Query-Competitive Algorithms for Cheapest Set Problems under Uncertainty

Thomas Erlebach[1], Michael Hoffmann[1], and Frank Kammer[2]

[1] Department of Computer Science, University of Leicester, England
{te17,mh55}@mcs.le.ac.uk
[2] Institut für Informatik, Universität Augsburg, Germany
kammer@informatik.uni-augsburg.de

Abstract. Considering the model of computing under uncertainty where element weights are uncertain but can be obtained at a cost by query operations, we study the problem of identifying a cheapest (minimum-weight) set among a given collection of feasible sets using a minimum number of queries of element weights. For the general case we present an algorithm that makes at most $d \cdot OPT + d$ queries, where d is the maximum cardinality of any given set and OPT is the optimal number of queries needed to identify a cheapest set. For the minimum multi-cut problem in trees with d terminal pairs, we give an algorithm that makes at most $d \cdot OPT + 1$ queries. For the problem of computing a minimum-weight base of a given matroid, we give an algorithm that makes at most $2 \cdot OPT$ queries, generalizing a known result for the minimum spanning tree problem. For each of our algorithms we give matching lower bounds.

1 Introduction

Motivated by applications where exact input data is not always easily available, we consider cheapest set problems under uncertainty: We are given a set E of elements, and a collection $\mathcal{S}$ of *feasible* subsets of E, where $\mathcal{S}$ may be specified explicitly or implicitly. Each element $e \in E$ has an exact weight (or cost) w_e, but initially only an *uncertainty area* A_e, which is a set that contains w_e, is known. We assume that each uncertainty area A_e is either trivial (i.e., a singleton set containing only w_e) or an open set with finite lower limit L_e and finite upper limit U_e (for example, an open interval (L_e, U_e)). The task is to find a cheapest set in $\mathcal{S}$, i.e., a set $S \in \mathcal{S}$ such that $\sum_{e \in S} w_e$ is minimized. It may not be possible to identify a cheapest set based on just the given uncertainty areas. We assume that it is possible to obtain the exact weight w_e of an element $e \in E$ using a *query* operation, but we wish to minimize the number of queries needed.

An algorithm solving the cheapest set problem under uncertainty may make more queries than absolutely necessary. To assess the quality of an algorithm, we use competitive analysis, i.e., for the given instance of the cheapest set problem we compare the number of queries the algorithm makes with the best possible number of queries, which we denote by OPT. An algorithm for a problem under uncertainty that is measured competitively with respect to the number of queries is also called a *query-competitive* algorithm. We restrict the uncertainty areas to

E. Csuhaj-Varjú et al. (Eds.): MFCS 2014, Part II, LNCS 8635, pp. 263–274, 2014.

be open sets or singleton sets because it is easy to see (as shown in [5] for the minimum spanning tree problem) that there are no query-competitive algorithms with non-trivial competitive ratio for the uncertainty problems that we consider if closed intervals are allowed as uncertainty areas.

We consider the cheapest set problem under uncertainty both in the general case, where the feasible sets can be arbitrary and are specified explicitly as part of the input, and in special cases that arise when the feasible sets have a certain structure. In the multi-cut problem for trees, the feasible sets are the sets of edges that separate the given terminal pairs. In the minimum matroid base problem, the feasible sets are the bases of a matroid. The minimum spanning tree problem is a special case of the minimum matroid base problem where the independent sets of the matroid are the spanning forests of the given graph.

Motivation for studying the cheapest set problem under uncertainty can be found in numerous application areas. Many optimization problems can be viewed as the problem of selecting a minimum-weight set among all feasible sets. Especially in distributed networks or mobile computing, it is often the case that the exact weight of an element is known only approximately (e.g., an estimate for the cost of a remote service or congestion of a remote link), but it may be possible to obtain the exact weight at an extra cost (e.g., a negotiation with a service provider, or a query message and response exchanged over the network). If the cost for obtaining the exact weight of an input element by a query is not negligible, the objective of minimizing the number of queries needed to solve the problem becomes natural. For example, consider the problem of installing monitoring equipment on links of a tree network so as to monitor all traffic between d given terminal pairs. The cost of the installation on a specific link depends on the total traffic on that link (generated by the terminal pairs and by background traffic), as all packets traversing the link need to be processed. The exact cost of a link can be determined by conducting traffic measurements, but this may be costly. The problem of identifying a set of edges of minimum total cost for solving the monitoring problem, while making a minimum number of traffic measurements on different links, is the multi-cut problem in trees under uncertainty.

