Automated Category Tree Construction: Hardness Bounds and Algorithms

SHAY GERSHTEIN, Tel Aviv University, Tel Aviv, Israel
URI AVRON, Tel Aviv University, Tel Aviv, Israel
IDO GUY, Ben-Gurion University of the Negev, Beer-Sheva, Israel
TOVA MILO, Tel Aviv University, Tel Aviv, Israel
SLAVA NOVGORODOV, Tel Aviv University, Tel Aviv, Israel

Category trees, or taxonomies, are rooted trees where each node, called a category, corresponds to a set of related items. The construction of taxonomies has been studied in various domains, including e-commerce, document management, and question answering. Multiple algorithms for automating construction have been proposed, employing a variety of clustering approaches and crowdsourcing. However, no formal model to capture such categorization problems has been devised, and their complexity has not been studied. To address this, we propose in this work a combinatorial model that captures many practical settings and show that the aforementioned empirical approach has been warranted, as we prove strong inapproximability bounds for various problem variants and special cases when the goal is to produce a categorization of the maximum utility.

In our model, the input is a set of n weighted item sets that the tree would ideally contain as categories. Each category, rather than perfectly match the corresponding input set, is allowed to exceed a given threshold for a given similarity function. The goal is to produce a tree that maximizes the total weight of the sets for which it contains a matching category. A key parameter is an upper bound on the number of categories an item may belong to, which produces the hardness of the problem, as initially each item may be contained in an arbitrary number of input sets.

For this model, we prove inapproximability bounds, of order $\tilde{\Theta}(\sqrt{n})$ or $\tilde{\Theta}(n)$, for various problem variants and special cases, loosely justifying the aforementioned heuristic approach. Our work includes reductions based on parameterized randomized constructions that highlight how various problem parameters and properties of the input may affect the hardness. Moreover, for the special case where the category must be identical to the corresponding input set, we devise an algorithm whose approximation guarantee depends solely on a more granular parameter, allowing improved worst-case guarantees, as well as the application of practical exact solvers. We further provide efficient algorithms with much improved approximation guarantees for practical special cases where the cardinalities of the input sets or the number of input sets each items belongs to are not too large. Finally, we also generalize our results to DAG-based and non-hierarchical categorization.

CCS Concepts: • **Theory of computation → Data structures and algorithms for data management**; *Approximation algorithms analysis*; *Problems, reductions and completeness.*

Additional Key Words and Phrases: maximum independent set, approximation algorithms, approximation hardness bounds, taxonomy construction, category tree construction

Authors' Contact Information: Shay Gershtein, Tel Aviv University, Tel Aviv, Israel; e-mail: shayg1@mail.tau.ac.il; Uri Avron, Tel Aviv University, Tel Aviv, Israel; e-mail: uriavron@mail.tau.ac.il; Ido Guy, Ben-Gurion University of the Negev, Beer-Sheva, Southern, Israel; e-mail: idoguy@acm.org; Tova Milo, Tel Aviv University, Tel Aviv, Israel; e-mail: milo@cs.tau.ac.il; Slava Novgorodov, Tel Aviv University, Tel Aviv, Israel; e-mail: slavanov@post.tau.ac.il.

ACM 1557-4644/2024/5-ART
https://doi.org/10.1145/3664283

1 INTRODUCTION

Category trees, or taxonomies, are rooted trees where each node corresponds to a labeled category defined as a set of related items. Each non-leaf category is more general than its descendants and contains the union of their item sets. Moreover, each item may typically appear in a bounded number of tree branches. Such trees enable browsing-style information access and play a central role in Web platforms. While taxonomists can identify many desirable categories, producing a single categorization in a compact structure to maximize a given utility measure is challenging. Therefore, multiple algorithms for automating construction in various domains, e.g., e-commerce [3, 14], document management [11], and question answering [29], have been proposed, employing a variety of clustering approaches, and crowdsourcing [11, 26]. However, to our knowledge, the complexity of this problem has not been studied w.r.t. a formal model, and solution evaluations were based on user-studies or a similarity score of the tree categories to a collection of desired categories [11, 22, 23], to measure how well these are captured by the much more succinct solution. Based on the latter evaluation method, we propose a model that captures practical settings and show that the aforementioned heuristic approach has been warranted, as we prove strong inapproximability bounds.

