
```

Lemma 10. *Let $\mathcal{I}$ be an instance of V-MST-U on a cycle C with uniform query cost which has a lower and upper limit tree T_L. Let ALG be an algorithm which queries the end vertices of the edge $f_1 \in C - T_L$ and applies RAND3 to the resulting uncertainty graph. Then ALG achieves competitive ratio 2.5 when applied to $\mathcal{I}$ unless $OPT(\mathcal{I}) \leq 2$ and $f_1 \cap Q^* = \emptyset$ for all optimal query sets Q^*.*

Proof. Again we denote by $\{u_i, v_i\}$ the set of vertices that *ALG* queries during the i'th step (i.e., $\{u_1, v_1\} = f_1$). During the first three steps the algorithm is identical to the deterministic procedure. The vertices $\{u_4, v_4\}$ are queried one after the other rather than simultaneously. If querying the first vertex is sufficient to find an always maximal edge the algorithm stops. Otherwise, it queries the second vertex immediately. Note that this is done prior to identifying a new edge g with largest upper limit. This makes sure that the same vertices are queried by *ALG* as by the deterministic procedure. The algorithm *ALG* thus makes at most as many queries as the deterministic procedure.

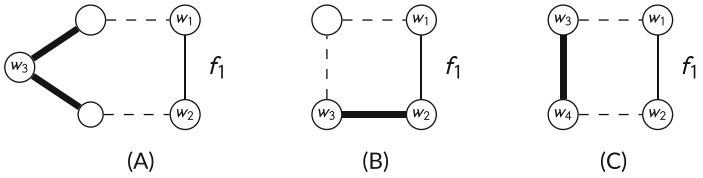

FIGURE 3 Sketches of cycles with $a = 1$ where edges in $X(f_1)$ are bold and dashed lines indicate parts of the cycle that are not in $X(f_1) \cup f_1$. In (A) and (B) RAND1 picks a vertex in $\{w_1, w_2, w_3\}$ with probability $\frac{1}{3}$ each. In (C) RAND1 picks a vertex in $\{w_1, w_2, w_3, w_4\}$ with probability $\frac{1}{4}$ each.

Hence, if $Q^* \cap f_1 \neq \emptyset$ or $OPT(\mathcal{I}) \geq 4$ then the claim follows from Lemma 9. Assume now that $Q^* \cap f_1 = \emptyset$ and $OPT(\mathcal{I}) = 3$. If the algorithm finishes after the first three iterations then it has made at most 6 queries and yields a competitive ratio of at most 2. If the algorithm has not finished after the first three iterations, then we know from the claim in the proof of Lemma 9 that two of the three sets $\{u_1, v_1\}$ $\{u_2, v_2\}$ and $\{u_3, v_3\}$ intersect with Q^* such that $\{u_4, v_4\}$ contains the last vertex in Q^*. Hence, we expect to make $0.5 \cdot (7 + 8)$ queries which yields a competitive ratio of at most 2.5. ∎

Following from Lemma 10, we only need to deal with instances where $OPT(\mathcal{I}) \leq 2$ and $f_1 \cap Q^* = \emptyset$. Thus we first check whether the uncertainty graph (after preprocessing but prior to any further queries) contains candidates for a feasible query set W of size at most 2 that does not contain an end vertex of f_1. We do so by computing the size a of a smallest vertex cover of $X(f_1)$ which does not intersect f_1 and consists of non-trivial vertices only. Note that if there is a feasible query set of size at most 2 which does not intersect the edge f_1 then $a \leq 2$ by Lemma 8.

If $a = 1$ or $a = 2$, we apply the randomized procedure *RAND*1 or *RAND*2 respectively, as described below.

Case a = 1 : If $a = 1$ then $X(f_1)$ either consists of a single edge e or of two edges g and h which are incident to each other (see Figure 3). The algorithm picks an order in which to query the three (or four) vertices in $(g \cap h) \cup f_1$ or $e \cup f_1$ uniformly at random and queries the vertices in the respective order up to the point where a longest edge in C can be identified. If necessary, it queries the remaining vertices in $X(f_1)$. See Algorithm 3 for a precise description.