Our Results. For the cheapest set problem under uncertainty, we give an algorithm that makes at most $d \cdot OPT + d$ queries, where d is the maximum cardinality of a feasible set in the given instance. We also give a matching lower bound, showing that the algorithm is best possible among deterministic algorithms. For the minimum multi-cut problem for d terminal pairs in trees under uncertainty, we give an algorithm that makes at most $d \cdot OPT + 1$ queries, and we prove a matching lower bound. For the minimum matroid base problem under uncertainty, we give an algorithm that makes at most $2 \cdot OPT$ queries, generalizing a known 2-competitive algorithm U-RED for minimum spanning trees under uncertainty [5]. We remark that the generalisation is not straightforward since in [5] properties of connected components are considered while the matroid setting requires set oriented proofs. The known lower bound for minimum spanning trees under uncertainty [5] implies that our algorithm for minimum matroid base is best possible. Some proofs from Sections 4 to 6 are omitted.

Related Work. The first study of query-competitive algorithms for problems under uncertainty that we are aware of is the work by Kahan [9], who gives query-competitive algorithms with optimal competitive ratio for the problems of computing the maximum, the median and the minimum gap of n real values that are constrained to fall into given real intervals. Bruce et al. [2] consider geometric problems where input points are not known exactly but lie in given uncertainty areas. They propose the concept of witness set algorithms that, in each step, query a set S of elements with the property that any query solution must query at least one element of S. Sets with this property are called witness sets. The competitive ratio of a witness set algorithm is bounded by the maximum size of a witness set. Bruce et al. present 3-competitive algorithms for computing maximal points or the points on the convex hull of a given set of uncertain points in two-dimensional space [2]. Erlebach et al. [5] give a 2-competitive algorithm for computing minimum spanning trees in graphs with uncertain edge weights and show that this is optimal for deterministic algorithms.

Feder et al. study the problem of minimizing the total cost of queries for the problem of computing an approximation of the value of the median of n uncertain values [7] or of the length of a shortest path in a graph with uncertain edge weights [6]. They also consider algorithms that must specify the whole set of queries in advance, rather than querying uncertain elements one by one as in our model. Other related work exploring trade-offs between query cost and solution accuracy includes [13] and [10]. Another line of work considers the problem of computing, for an input with uncertain elements, the minimum and maximum possible cost of an optimal solution, over all possible precise values of the input [12,4]. In these problems, there is no concept of queries. There are also numerous other models of optimization under uncertainty, e.g., min-max regret versions of standard optimization problems [1].

For the standard version of the multi-cut problem in trees, Garg et al. [8] show that the problem is $\mathcal{NP}$-hard and MAX SNP-hard and admits a 2-approximation (with respect to the cost of the multi-cut). The standard version of the minimum matroid base problem can be solved by a greedy algorithm, see e.g. [3]. We conclude in Section 7.

2 Preliminaries

Based on the terminology of Section 1 we now formally define the cheapest set problem under uncertainty (CSU). An instance of the CSU problem is represented by a quadruple $(E, \mathcal{S}, w, A)$ where E is the finite set of elements, $\mathcal{S} \subseteq \mathcal{P}(E)$ is a family of subsets of E, w is a real-valued weight function that maps each element $e \in E$ to its precise weight $w(e)$, and A is a function that maps each element $e \in E$ to its uncertainty area $A(e)$. From here on, we write w_e and A_e for $w(e)$ and $A(e)$, respectively. Note that for the CSU problem we allow w_e to be smaller than 0. The goal of the CSU problem is to find a *cheapest set* S, i.e., a set $S \in \mathcal{S}$ such that $\sum_{e \in S} w_e \leq \sum_{e \in S'} w_e$ for all $S' \in \mathcal{S}$, using a minimum number of queries. Let w' be a function that maps elements to weights

and let A' be a function that maps elements to uncertainty areas. We say that w' is *consistent with* A' if $w'_e \in A'_e$ for all $e \in E$. Note that in any instance $(E, \mathcal{S}, w, A)$ of the CSU problem w is consistent with A by definition. For $e \in E$, we refer to all elements of A_e as *potential weights* of e. An instance $(E, \mathcal{S}, w, A)$ is *solved* if there exists a set $S \in \mathcal{S}$ such that for every weight function w' that is consistent with A, S is a cheapest set in the instance $(E, \mathcal{S}, w', \mathcal{A})$. If an instance is solved and S satisfies the condition stated in the previous sentence, we also say that S can be *identified* as cheapest set. An instance that is not solved is also called *unsolved*. A *query* of an element e changes an instance $(E, \mathcal{S}, w, A)$ to $(E, \mathcal{S}, w, A')$ where A' is the same as A apart from A_e being set to $\{w_e\}$.