Before describing our results, we first define the formal setting.

Model. The input is a set of n sets of items. The solution space consists of rooted trees (we also examine other structures, as described in the sequel). Each node (category) corresponds to a set of items (not necessarily identical to any set in the input), and every non-leaf category contains the union of all the items of its descendants. Ideally, the tree would have, for every input set, a category that is very similar to it. Each input set is weighted to reflect how valuable it is for a solution to contain a matching category. In practice, an input set represents items that match some criteria a user may have in mind when performing a search, and its weight implies the predicted likelihood of seeking these criteria. The sets are derived from a dataset of result sets to search queries, or, more generally, are formed by grouping items w.r.t. shared properties.

This model has multiple variants, defined by two parameters. The first parameter is a *similarity function*, which measures the similarity of an input set and a category. We examine several variations of commonly used set-similarity functions, which extend the original function with a threshold parameter. A similarity score below this parameter is rounded down to 0, to capture the fact that, when the similarity score is too low, no category is identifiable by the user as a matching category, and has no utility. Given such a function, the tree score for a given input set is the maximum similarity score of any category for this set. The overall tree score is the weighted (w.r.t. input weights) sum of the scores for all the input sets. The goal is to produce a tree of the highest score.

The second parameter is a *copy-bound*, which limits the number of *independent* categories any item can belong to, where categories are called independent if no two are on the same path from the root to a leaf. Most real-life platforms set a low copy-bound, to ensure that the categorization is coherent, compact, and easy to navigate. For example, eBay allows listing an item in a single (lowermost) category for free, or in two categories for an extra fee [1]. In our model, we assume the copy-bound to be constant.

Our bounds also apply to the related problems, where the aim is to produce a flat categorization or more general DAG structures. Flat categorization may be of independent interest, as it also captures the setting where one seeks, given a collection of overlapping sets, a partition that is maximally similar to the original collection. This may be particularly relevant for clustering and partitioning problems in hypergraphs (see Section 7).

Results. For the optimization problem of maximizing the tree score (as defined above), we prove for all examined variants an inapproximability bound of $\tilde{\Theta}(\sqrt{n})$ or $\tilde{\Theta}(n)$, where n is the number of input sets, highlighting how different problem parameters may affect the hardness. These bounds also apply to unweighted inputs and various special cases. On the other hand, we show that finer properties, such as bounds on the cardinality of input sets or the number of intersections among sets, aid in deriving more relaxed parameterized hardness bounds. To that end, we provide multiple results for special cases of the problem, where the guarantee in practical settings

can be much improved. Concretely, we first show how one can at a small cost to the approximation guarantee transform the problem such that the input is unweighted and all input sets are of roughly the same size. Moreover, we provide a positive result in the form of an algorithm for the *Exact variant*, where a category must match an input set exactly to contribute to the objective function. The tight performance guarantee of this algorithm depends only on a parameter that measures the number of intersections between the input sets. Importantly, we reduce this variant to the Maximum Independent Set problem, for which, despite its inapproximability, practical solvers have been devised. We demonstrate the practical utility of this result, in [5] and [4], complementary works focusing on constructing an e-commerce category tree that is maximally similar to result sets of user queries. In these works, we show that leveraging these solvers enables finding optimal solutions to real-world instances and extend this approach to heuristic algorithms that solve well instances of more general variants. We further study more granular parameters of the problem, such as the maximum number of input sets an item may belong to or the maximum cardinality of an input set. We devise efficient algorithms that provide much improved approximation guarantees (including even constant approximation ratios) for practical special cases where these parameters have relatively low values.