Algorithm 3. The subroutine *RAND*1 of V-RANDOM$_C$

input : An instance $\mathcal{I}$ of V-MST-U with uniform query costs with a cycle $C = (V, E)$ and uncertainty sets A_v, $v \in V$
output: A feasible query set Q

```
Compute the associated edge instance;
Initialize Q = ∅;
Let f_1 be a undominated edge with largest upper limit and let X(f_1) be the neighbor set of f_1;
if no edge in C is always maximal then
    index the vertices v_1, ..., v_k in ⋂ X(f_1) ∪ f_1 arbitrarily;
    index the permutations σ_1, ..., σ_{k!} in S_k arbitrarily;
    draw b uniformly at random from [0, 1] and pick permutation σ = σ_i if b ∈ [ (i−1)/k! , i/k! );
    j := 1;
    while no edge in C is always maximal and j≤k do
        Query v_{σ(j)} and add it to Q;
        j := j + 1;
    if no edge in C is always maximal then
        Query the remaining vertices in X(f_1)
```

Case a = 2 : The algorithm starts by querying the end vertices of f_1. Then the algorithm picks a vertex cover W of $X(f_1)$ which contains $|W| = 2$ non-trivial vertices such that $W \cap f_1 = \emptyset$ uniformly at random. The algorithm queries the vertices in W. After that it queries the end vertices of undominated edges in order of decreasing upper limit. For a precise description, see Algorithm 4.

Algorithm 4. The subroutine *RAND*2 of V-RANDOM$_C$

input : An instance $\mathcal{I}$ of V-MST-U with uniform query costs with a cycle $C = (V, E)$ and uncertainty sets A_v, $v \in V$
output: A feasible query set Q

Compute the associated edge instance and T_L;
Initialize $Q = \emptyset$;
Let f_1 be the edge in $E - T_L$ and let $X(f_1)$ be the set of neighbors of $X(f_1)$;
Query f_1 and add its end vertices to Q;
if *no edge in C is always maximal* **then**
  Compute $\mathcal{W} = \{W \subseteq V(C) - f_1 \mid$ W is a smallest vertex cover of $X(f_1)$ which consists of non-trivial vertices only$\}$ and index the elements $W_1, \dots, W_{|\mathcal{W}|}$ arbitrarily;
  draw b uniformly at random from $[0, 1]$ and pick a vertex cover $W = W_i$ if $b \in \left[\frac{i-1}{|\mathcal{W}|}, \frac{i}{|\mathcal{W}|}\right)$;
  query the vertices in W and add them to Q;
  while *no edge in C is always maximal* **do**
    Let g be an undominated edge with largest upper limit in C;
    query the end vertices of g and add them to Q

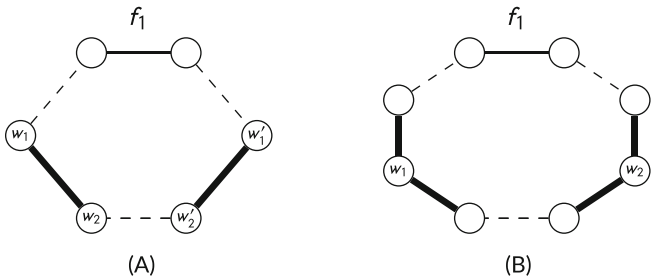

FIGURE 4 Sketches of cycles with $a = 2$ where edges in $X(f_1)$ are bold and dashed lines indicate parts of the cycle that are not in $X(f_1) \cup f_1$. In (A) RAND2 picks a cover $\{w_i, w_j'\}, i, j = 1, 2$ with probability $\frac{1}{4}$ each. In (B) RAND2 picks $\{w_1, w_2\}$ deterministically.