We say that an element whose area of uncertainty is trivial (i.e., a singleton set) is a *certain* element while an element whose area of uncertainty is non-trivial is an *uncertain* element. For $X \subseteq E$, we denote by X^U the set of its uncertain elements and by X^C the set of its certain elements.

For every instance $(E, \mathcal{S}, w, A)$, a *query solution* is a set of elements that when queried results in a solved instance. A query solution of minimum cardinality is called an *optimal query solution*, and the size of an optimal query solution is denoted by OPT. Throughout this paper, we consider only instances where each area of uncertainty is trivial or a (bounded) open set. For an element $e \in E$, we write L_e for the lower limit and U_e for the upper limit of the uncertainty area of e. If e has a trivial uncertainty area, then $L_e = U_e = w_e$. For any set $T \subseteq E$, we let $T_{min} = \sum_{e \in T} L_e$. Note that T_{min} is a lower bound on the exact weight of the set T. T_{max} is defined analogously via U_e instead of L_e. Furthermore, we sometimes write T_w for $\sum_{e \in T} w_e$. We say that $T \in \mathcal{S}$ *lies within the uncertainty bounds* of $S \in \mathcal{S}$ if for any potential weights of the elements in T there exist potential weights of the elements in $S \setminus T$ such that S is cheaper than T and (other) potential weights of the elements in $S \setminus T$ such that T is cheaper than S. Note explicitly that the definition implies $T \neq S$.

For an instance $(E, \mathcal{S}, w, A)$ the input of an algorithm is $(E, \mathcal{S}, A)$. The aim is to make queries until the resulting instance is solved. The quality of an algorithm is measured by the number of queries it makes compared to OPT of the initial instance. As introduced in [2], for a given instance a set of elements is a *witness set* if the instance cannot be solved without querying at least one element of the set. Let I' be an instance that has resulted from querying some elements in an instance I. Then a witness set of I' is also a witness set of I. Hence, we have the following observation, which can be shown using similar arguments as in [2]:

Observation 1. If an instance I is solved by an algorithm that queries all elements of all sets $U_1, U_2, \ldots, U_l$ where U_i is a witness set of the instance resulting from I after querying $U_1, \ldots U_{i-1}$, then the value OPT for the instance I is at least l. If only k of the l sets U_i are witness sets, then the value OPT for the instance I is at least k.

Definition 2. *For an instance $I = (E, \mathcal{S}, w, A)$, a set $S \in \mathcal{S}$ is called a* Robust Potential Cheapest (RPC) set*, if for any potential weights of elements in $E \setminus S$,*

there exist potential weights for the elements in S such that S is a cheapest set among all sets in $\mathcal{S}$.

We also refer to an RPC set of an instance $(E, \mathcal{S}, w, A)$ as an *RPC set in* $\mathcal{S}$ if the instance is clear from the context. The following observation is a direct consequence of the definition. The next lemma is needed to prove Lemma 5.

Observation 3. Let S be an RPC set of an instance I. Then we have that $S_{min} = \min\{X_{min} \mid X \in \mathcal{S}\}$. Further, if S contains only certain elements, I is solved and S is a cheapest set for the instance I.

Lemma 4. *A set S is an RPC set if, for all $X \in \mathcal{S}\backslash\{S\}$, either 1. or 2. holds:*
1. $S_{min} < X_{min}$, or
2. $S_{min} = X_{min}$ and $X^U \not\subset S^U$, i.e., X^U is not a proper subset of S^U.

Proof. Let w' denote an arbitrary choice of potential values for $E \setminus S$.

Consider any $X \in \mathcal{S}$ for which Condition 1 holds. As the potential values for S can be chosen arbitrarily close to their lower limits (since the uncertainty areas are open sets), there exist potential values w_e^X for $e \in S$ such that S is not more expensive than X_{min} and thus not more expensive than $X_{w'}$.