An essential component in our methods is defining a generalized similarity function with two threshold parameters that address two more granular similarity measures: *precision* and *recall*. Our key results consist of reductions from the Maximum Independent Set problem in hypergraphs, where we integrate into the reduced instance randomized constructions that closely capture the precision and recall parameters. This not only enables us to derive improved results for more subtle special cases but also to capture more refined properties of hard inputs, which we then leverage to prove hardness w.r.t. other similarity functions (we outline how to schematically apply our arguments to derive hardness bounds for similarity functions not examined here).

While we are not aware of any theoretical results directly comparable to ours, Section 7 discusses motivating empirical research and possible applications to hypergraph partitioning.

We note that the complementary problem of labeling the resulting categories has been studied in various settings (e.g., [7]), and is outside the scope of our model.

Real world problem setting. As detailed in Section 7, the construction of category trees has been studied in numerous fields, including e-commerce, document management, and question answering. To motivate the problem and provide practical intuition, we now describe typical real-world issues when constructing category trees in e-commerce.

Product category trees are integral to the user experience on major e-commerce platforms, including eBay, Amazon, Walmart, and Taobao, all billion-dollar corporations. These structures facilitate the organization of products into well-defined groups based on specific attributes, such as "memory size" for smartphones or "sleeve length" for shirts. However, maintaining up-to-date and relevant category trees is a labor-intensive task, typically managed by in-house taxonomist teams, which is expensive and time-consuming. The latter is particularly problematic when the item repository is massive and evolves rapidly along with user interests. This manual construction often leads to category trees that are lacking or outdated since it is hard to keep track of market trends, seasonal changes, holidays, and special events [30]. To address this, many works have devised effective automated construction and maintenance algorithms for tree-based categorizations, which struggle to keep pace with changing market trends and user preferences [3, 12, 14].

While taxonomists can manually and algorithmically [20] identify numerous candidate categories, i.e. item subsets with a shared label, most such categories cannot simultaneously exist in a category tree, due to the aforementioned combinatorial restrictions. Concretely, all large retailers impose a very low bound (the copy-bound in our model) on the number of leaf categories an item may belong to. Nevertheless, the goal is to have in a tree a category that is very similar to the set of items the user is interested in examining (e.g. 60-inch Samsung TVs), such that as many of the matching items are contained in a corresponding tree category (so that a user doesn't have to look in many places), and at the same time there are not many items in the category that are not

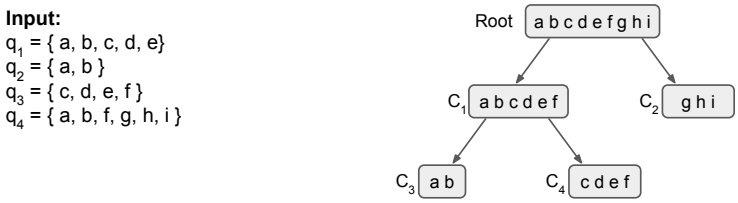

Fig. 1. Example of input sets and a category tree.

Fig. 2. A sample of products from the shirts category.

in the set of items the user is interested in, so that the user would not have to filter many items while browsing. These two objectives are naturally mapped to the two granular similarity measures we discussed above: the former objective maps to a recall requirement and the latter maps to a precision requirement. As we show in the present work, adhering to the combinatorial constraints imposed on category trees in real-world settings, does not lend itself to efficient solutions when aiming to maximize precision and recall.

We note that one approach to automatically identifying the sets of items users are interested in, such that trends are captured, is to use search queries, submitted to the platform's search engine. This has been explored in [4, 5], works based on the model proposed here, and for this reason we will refer to the input sets as queries (to simplify terminology, as sets are mentioned in many different contexts). The intuition for this approach is that if the tree categories are very similar to result sets of user queries, then the tree arguably allows users to identify exactly the sets of items they are interested in examining.