Lemma 11. *Let $\mathcal{I}$ be an instance of V-MST-U on a cycle C with uniform query cost which has a lower and upper limit tree T_L. Let f_1 be the edge in $C - T_L$ and denote by a the size of a smallest vertex cover of $X(f_1)$ which consists of non-trivial vertices only. If $a = 2$ then Algorithm 4 achieves competitive ratio 2.5 when applied to $\mathcal{I}$.*

Proof. As $|f_1 \cup V(X(f_1))| \leq 8$ (see Figure 4) we only need to discuss instances with $OPT(\mathcal{I}) = |Q^*| < 4$ where Q^* is an optimal query set. If $OPT(\mathcal{I}) = 1$ then $a = 2$ implies that $Q^* \subseteq f_1$ by Lemma 8 and the vertex in Q^* is queried within the first two queries. Say $OPT(\mathcal{I}) = 2$. As the queried vertex cover W has size $a = 2$, the claim trivially holds if $f_1 \cup W$ is a feasible query set. We thus only consider the case where querying $f_1 \cup W$ is not sufficient. We will distinguish the cases where Q^* either intersects f_1 or not. First assume that $Q^* \cap f_1 \neq \emptyset$. If $Q^* = f_1$ then the claim trivially holds. We thus assume that $|Q^* \cap f_1| = 1$. Let $g = \{v, w\}$ be an undominated edge with largest upper limit in the associated edge instance after having queried the end vertices of f_1 and the vertices in the vertex cover W. Say w is the vertex in $W \cap g$ and v is the non-trivial end vertex of g. We will now argue that if still no longest edge is found after querying vertices in f_1 and W, then the vertex $u \in Q^* - f_1$ is the unique non-trivial end vertex of g; that is, $u = v$. Assume to the contrary that $u \neq v$. By Lemma 6, g can still realize strictly longer than all edges which do not have always identical edge length with g after querying f_1 and W. Thus, if Q^* does not contain v, then we can only verify an MST T where either (a) $g \notin T$ or (b) $C - T = \{h\}$ where g and h have identical edge lengths. In both cases Q^* has to verify that g is a longest edge. Note that g is not incident to f_1 otherwise g would be trivial after querying f_1 and a vertex cover of $X(f_1)$. Hence, after querying f_1, we have that $w_{f_1} \in A_g$ due to the preprocessing and thus we need to query a vertex of g if we want to verify that g is not shorter than f_1. This proves that the vertex u in $Q - f_1$ is a vertex of g. As $W + f_1$ is no feasible query set, we have that $u \neq w$ and thus $u = v$, a contradiction. Hence, if $OPT(\mathcal{I}) = 2$ and $Q^* \cap f_1 \neq \emptyset$ we can verify an MST after querying the end vertices of f_1, a vertex cover of size 2 and an end vertex of g which proves a competitive ratio of at most 2.5.

Assume now that $Q^* \cap f_1 = \emptyset$. By Lemma 8, this implies that Q^* is a vertex cover of $X(f_1)$ (which verifies that f_1 is a longest edge in C). Say *RAND*2 picks a vertex cover which is not a feasible query set. During any of

the following steps, let g be an undominated edge which currently has largest upper limit in C. Then A_g contains w_{f_1} and thus Q^* contains the non-trivial vertex of g by Lemma 8. Note that we can have at most four different smallest vertex covers of $X(f_1)$ which do not intersect f_1 and at most two which are pairwise disjoint from each other (see Figure 4). Thus with probability at most $\frac{1}{4}$ we pick Q^* right away, with probability $\frac{1}{4}$ we pick a vertex cover that is disjoint from Q^* and need two additional queries while with probability $\frac{1}{2}$, we pick a vertex cover that intersects with Q^* in a single vertex after which we need to make one more query. This leads to a competitive ratio of $\frac{0.25\cdot 4+0.25\cdot 6+0.5\cdot 5}{2} = 2.5$.