Consider any $X \in \mathcal{S}$ for which Condition 2 holds. If $X^U = S^U$, the weights of X and S are equal (and hence S is not more expensive than X) for all potential values for $e \in S$. Therefore, assume that $X^U \neq S^U$. As X^U is not a proper subset of S^U, there must exist an element e^* in $X^U \setminus S^U = (X \setminus S)^U$. Let $S' = S \setminus X$ and $X' = X \setminus S$. Note that $e^* \in (X')^U$. By Condition 2, we also have $S'_{min} + (S \cap X)_{min} = S_{min} = X_{min} = X'_{min} + (S \cap X)_{min}$, and hence $S'_{min} = X'_{min}$. As $w'_{e^*} > L_{e^*}$ and $e^* \in X'$, we can choose values w_e^X for the elements $e \in S'$ such that $S'_{w^X} < X'_{w'}$. For any choice of values w_e^X for the elements of $S \cap X$, we then have $S_{w^X} < X_{w'}$.

Combining the arguments from the two previous paragraphs, we have that for each set $X \in \mathcal{S}\backslash\{S\}$, there exist values w_e^X for $e \in S$ such that S is not more expensive than X. If we now choose $w_e^* = \min_{X \in \mathcal{S}\backslash\{S\}} w_e^X$, the weights w_e^* for $e \in S$ are such that S is not more expensive than any of the sets in $\mathcal{S} \setminus \{S\}$. As the choice of w' was arbitrary, we have shown that S is an RPC set. □

Lemma 5. *Every instance of the CSU problem has at least one RPC set.*

Proof. We find an RPC set as follows: Initially, we choose any S such that $S_{min} \leq X_{min}$ for all $X \in \mathcal{S}$. As long as there exists a set $X \in \mathcal{S} \setminus \{S\}$ with $X_{min} = S_{min}$ and $X^U \subset S^U$, set $S := X$. This process must terminate as $|S^U|$ decreases in each step. In the end, S will be an RPC set by Lemma 4. □

Lemma 6. *An instance is solved if and only if all cheapest sets are identified. Moreover, in such an instance, $S_{min} = X_{min}$ and $S^U = X^U$ for any two cheapest sets S and X.*

Proof. As the instance is solved, there is at least one set S that can be identified as a cheapest set. Let X be another cheapest set. Assume that S contains an

uncertain element e that is not in X. As all non-trivial areas of uncertainty are open, there is a potential value for e that is higher than w_e. So S_w has the potential to increase while X_w remains the same, and potentially $X_w < S_w$. A contradiction to S_w being identified as a cheapest set.

Similarly, assume that X contains an uncertain element e that is not in S. As all non-trivial areas of uncertainty are open there is a potential value for e that is lower than w_e. So X_w has the potential to decrease while S_w remains the same, and potentially $X_w < S_w$. A contradiction.

Consequently, $X^U = S^U$. Since $S_w = X_w$, the sum of all certain elements in S and in X must also be the same, and $S_{min} = X_{min}$. Hence, if S is identified as a cheapest set, then also X must be identified as a cheapest set. □

Lemma 7. *Let $I = (E, \mathcal{S}, w, A)$ be an instance of the CSU problem. If S is an RPC set but not a witness set of I, then S is a cheapest set and $X^U = S^U$ for each cheapest set X of I.*

Proof. Let I' be the instance obtained after querying $E \setminus S$ in I. As S is not a witness set, the instance I' must be solved. Since S was an RPC set in I and no element of S was queried, S is also an RPC set in I' and hence S is potentially a cheapest set in I'. As I' is solved, by Lemma 6 we have that S must be identified either as being a cheapest set or as not being a cheapest set in I'. As S is potentially a cheapest set in I', it can only be the case that S is identified as a cheapest set in I'. Hence, S is a cheapest set of I.

Let X be another cheapest set in I, and therefore also in I'. As I' is solved, Lemma 6 implies that $X_{min} = S_{min}$ and $X^U = S^U$ in I'. Assume that an element of X was queried by the query of $E \setminus S$. Then X_{min} in I must be smaller than X_{min} in I' as all non-trivial areas of uncertainty are open. Hence, we must have $X_{min} < S_{min}$ in I. This contradicts S being an RPC set in I. So no element of X was queried by $E \setminus S$, and we have that $X^U = S^U$ also in I. □

3 Cheapest Set

We denote the maximum number of elements in any set of $\mathcal{S}$ by d. Let us define algorithm SIMPLE for the CSU problem as an algorithm that, while the instance is not solved, queries all elements of an RPC set. First, we need the next lemma.