We next illustrate the setting of our problem with the following toy example, depicted in Figure 1. The figure presents a possible tree that can be constructed over the input sets, denoted by q_1, q_2, q_3 and q_4, provided on the left side. For brevity and consistency, we will denote the items throughout the paper by short literal notation. Nevertheless, to provide a practical context, we show in Figure 2 a possible real-world instantiation of the items in the example. Concretely, the 9 items depicted in the figure, corresponding to the items $a, b, \ldots, i$ in Figure 1 (the corresponding item appears under each photo), are a sample of the shirts available in a company's catalog. These shirts have different brands (a, b are Adidas, c, d, e, f are Nike, g is Puma, h is Reebok, and i is Umbro), colors (a, b, c, d, e are black, f is red, g is blue, h is grey, and i is white), and sleeve lengths (a, b, f, g, h, i have long sleeves, and the rest have short sleeves). Then the sets q_1, q_2, q_3, and q_4 may correspond, respectively, to the result sets of the following 4 queries: "black shirt," "black Adidas shirt," "Nike shirt," and "long sleeve shirt." We will use these as the input sets for our model in the current example. We are interested in constructing a tree whose categories are maximally similar to these four sets.

For this input, it is not hard to convince oneself that it is impossible to construct a tree that contains four categories that match the queries exactly, while satisfying the requirements that every category contains the union of its descendants and that every item appears in exactly one tree branch. Therefore, one must relax the

objective to either only exactly matching a subset of queries or not matching them exactly. Figure 1 depicts a possible tree for the latter case. It has two leaf categories, C_3 and C_4, that match q_2 and q_3 exactly, respectively. For $q1$, the most similar matching category is $C1$. While it contains all the items of q_1 (perfect recall score), it contains also the item f, which means that the precision score is $\frac{5}{6}$. In a practical scenario, if we believe that the user is only interested in seeing the items in $q1$, then including an irrelevant item diminishes the user experience. Nevertheless, we cannot simply remove the item without negative consequences, as we would then also have to remove it from C_4, and it would no longer match q_3 exactly. As for q_4, the most similar category is C_2. It has perfect precision but only 0.5 recall (the category contains half of the items in q_4). In some cases, such low recall may not be considered satisfactory, and when evaluating the tree, it may not contribute to the tree at all if one requires a higher standard of matching the input queries. In this work, we will formalize such considerations.

Outline. Section 2 provides the necessary formalism for our model, while Section 3 presents useful theoretical tools. In Section 4 we prove various approximation hardness results derived w.r.t. the generalized similarity function (with recall and precision thresholds). In Section 5 we leverage these results to derive hardness bounds w.r.t. all other similarity functions defined in Section 2. In Section 6 we provide a positive result for a common problem variant, as well algorithm with much improved approximation guarantees for various practical special cases of the problem. The related work appears in Section 7 and we conclude in Section 8.

2 MODEL

We now define the model underlying our work, followed by a discussion of problem parameters. We conclude this section with illustrative examples of problem instances in our model.

2.1 Problem definition

The two problems we study are the Optimal Category Tree problem (*OCT*) and the Optimal Category Partition problem (*OCP*). The input to both problems is $\langle Q, U, W \rangle$, where $Q \subseteq 2^U$ is a set of n sets over a finite universe U of size $O(poly(n))$, and $W : Q \to [0, 1]$ is a weighting function that assigns a non-negative weight to each set in Q. We use the term *query*, to denote each set in Q. Note, in advance, that in the definition of the model we discuss two types of element sets: the queries and the sets corresponding to the tree nodes. In general, these sets are not identical (however, these are typically similar, as the objective is, roughly speaking, to maximize the similarity of the two types of sets, as defined formally below). To motivate the use of the *query* terminology, recall that as mentioned in the introduction, each set in the input may in practice correspond to a result set for a user query, and the weight can reflect, e.g. the query frequency.

Both problems have multiple variants defined by two parameters (explained below, in the context of the solution space): a *copy-bound* $r \in \mathbb{N}$ (assumed to be constant), and a *similarity function* $\mathcal{S} : [2^U] \times [2^U] \to [0, 1]$. We denote by $OCT^r(\mathcal{S})$ the r-copy *OCT* problem with similarity function $\mathcal{S}$, and the analogous *OCP* variant is denoted by $OCP^r(\mathcal{S})$.