To complete the proof, we need to discuss instances with $OPT(\mathcal{I}) = 3$. Note that the claim trivially holds if we have that $|f_1 \cup V(X(f_1))| \leq 7$ which is the case unless $X(f_1)$ consists of two paths which are neither incident to each other nor to f_1 (as in Figure 4B). Assume that the latter is the case and that we have queried the end vertices of f_1 and the vertices in the vertex cover W of $X(f_1)$. Thus, all remaining vertices in $X(f_1)$ have at most one non-trivial end vertex. As $V(C) - W - f_1$ contains no vertex cover of $X(f_1)$ of size 3, Lemma 8 guarantees that we have already queried one of the vertices in Q^* before we consider the edge g. Analogously to the proof of Lemma 9 for the deterministic procedure, we can argue that at least two out of the next three queries contain vertices in Q^* and thus we make at most seven queries in total which proves the claim for this case. ∎

4.1.4 | The algorithm V-RANDOM$_C$

In the following, we present the algorithm V-RANDOM$_C$ which solves the V-MST-U with unform query cost on cactus like graphs and analyze its performance. A precise description of the algorithm V-RANDOM$_C$ is displayed in Algorithm 5.

Algorithm 5. The algorithm V-RANDOM$_C$ for V-MST-U with uniform query costs in cactus-like graphs

input : An instance $\mathcal{I}$ of V-MST-U with uniform query costs, a graph $G = (V, E)$ and uncertainty sets A_v, $v \in V$ such that no two cycles share a non-trivial vertex
output: A feasible query set Q

```
Compute the associated edge instance;
Initialize Q = ∅;
Preprocess the instance such that T_L = T_U;
Index the edges f_1, ..., f_{m-n+1} in E-T_L arbitrarily;
for i ← 1 to m − n + 1 do
    Let C_i be the unique cycle in T_L + f_i;
    Let X(f_i) be the set of neighbors of f_i;
    Compute the size a of a smallest vertex cover of X(f_1) which does not intersect f_1 and consists of
    non-trivial vertices only;
    if no edge in C_i is always maximal then
        if a = 1 then
            Q = Q ∪ RAND1(C_i, {A_v | v ∈ V(C_i)})
        if a = 2 then
            Q = Q ∪ RAND2(C_i, {A_v | v ∈ V(C_i)})
        else
            Q = Q ∪ RAND3(C_i, {A_v | v ∈ V(C_i)})
```

Theorem 12. *For instances of V-MST-U with uniform query cost where no two cycles intersect in a non-trivial vertex the algorithm V-Random$_C$ achieves a competitive ratio of 2.5 and this is best possible.*

Proof. Let $\mathcal{I}$ be an instance with uncertainty graph $\mathcal{G} = (G, (A_v)_v)$ to which we apply V-RANDOM$_C$. For a cycle C in G we denote by Q^*_C an optimal query set for an instance of V-MST-U with graph C and uncertainty sets and vertex positions as in $\mathcal{I}$. If we prove for each such cycle C that V-RANDOM$_C$ achieves a competitive ratio of 2.5 when applied to C the claim follows because $Q^* = \dot{\bigcup}_C Q^*_C$ is an optimal query set for $\mathcal{I}$. Moreover, Lemma 7 guarantees that each vertex pair queried during the preprocessing intersects with Q^*. Thus we assume that $\mathcal{I}$ is an instance of V-MST-U with uncertainty graph $\mathcal{G} = (C, (A_v)_v)$ where C is a cycle and that we have a lower and

upper limit tree $T_L \subset C$. Let Q^* be an optimal query set for $\mathcal{I}$. Now we prove that V-RANDOM$_C$ queries at most $2.5 \cdot OPT(\mathcal{I})$ vertices when applied to $\mathcal{I}$. First we consider the case where $a \geq 3$. In this case the algorithm queries the end vertices of f_1 and applies *RAND*3 to the resulting uncertainty graph. Lemma 8 implies that $Q^* \cap f_1 \neq \emptyset$ or $OPT(\mathcal{I}) \geq a \geq 3$ and thus the claim follows from Lemma 10.