Lemma 8. *Let $S \in \mathcal{S}$ be an RPC set in $\mathcal{S}$ and let $T \in \mathcal{S}$ be such that T lies within the uncertainty bounds of S. Then S is a witness set.*

Proof. Assume that S is not a witness set. Then S is a cheapest set by Lemma 7, and $E \setminus S$ is a query solution. Let I' be the instance after querying $E \setminus S$. As no element of S has been queried, T still lies within the uncertainty bounds of S. Hence T is still potentially cheaper than S and S cannot be identified in I' as a cheapest set. So by Lemma 6, I' cannot be a solved instance. This is a contradiction to $E \setminus S$ being a query solution. So, S must be a witness set. □

Theorem 9. *Algorithm SIMPLE makes at most $d \cdot OPT + d$ queries.*

Proof. Let t be the number of sets queried by the algorithm. For $1 \leq i \leq t$, let S_i be the set queried by the algorithm in the ith iteration of the while-loop. We claim that there is at most one $i \in \{1, \ldots, t\}$ such that S_i is not a witness set in iteration i. Choose i as small as possible such that S_i is not a witness set in iteration i. (If no such i exists, all t sets are witness sets, and the claim holds.) By Lemma 7, S_i is a cheapest set in iteration i. Assume for contradiction that there exists $j > i$ such that S_j is not a witness set. Again by Lemma 7, S_j is also a cheapest set and $S_i^U = S_j^U$ before iteration i. As S_i^U was queried at iteration i, at iteration j the set S_j does not contain uncertain elements. By Observation 3, the instance is solved before iteration j. A contradiction.

The algorithm queries t sets of which at least $t-1$ are witness sets. Hence, $OPT \geq t-1$ and the algorithm makes at most $dt \leq d \cdot OPT + d$ queries. □

In the following, we want to reduce the number of queries for variants of the CSU problem that satisfy a special property. The next lemma allows us to determine a witness set for any unsolved instance of the CSU problem.

Lemma 10. *Let $S, T \in \mathcal{S}$ such that S is an RPC set in $\mathcal{S}$ and T is potentially cheaper than S. Then $S \cup T$ is a witness set.*

Proof. Assume that $S \cup T$ is not a witness set. Then it is possible to solve the instance without querying any element of $S \cup T$. As $S \subseteq S \cup T$, S is not a witness set either. By Lemma 7, S is a cheapest set. If we do not query any element of $S \cup T$, T is potentially cheaper than S, and S cannot be identified as a cheapest set. So there is at least one cheapest set that cannot be identified. By Lemma 6, we cannot solve the instance, a contradiction. □

We say that a variant (special case) of the CSU problem has the 1-*gap property* if, for every unsolved instance $(E, \mathcal{S}, w, A)$ of that problem, there exist $S, T \in \mathcal{S}$ such that (1) S is an RPC set in $\mathcal{S}$, (2) T is potentially cheaper than S, and (3) $|S \cup T| \leq d+1$. We now show that the following algorithm U-SET for 1-gap CSU problems makes at most $d \cdot OPT + 1$ queries.

Algorithm 1. Algorithm U-SET for 1-gap CSU problems

1. **while** instance is not yet solved **do**
2. **if** there exist $S, T \in \mathcal{S}$ such that S is an RPC set and T lies within the uncertainty bounds of S **then** query S
3. **else** query all uncertain elements of $S \cup T$ where S is an RPC set and T is a potentially cheaper set than S such that $|S \cup T| \leq d+1$

Lemma 11. *When Step 3 in algorithm U-SET is executed any time except the first time, the set S must contain a certain element.*

Proof. The first situation that we consider is when Step 3 is executed for the first time. Let us call the RPC set X and the set that is potentially cheaper Y. The second situation we consider is when Step 3 is executed another time (second, third, and so on). This is after the first, so $X \cup Y$ has been queried and

the algorithm now takes an RPC set S. Let us assume that S consists only of uncertain elements. Then X and S must be disjoint. All elements of S had in the first situation the same uncertainty information as in the second situation, and S did not lie within the uncertainty bounds of X in the first situation. As X and S are disjoint and X was an RPC set, X could have been potentially cheaper than S for any weights of the elements of S in the first situation. So as S did not lie within the uncertainty bounds of X, there must exist some potential weight for the elements in S such that S cannot be cheaper than X. Since X and S are disjoint, for any potential weight of elements in X, the set S can be more expensive than X. In the second situation $X \cup Y$ has been queried and the weight of X is known precisely. So even for the queried set X, S can still be more expensive than X. Since S is an RPC set, S can also be cheaper than X. Thus, the queried set X lies with in the uncertainty bounds of S. Thus, in the second situation, the algorithm would execute Step 2 and not Step 3. A contradiction. Hence, our assumption that S does not contain any certain elements is false. □

Theorem 12. *For any uncertainty set problem with the 1-gap property, there exists an algorithm that makes at most $d \cdot OPT + 1$ queries.*