We next formally define the solution space for each of the two problems. We start with $OCP^r(\mathcal{S})$, as it is a simpler form of the model for $OCT^r(\mathcal{S})$.

OCP. We call a set of sets over U an *r-weak partition*, if every element appears in at most r sets. A 1-weak partition is a standard partition. The solution space of $OCP^r(\mathcal{S})$ consists of all r-weak partitions of U. Any such solution is termed as a *category partition*, denoted by P, with the sets contained in it termed as *categories*.

Given a query, $q \in Q$, and a category partition, P, we define the *similarity score* of a category $C \in P$ for q as $\mathcal{S}(q, C)$. The score of P for this query is defined by the category that most closely matches the query as $S(q, P) = \max_{C \in P} \mathcal{S}(q, C)$. The overall score of the category partition is defined as $S(Q, P) = \sum_{q \in Q} W(q) \cdot S(q, P)$. This score is the weighted sum of the scores for all queries, where the weight of each score is the corresponding

query weight. The objective of the $OCP^r(\mathcal{S})$ problem is to produce a category partition of the maximum score: $\arg\max_P S(Q,P)$.

OCT. The solution space of $OCT^r(\mathcal{S})$ consists of rooted trees, termed *category trees*, where every tree node, termed *category*, contains a subset of U. We abuse notation, and, when clear from context, we use T to denote both the category tree and the set of its categories. Similarly, we use C to denote a category as well as the set of elements it contains.

A category tree must satisfy the following two requirements. First, every non-leaf category contains the union of the sets of elements contained by its child categories (and possibly other elements). The root of the tree, thus, contains all the elements that appear in any category. Second, for each element $e \in U$ there are at most r semi-leaves w.r.t. e, where the semi-leaves w.r.t. e are the most specific categories to which e belongs (i.e. none of the descendants of any such semi-leaf contain e). Alternatively stated, any set of tree categories with no two categories in the set belonging to the same *branch*, where a branch is a simple path from the root to a leaf, must be an r-weak partition. Note that for a category tree, unlike a category partition, it is no longer true (nor desirable) that an element is contained in at most r categories. Even for $r = 1$, if an element is contained in some category in the tree, it must also be contained in all its ancestor categories. Therefore, the copy-bound is applied to the number of semi-leaves w.r.t. any given element, with the only other nodes containing the element being all the ancestors of these semi-leaves. For $r = 1$ this requirement implies that any given element in the tree is contained only in categories that are all on the same branch.

All definitions of relevant scoring functions are analogous to $OCP^r(\mathcal{S})$. Concretely, the score of a tree, T, for a query, q, is defined as $S(q,T) = \max_{C \in T} \mathcal{S}(q,C)$. The overall score of T is $S(Q,T) = \sum_{q \in Q} W(q) \cdot S(q,T)$. When Q is clear from context, we use the shorthand $S(T)$. The objective of the $OCT^r(\mathcal{S})$ problem is to produce $\arg\max_T S(Q,T)$.

Unweighted variants. We refer to the special case of *OCT* (*OCP*) where all weights are uniform as *unweighted OCT* (*OCP*) and set all weights to 1. Our hardness proofs leverage unweighted inputs, and therefore our hardness bounds also apply to the unweighted case. Accordingly, in our hardness discussions, the reader may assume this context. We directly use weights only in the algorithms we provide in Section 6.

2.2 Similarity functions

We study several similarity functions, that are dependent (in some cases, implicitly) on the following two underlying similarity measures, *precision* $p(q,C) = \frac{|C \cap q|}{|C|}$ and *recall* $r(q,C) = \frac{|C \cap q|}{|q|}$. We distinguish between *cutoff functions* and *threshold functions*. Both have a threshold parameter $\delta \in (0,1]$ and use an underlying similarity function f. In both cases, the function outputs 0 when $f(q,C) < \delta$. However, when $f(q,C) \geq \delta$ a cutoff function equals $f(q,C)$, whereas a threshold function equals 1. We first focus, however, on the following, more general, threshold function, which sets a separate threshold for each measure.