If we assume that $a = 2$ then the claim follows immediately from Lemma 11. Finally, assume that $a = 1$. Note that the maximum number of vertices in $f_1 \cup V(X(f_1))$ is 5. Thus we make at most five queries and only need to consider the case where $OPT(\mathcal{I}) = 1$. The neighbor set $X(f_1)$ consists either of a single edge e or of two intersecting edges g and h (see Figure 3). In the first case, each non-trivial vertex in $e \cup f_1$ could be a feasible query set of size one and we find the vertex in Q^* within at most $\frac{1}{4}(1+2+3+4) = 2.5$ or $\frac{1}{3}(1+2+3) = 2$ queries in expectation, depending on whether f and e are adjacent or not. In the second case, each non-trivial vertex in $(g \cap h) \cup f_1$ could be a feasible query set of size one and we find the vertex in Q^* within $\frac{1}{3}(1+2+3) = 2$ queries in expectation. Note that if $a = 0$ then f_i has no neighbors and is thus always maximal right away. Hence, V-RANDOM$_C$ queries at most $2.5 \cdot OPT(\mathcal{I})$ vertices which completes the analysis of the competitive ratio. Finally, Theorem 5 implies that the competitive ratio is best possible. ∎

V-RANDOM$_C$ runs in polynomial time under the reasonable assumption that $\sup\{d(u', v') | u' \in A_u, v' \in A_v\}$ and $\inf\{d(u', v') | u' \in A_u, v' \in A_v\}$ can be accessed in $\mathcal{O}(1)$ time for all vertices u, v and that for any two edges e and f we can determine in $\mathcal{O}(1)$ time whether e dominates f. While we omit a detailed analysis of the run-time complexity, we will briefly discuss the complexity of any operation where the run-time is not obvious. V-RANDOM$_C$ requires computing the size a of a smallest vertex cover $W \subseteq V(C) - f_1$ of $X(f_1)$ which consists of non-trivial vertices only. Note that there is a set U of vertices which has to be contained in any such vertex cover W: For each edge in $X(f_1)$ with an end vertex that is either trivial or lies in f_1, U contains the other end vertex respectively. Thus, finding such a W only involves finding a smallest vertex cover of the edges uncovered by U; that is, of $X(f_1) - \delta_{X(f_1)}(U) = \{e \in X(f_1) | e \cap U = \emptyset\}$. This can be done in polynomial time as $X(f_1)$ (and thus also $X(f_1) - \delta_{X(f_1)}(U)$) is a collection of paths. Hence, V-RANDOM$_C$ runs in polynomial time if all subroutines run in polynomial time.

Finally, *RAND*1, *RAND*2 and *RAND*3 are clearly polynomial except for the enumeration of permutations in *RAND*1 and the vertex cover computation in *RAND*2. *RAND*1 requires to list k permutations where k is the number of vertices in $X(f_1) \cup f_1$. However, note that we only execute *RAND*1 if $a = 1$ which means that $|X(f_1) \cup f_1|$ is bounded by 5. *RAND*2 involves the computation of *all* smallest vertex covers of $X(f_1) - \delta_{X(f_1)}(U)$ which can be done in polynomial time as *RAND*2 is only applied in the case where $a = 2$ and thus $|X(f_1)| \leq 4$ holds.

5 | MINIMUM SPANNING STARS

Natural specializations of MST-U arise when the optimization is restricted to spanning trees of a certain prespecified structure. In many cases, structural restrictions of this type lead to NP-hard problems in the classical optimization setting (without uncertainty), for example, in the case of Hamiltonian paths or spiders (see [11]). In the case of a spanning star the classical problem of finding a minimum weight spanning star (MSS) is trivial as there are at most n spanning stars in a graph on n vertices. In contrast, the explorable uncertainty version of the problem (MSS-U) is not trivial. More precisely, we show that unlike for MST-U there exists no algorithm for the MSS-U with a constant performance guarantee in terms of query cost.