Proof. By Lemmas 8 and 10, all sets of queries performed by the algorithm are witness sets. By Lemma 11, the algorithm requests a set of queries of size $d+1$ at most once, and all other query sets are of size at most d. Hence, by Observation 1, the algorithm makes at most $d \cdot (OPT - 1) + d + 1$ queries. □

4 Minimum Multicut in Trees

We now consider the minimum multicut problem in trees under uncertainty (MMCTU). An instance of it is given by a tuple $(E, (G, D), w, A)$, where G is an undirected tree with edge set E, D is a set of d terminal pairs that need to be cut, w maps each edge $e \in E$ to its actual weight $w_e > 0$, and A maps each $e \in E$ to its uncertainty area A_e. The family $\mathcal{S}$ of feasible sets is not given explicitly, but is determined by G and D: A set $S \subseteq E$ is feasible if removing the edges in S from the tree G separates all terminal pairs in D. In other words, $\mathcal{S}$ is the family of all possible multicuts for the given terminal pairs. Multicuts containing more than d edges can be ignored because they must contain redundant edges, so we only need to consider multicuts consisting of at most d edges.

Let S be a potential minimum multicut. Then each element in S cuts at least one terminal pair that is not cut by any other element of S. By considering the elements of $S = \{s_1, \ldots, s_{|S|}\}$ one by one, a partition $P = \{P_1, P_2, \ldots, P_{|S|}\}$ of D is formed, where P_i is the set of terminal pairs that are cut by s_i and not by any $s_1, \ldots, s_{i-1}$. We say P is a *partition induced by* S. We also say that the element $s_i \in S$ *leads to* the element $P_i \in P$.

Lemma 13. *Let P and Q be two partitions of a finite set K. If, for all proper subsets P' of P and Q' of Q, $\bigcup_{X \in P'} X \neq \bigcup_{X \in Q'} X$, then $|P| + |Q| \leq |K| + 1$.*

Lemma 14. *The MMCTU problem has the 1-gap property.*

Proof. Let $I = (E, (G, D), w, A)$ be an unsolved instance of the MMCTU problem. Let S be an RPC set of I. Let $\mathcal{F}_S$ be the family of multicuts that are potentially cheaper than S. Let $T \in \mathcal{F}_S$ be such that $|T \backslash S| \leq |T' \backslash S|$ for all $T' \in \mathcal{F}_S$. Let D' be the set of pairs that are not cut by $S \cap T$. So, let P be a partition of D' induced by $S \backslash T$ and let Q be a partition of D' induced by $T \backslash S$.

Assume that there exist proper subsets P' of P and Q' of Q with $\bigcup_{X \in P'} X = \bigcup_{X \in Q'} X$. Let S' be the set of elements in S that lead to elements in P' and similarly T' be the set of elements in T that lead to elements in Q'. Note that the pairs of $D \backslash D'$ are cut by $S \backslash S'$ as well as by $T \backslash T'$. So the sets $Mix = S' \cup (T \backslash T')$ and $Mix' = T' \cup (S \backslash S')$ are also cuts of all pairs in D. Since

$$S'_{min} + (S \backslash S')_{min} = S_{min} \overset{S \text{ is an RPC set}}{\leq} Mix'_{min} = T'_{min} + (S \backslash S')_{min},$$

we have $S'_{min} \leq T'_{min}$. We now consider two cases.

Case 1. If $S'_{max} \leq T'_{min}$, then Mix is always cheaper than T. As T was potentially cheaper than S, Mix also must be potentially cheaper than S. Moreover, $|Mix \backslash S| = |(T \backslash S) \backslash T'| < |T \backslash S|$, which is a contradiction to our choice of T.

Case 2. If $S'_{max} > T'_{min}$, then S can be more expensive than Mix'. This means that Mix' is potentially cheaper than S, and $|Mix' \backslash S| = |T'| < |T \backslash S|$. The existence of Mix' contradicts our choice of T.

Thus, no proper subsets P' of P and Q' of Q as described above exist. Hence, by Lemma 13, $|S \Delta T| \leq |D'| + 1$. As $|S \cap T| \leq d - |D'|$ we get $|S \cup T| \leq d + 1$. □

Lemma 14 and Theorem 12 give us the following.