Definition 2.1 (Granular threshold function). Given parameters $\alpha, \beta \in [0,1]$, the *granular threshold similarity* $\mathcal{T}_{\alpha,\beta}$ of a query q and category C is defined as follows: $\mathcal{T}_{\alpha,\beta}(q,C) = 1$ when $p(q,C) \geq \alpha$ and $r(q,C) \geq \beta$, and $\mathcal{T}_{\alpha,\beta}(q,C) = 0$ otherwise.

We will also study the common similarity functions defined below.

Definition 2.2 (Jaccard similarity). The *Jaccard similarity* of a category C and a query q is defined as $J(q,C) = \frac{|q \cap C|}{|q \cup C|}$. The *threshold Jaccard similarity*, with threshold parameter $\delta \in (0,1]$, is defined as $\hat{J}_\delta(q,C) = 1$ when $J(q,C) \geq \delta$ and $\hat{J}_\delta(q,C) = 0$ otherwise. The *cutoff Jaccard similarity*, with threshold parameter $\delta \in (0,1]$, is defined as $\bar{J}_\delta(q,C) = J(q,C)$ when $J(q,C) \geq \delta$ and $\bar{J}_\delta(q,C) = 0$ otherwise.

Definition 2.3 (F_1 score). The F_1 *score* of a category C for a query q is defined as the harmonic mean of the precision and the recall: $F_1(q,C) = 2\frac{p(q,C)\cdot r(q,C)}{p(q,C)+r(q,C)}$. The *threshold* F_1 *score*, with parameter $\delta \in (0,1]$, is defined as $\hat{F}_{1(\delta)}(q,C) = 1$ when $F_1(q,C) \geq \delta$ and $\hat{F}_{1(\delta)}(q,C) = 0$ otherwise. The *cutoff* F_1 *score*, with threshold $\delta \in (0,1]$, is defined as $\bar{F}_{1(\delta)}(q,C) = F_1(q,C)$ when $F_1(q,C) \geq \delta$ and $\bar{F}_{1(\delta)}(q,C) = 0$ otherwise.

We remark that unless specified otherwise, we assume henceforth that the threshold parameters (α, β, and δ) are constants.

Cover terminology. If a category C has the highest score for a query q (if necessary, ties are broken arbitrarily), and that score is not 0, we say that C *covers* q. We call a category that covers at least one query a *covering* category, and a branch containing a covering category is a *covering* branch. A set of categories is *independent* if no two categories are on the same branch (*OCP* categories are independent). Similarly, a set of queries are *independently-covered*, if each is covered by a different category, and the covering categories are independent. Observe that in unweighted instances (where, as noted earlier, all weights are assumed to be 1) with threshold functions the score equals the number of covered queries.

Note that, all functions defined above share the special case of $\mathcal{T}_{1,1}$ (Definition 2.1) (equivalent to setting $\delta = 1$ in Definitions 2.2 and 2.3), where a query q is covered by a category C only if $q = C$. We refer to this variant as the *Exact variant*.

Canonical form. Any category tree can be reduced to a canonical form, without decreasing the score, by (1) removing non-covering categories, (2) connecting the parent and children of any removed category, and (3) removing from category C and its descendants any element not contained in any query covered by C or categories below C (this may even improve the precision and the score). If a query is covered by multiple categories, one can assign arbitrarily a single category that is said to cover it, and then reduce it to a canonical form, as described above, w.r.t. this assignment. Similarly, adding new categories that do not affect the contents of the existing categories cannot decrease the tree score. This discussion applies analogously to category partitions.

Choices of parameters. For practical applicability, we focus on variants where $r = \Theta(1)$ and the similarity functions have threshold parameters. A low copy-bound ensures a concise categorization, and in many platforms, r is a small constant, typically, 1 (e.g., [1]). This parameter controls the trade-off between the score and the conciseness, and our parameterized bounds hint at a quantification of this trade-off. Threshold parameters capture the fact that, below a certain similarity score, a category has no utility. Without thresholds, trees that cover unacceptably poorly all queries may be mathematically preferable to trees that cover well a smaller number of queries. Nevertheless, to capture more tolerant settings, we also provide approximation bounds for polynomially small threshold parameters.