Definition. The complete bipartite graph $K_{1,k}$ is called a *star* and is denoted by S_k. The vertex with degree k is called centre of S_k. If a graph G with n vertices contains S_{n-1} as a subgraph, then S_{n-1} is said to be a spanning star in G.

We define the *Minimum Spanning Star Problem under Explorable Uncertainty* (MSS-U) analogously to MST-U; that is, we want to find a set of edge queries of smallest weight which allows to identify a minimum spanning star. However, it turns out that no algorithm for MSS-U can achieve constant competitive ratio.

Theorem 13. *There exists no (deterministic or randomized) algorithm for MSS-U which achieves constant competitive ratio. This remains true even for instances with only two spanning stars and uniform query cost.*

Proof. We first prove the theorem for deterministic algorithms by defining for each $n \in \mathbb{N}_{>3}$ a finite family $(\mathcal{I}_n^j)_j$ of instances with uniform query cost on a graph G with n vertices such that every deterministic algorithm needs to make at least $(n-2)$ times as many queries as necessary on at least one instance in $(\mathcal{I}_n^j)_j$. For $n = 5$ the construction is illustrated in Figure 5.

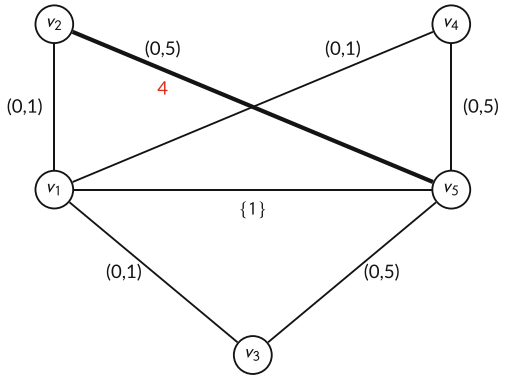

FIGURE 5 The instance $\mathcal{I}_5^2$ from the family $(\mathcal{I}_5^j)_{j=2}^4$. The weight of $\{v_2, v_5\}$ is 4. All missing edge weights equal 0.5. An optimal solution only needs to query $\{v_2, v_5\}$ which has larger weight than the star with center v_1 can have. A deterministic algorithm can not distinguish between the edges $\{v_2, v_5\}, \{v_3, v_5\}$ and $\{v_4, v_5\}$ and might have to query all three of them.

Consider a graph $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and $E = \{\{v_1, v_i\} | i = 2, \dots, n\} \cup \{\{v_n, v_i\} | i = 1, \dots, n-1\}$. Note that G has precisely two spanning stars, one with centre v_1 and the other one with centre v_n. (All other vertices have degree 2.) Let $q_e = 1$ for all $e \in E$. We define the uncertainty sets as follows:

- $A_{\{v_1, v_n\}} = \{1\}$,
- $A_{\{v_1, v_i\}} = (0, 1)$ for $i = 2, \dots, n-1$ and
- $A_{\{n, v_i\}} = (0, n)$ for $i = 2, \dots, n-1$.

For $j \in \{2, \dots, n-1\}$, $\mathcal{I}_n^j$ is defined such that $w_{\{v_j, v_n\}} = n-1$ and $w_e = 0.5$ for all $e \in E \setminus \{\{v_j, v_n\}, \{v_1, v_n\}\}$. For each instance $\mathcal{I}_n^j$, it is then sufficient to query only the edge $\{v_j, v_n\}$ to know that the spanning star with center v_n has larger weight. Conversely, without querying $\{v_j, v_n\}$, it is impossible to tell which spanning star has minimum weight. Note that all non-trivial edges adjacent to v_n are undistinguishable with respect to the uncertainty sets. Thus, a deterministic algorithm has to decide on an order in which these edges are queried. If j is such that $\{v_j, v_n\}$ is the last edge adjacent to v_n to be queried by an algorithm, then the algorithm's competitive ratio is at least $n-2$ when applied to instance $\mathcal{I}_n^j$.