Theorem 15. *For the MMCTU problem, there exists an algorithm that makes at most $d \cdot OPT + 1$ queries.*

5 Minimum Matroid Base

We present an algorithm for the minimum matroid base under uncertainty (MMBU) problem using at most $2 \cdot OPT$ queries. We first recall the basic notation of matroids (see, e.g., [3]). A matroid $M = (E, I)$ consists of a set of elements E and a set of independent sets $I \subseteq \mathcal{P}(E)$ such that the following properties are satisfied.

Non-emptiness: $\emptyset \in I$,
Heredity: Every subset of a set in I is also in I,
Exchange: If $S, T \in I$ and $|S| < |T|$, then $\exists e \in T$ such that $S \cup \{e\} \in I$.

Each element has a real-valued weight, which may be negative. A subset of E that is not independent is called *dependent*. A *circuit* of M is a dependent set over E such that all its proper subsets are independent. A set $S \subseteq E$ is called a *base* of M if S is independent and for any $e \in E \setminus S$ the set $S \cup \{e\}$ is dependent. We write *circuit* (or *base*) *of M over $E' \subseteq E$* for a circuit (or base) of $(E', \{S \cap E' | S \in I\})$. When M is clear from the context, we write just *circuit* or *base over E'*. The following observation is well known.

Observation 16. Every base of a matroid M has the same number of elements.

Let M be a matroid with a weight function w that assigns each $e \in E$ a weight w_e. M is then called a *weighted matroid*. A *minimum base* of M is a base such that the sum of the weights of its elements is minimum among all bases of M. An instance of the MMBU problem is given by a tuple (E, I, w, A), where $M = (E, I)$ is a matroid, w maps each $e \in E$ to its (actual) weight w_e, and A maps each $e \in E$ to its uncertainty area A_e.

Lemma 17. *Let C be a circuit of M, and let B be a base of M containing an element $e \in C$. Then there exists an element $f \in C$ such that $(B \setminus \{e\}) \cup \{f\}$ is a base of M.*

It follows from the previous lemma that:

Corollary 18. *Let C be a circuit of M, and let e be an element in C with a highest weight among all elements in C. Then a minimum base of $E \setminus \{e\}$ is also a minimum base of E.*

Before introducing the algorithm we define an order of elements denoted by $<_e$. Let f and g be two elements in E. We say $f <_e g$ if $L_f < L_g$ or ($L_f = L_g$ and $U_f < U_g$). Edges with the same upper and lower weight limit are ordered arbitrarily. We also say E is *indexed by* $<_e$ if $\{e_1, \ldots, e_n\} = E$ where $e_i <_e e_{i+1}$. Based on the order $<_e$ and the resulting indexing, the first circuit created by taking the lowest indexed elements of any set $E' \subseteq E$ is called the *first circuit* of E'.

Algorithm 2. U-RED2

1. fix $<_e$; index E by $<_e$; set $E' := E$
2. repeatedly remove an element b from E' that is a highest element of some circuit C of E', i.e., that satisfies $L_b \geq U_c$ for all $c \in C \setminus \{b\}$
3. **if** E' is independent **then**
4. stop and output E' as a minimum base of M
5. **else**
6. set $C :=$ the first circuit over E'
7. set $h :=$ an element in C with a maximum upper limit
8. set $f :=$ an element in $C \setminus \{h\}$ that could be potentially higher than h
9. query the witness set $\{f, h\}$ and restart

Algorithm U-RED2 is shown in Algorithm 2. In Step 2, it repeatedly removes an element that can be identified as a highest element in a circuit, based on just the uncertainty areas that are known to the algorithm at that time. In Step 9, we query $\{f, h\}$. The basic idea to prove that $\{f, h\}$ is a witness set is as follows: By the choice of h and f, on the one hand, h could be a highest element of C and would then be excluded from any minimum base over E. On the other hand, even if all elements except h and f are queried, C being the first circuit implies that for each circuit $C' \neq C$ containing h, C' contains an element higher than L_h, i.e., h could be part of any minimum base over E. So, we must query at least one element in $\{f, h\}$.

Lemma 19. *The set $\{f, h\}$ in the algorithm U-RED2 is a witness set.*

Lemma 20. *Once the algorithm does not restart, E' is a minimum base of M.*

Proof. The set E' is independent and hence does not contain any circuits. Thus, E' is the only, and therefore also minimum, base over E'. Since E' was derived from E by removing only highest elements of a circuit, by a repeated application of Corollary 18, E' is a also a base for E. □

Finally we note that $OPT \geq k$ where k is the number of times the algorithm restarts. Since the number of queries requested by the algorithm is $2k$, U-RED2 makes at most $2 \cdot OPT$ queries.