To prove the claim for randomized algorithms, we apply again Yao's Principle. For $n \in \mathbb{N}_{>3}$, we prove that no randomized algorithm for MSS-U achieves a competitive ratio below $\frac{(n-1)}{2}$. Consider the randomized family of instances $(\mathcal{I}_n, p_n)$ where $\mathcal{I}_n = (\mathcal{I}_n^j)_{j=2}^{n-1}$ is defined as for the deterministic case and $p_n[\mathcal{I} = \mathcal{I}_n^j] = \frac{1}{n-2}$ for $j = 2, \dots, n-1$. Then, no deterministic algorithm ALG achieves a better expected competitive ratio $\mathbb{E}_{\mathcal{I} \sim_{p_n} \mathcal{I}_n}\left(\frac{ALG(\mathcal{I})}{OPT(\mathcal{I})}\right)$ than the algorithm ALG_1 which queries $\{v_2, v_n\}, \{v_3, v_n\}, \dots, \{v_{n-1}, v_n\}$ (or less edges if an MSS can already be identified) in this order independently from the queries' results. This is due to the fact that querying $\{v_j, v_n\}$ for $j \in \{2, \dots, n-1\}$ only reveals whether or not we are facing instance $\mathcal{I}_n^j$ but if not, it is indistinguishable which of the remaining instances it might be. Then, by Yao's Principle, no randomized algorithm has a better performance than

$$\min_{ALG \in \mathcal{A}} \mathbb{E}_{\mathcal{I} \sim_{p_n} \mathcal{I}_n}\left(\frac{ALG(\mathcal{I})}{OPT(\mathcal{I})}\right) = \mathbb{E}_{\mathcal{I} \sim_{p_n} \mathcal{I}_n}\left(\frac{ALG_1(\mathcal{I})}{OPT(\mathcal{I})}\right) = \frac{1}{n-2} \cdot \sum_{j=2}^{n-1} ALG_1(\mathcal{I}_n^j)$$
$$= \frac{1}{n-2} \cdot \sum_{j=2}^{n-1} (j-1) = \frac{n-1}{2},$$

where $\mathcal{A}$ denotes the class of all deterministic algorithms. ∎

6 | CONCLUSION

In this paper we considered the Minimum Spanning Tree Problem Under Explorable Uncertainty (MST-U) and the related Minimum Spanning Tree Problem Under Explorable Vertex Uncertainty (V-MST-U) for specified instance types where cycles can be considered independently from each other. We provided a randomized algorithm for MST-U on cactus graphs and proved that it achieves a competitive ratio of 1.5 which is best possible. For V-MST-U instances where

cycles do not share non-trivial vertices and query costs are uniform, we provided the algorithm V-RANDOM$_C$ which achieves a competitive ratio of 2.5. We showed that 2.5 is a lower bound for the performance of randomized algorithms for V-MST-U which remains true even for instances with uniform query cost where the uncertainty graph is a cycle. Thus, the performance guarantee shown for V-RANDOM$_C$ is best possible. Finally, we introduced the Minimum Spanning Star Problem under Explorable Uncertainty (MSS-U) and proved that no algorithm for MSS-U can achieve constant competitive ratio.

(V-)MST-U in itself is a problem which still deserves further investigation. A major open question in the setting of MST-U is whether there exists a randomized algorithm with a competitive ratio of 1.5 for general instances as well or whether the lower bound of 1.5 can be improved for instances where the graph is not necessarily a cactus.

As for general instances of V-MST-U, no randomized algorithm which improves upon the performance of deterministic algorithms is known so far and even for the special case where cycles intersect in trivial vertices only, the performance guarantee of V-RANDOM$_C$ relies on the fact that query costs are uniform. Note that no deterministic algorithm with constant competitive ratio for V-MST-U with non-uniform query costs is known either.