Theorem 21. *The U-RED2 algorithm solves the minimum matroid base under uncertainty problem with at most $2 \cdot OPT$ queries.*

6 Competitive Lower Bounds

In this section we present lower bounds on the competitive ratio of online algorithms for cheapest set under uncertainty as well as its special cases.

An algorithm is called strongly ρ-competitive if $ALG \leq \rho OPT$ for all instances of the problem, and weakly ρ-competitive if there is a constant c such that $ALG \leq \rho OPT + c$ for all instances. Here, ALG is the number of queries made by the algorithm, and OPT is the size of an optimal query solution. A lower bound on the best possible competitive ratio of weakly competitive algorithms is also a lower bound for strongly competitive algorithms. We can prove lower bounds for weakly competitive algorithms, covering both the multiplicative ratio ρ and the additive constant c. The proofs use only singleton sets and open intervals as areas of uncertainty.

Definition 22. *We say that a variant of the cheapest set problem with uncertainty has (deterministic) lower bound $\lambda OPT + \delta$ if the two conditions hold:*
- Any algorithm that satisfies $ALG \leq \rho OPT + O(1)$ for all instances has $\rho \geq \lambda$.
- Any algorithm that satisfies $ALG \leq \lambda OPT + c$ for all instances has $c \geq \delta$.

Theorem 23. *CSU (MMCTU) has a lower bound of $d \cdot OPT + d$ (of $d \cdot OPT + 1$).*

Finally, we remark that the lower bound proof in [5] implies the following:

Theorem 24. *MMBU has a lower bound of $2 \cdot OPT$.*

7 Conclusion

In this paper, we have studied online and offline variants of cheapest set problems under uncertainty. While our lower bounds are tight for deterministic algorithms, an interesting direction for future research is to determine whether

query-competitive algorithms with better ratios are possible using randomization. Another direction would be to identify further variants of cheapest set problems whose structure admits better competitive ratios than the general case.

Acknowledgements. We would like to thank Gerhard Woeginger for pointing out that Lemma 13 can be proved using Theorem 1 in [11]. The first author would also like to thank Anita Maring for helpful discussions about lower bound examples for the general cheapest set problem. The second author would like to thank the University of Leicester to support this research in granting him academic study leave.

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Min-max and min-max regret versions of combinatorial optimization problems: A survey. European Journal of Operational Research 197(2), 427–438 (2009)
2. Bruce, R., Hoffmann, M., Krizanc, D., Raman, R.: Efficient update strategies for geometric computing with uncertainty. Theory of Computing Systems 38(4), 411–423 (2005)
3. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial Optimization. John Wiley and Sons, New York (1998)
4. Dorrigiv, R., Fraser, R., He, M., Kamali, S., Kawamura, A., López-Ortiz, A., Seco, D.: On minimum-and maximum-weight minimum spanning trees with neighborhoods. In: Erlebach, T., Persiano, G. (eds.) WAOA 2012. LNCS, vol. 7846, pp. 93–106. Springer, Heidelberg (2013)
5. Erlebach, T., Hoffmann, M., Krizanc, D., Mihalák, M., Raman, R.: Computing minimum spanning trees with uncertainty. In: Albers, S., Weil, P. (eds.) 25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008). LIPIcs, vol. 1, pp. 277–288. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany (2008)
6. Feder, T., Motwani, R., O'Callaghan, L., Olston, C., Panigrahy, R.: Computing shortest paths with uncertainty. Journal of Algorithms 62(1), 1–18 (2007)
7. Feder, T., Motwani, R., Panigrahy, R., Olston, C., Widom, J.: Computing the median with uncertainty. SIAM Journal on Computing 32(2), 538–547 (2003)
8. Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica 18(1), 3–20 (1997)
9. Kahan, S.: A model for data in motion. In: 23rd Annual ACM Symposium on Theory of Computing (STOC 1991), pp. 267–277 (1991)
10. Khanna, S., Tan, W.C.: On computing functions with uncertainty. In: 20th Symposium on Principles of Database Systems (PODS 2001), pp. 171–182 (2001)
11. Lindström, B.: A theorem on families of sets. Journal of Combinatorial Theory (A) 13, 274–277 (1970)
12. Löffler, M., van Kreveld, M.: Largest and smallest tours and convex hulls for imprecise points. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 375–387. Springer, Heidelberg (2006)
13. Olston, C., Widom, J.: Offering a precision-performance tradeoff for aggregation queries over replicated data. In: 26th International Conference on Very Large Data Bases (VLDB 2000), pp. 144–155 (2000)