Moreover, different models of uncertainty exploration could be subject of further research. Consider for instance a scenario where installing a camera in a certain location allows to measure the distance between itself and all surrounding objects. A setting like this could motivate a hybrid model between edge and vertex uncertainty where edge weights are uncertain but known to lie inside given uncertainty sets and can be revealed upon querying an adjacent vertex.

ACKNOWLEDGEMENT

Open Access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Corinna Mathwieser https://orcid.org/0000-0002-5983-2616
Eranda Çela https://orcid.org/0000-0002-5099-8804

REFERENCES

[1] E. Bampis, C. Dürr, T. W. Erlebach, M. S. de Lima, N. Megow, and J. Schloter. Orienting (hyper)graphs under explorable stochastic uncertainty. Paper presented at: 29th annual European symposium on algorithms (ESA 2021). 2021 10:1–10:18.
[2] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, Cambridge, 1998.
[3] S. Chaplick, M. M. Halldórsson, M. S. de Lima, and T. Tonoyan, *Query minimization under stochastic uncertainty*, Theor. Comput. Sci. **895** (2021), 75–95.
[4] C. Dürr, T. Erlebach, N. Megow, and J. Meißner. Scheduling with Explorable uncertainty, 9th innovations in theoretical computer science conference, Vol. 94 of Leibniz international proceedings in informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 2018 30:1–30:14.
[5] T. Erlebach, M. S. de Lima, N. Megow, and J. Schloter. Learning-augmented query policies for minimum spanning tree with uncertainty. Paper presented at: 30th annual European symposium on algorithms (ESA 2022). 2022 49:1–49:18.
[6] T. Erlebach and M. Hoffmann. Minimum spanning tree verification under uncertainty, Proceedings of the international workshop on graph-theoretic concepts in computer, Science. 2014 164–175.
[7] T. Erlebach and M. Hoffmann, *Query-competitive algorithms for computing with uncertainty*, Bulletin Eur. Assoc. Theor. Comput. Sci. **116** (2015), 21-39.
[8] T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihal'ák, and R. Raman, C*omputing minimum spanning trees with uncertainty*, Proc. Symp. Theor. Aspects Comput. Sci. **1** (2008), 277–288.
[9] T. Feder, R. Motwani, L. O'Callaghan, C. Olston, and R. Panigrahy, *Computing shortest paths with uncertainty*, J. Algorithms **62** (2007), 1–18.
[10] J. Focke, N. Megow, and J. Meißner. Minimum spanning tree under Explorable uncertainty in theory and experiments, 16th international symposium on experimental algorithms, Vol. 75 of Leibniz international proceedings in informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 2017 22:1–22:14.
[11] L. Gargano, M. Hammar, P. Hell, L. Stacho, and U. Vaccaroa, *Spanning spiders and light-splitting switches*, Discret. Math. **285** (2004), 83–95.
[12] M. Goerigk, M. Gupta, J. Ide, A. Schöbel, and S. Sen, *The robust knapsack problem with queries*, Comput. Oper. Res. **55** (2015), 12–22.
[13] S. Kahan. A model for data in motion, 23rd Ann, ACM Symp. Theory Comput. STOC'91. 1991 267–277.
[14] S. Khanna and W. C. Tan. On computing functions with uncertainty, Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. 2001 171–182.
[15] N. Megow, J. Meißner, and M. Skutella, *Randomization helps computing a minimum spanning tree under uncertainty*, SIAM J. Comput. **46** (2017), 1217–1240.

[16] A. I. Merino and J. A. Soto. The minimum cost query problem on Matroids with uncertainty areas, 46th international colloquium on automata, languages, and programming (ICALP 2019), Vol. 132 of Leibniz international proceedings in informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany. 2019 83:1–83:14.

How to cite this article: C. Mathwieser and E. Çela, *Special cases of the minimum spanning tree problem under explorable edge and vertex uncertainty*, Networks.. (2024), 1–18. https://doi.org/10.1002/net.22204