We also note that, in practice, errors in precision and recall have an asymmetric effect. For example, perfect recall with precision of $\frac{1}{2}$, enables the user to examine all relevant items to identify the best matches, while ignoring every other item. This may be acceptable, especially for smaller categories. However, in the analogous case of perfect precision and recall of $\frac{1}{2}$, other categories may or may not contain better matching items, and the user might waste time looking for non-existing or hard-to-find categories or be unaware of better options. It may, therefore, be tempting, in some applications, to require perfect recall. To that end, we examine this case separately and show that it admits the strictest inapproximability.

More generally, there exists a key tension between the recall and precision thresholds. Consider, as an extreme example, a recall threshold of 1, and a precision threshold of 0 (i.e., perfect recall with no precision requirement). For this variant, a tree consisting only of a root that contains all the elements is an optimal solution. At the other extreme, if we require perfect precision with no constraint on the recall, then an optimal solution is a tree where there is a leaf for each element, containing only that element, and a root connected directly to all the leaves. These edge cases illustrate the following intuitive phenomenon: increasing precision thresholds leads to more granular trees with smaller covering categories, whereas increasing recall thresholds generally produces a more

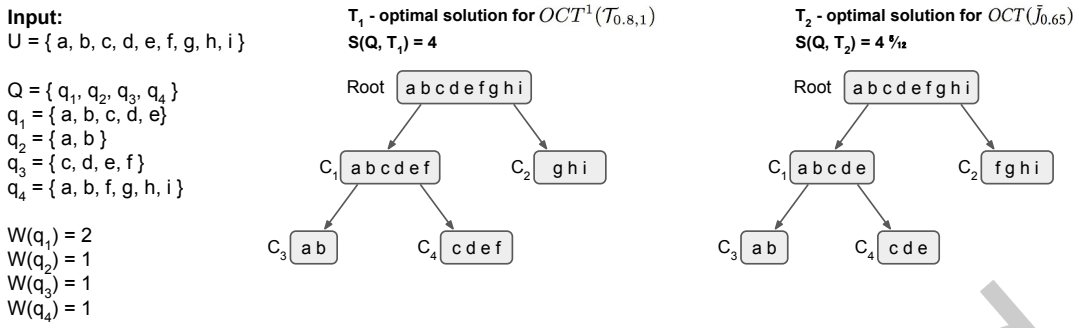

Fig. 3. Optimal solutions for two *OCT* variants over the same input (where, for simplicity, the input weights are not normalized), depicted on the left side. The category tree, T_1, is an optimal solution for the $OCT^1(\mathcal{T}_{0.8,1})$ variant, where C_1 covers q_1, C_3 covers q_2, and C_4 covers q_3, with the overall score of $W(q_1) + W(q_2) + W(q_3) = 4$. The rightmost tree, T_2, is the optimal solution for the cutoff Jaccard variant with $\delta = 0.6$, where C_1 covers q_1 with the score of 1, C_2 covers q_4 with the score of $\frac{2}{3}$, C_3 covers q_2 with the score of 1, and C_4 covers q_3 with the score of $\frac{3}{4}$, resulting in the overall score of $W(q_1) \cdot 1 + W(q_2) \cdot 1 + W(q_3) \cdot \frac{3}{4} + W(q_4) \cdot \frac{2}{3} = 4\frac{5}{12}$.

coarse categorization with larger covering categories. These properties of the precision and recall thresholds are formalized and leveraged in our hardness proofs.

The above edge cases also motivate the use of thresholds in our similarity functions, below which the score is 0. Concretely, consider the following extreme example: if no restrictions are imposed, assume a very large number of queries, each at most of cardinality 100. One could trivially achieve an approximation ratio of at least 100, by simply placing every item in a separate leaf category. The precision in all covers would be 1 and the recall is at most 1/100. However, such a tree is of no practical use (e.g. when searching for clothes, the user would need to inspect a separate category for every item).

Multiplicity parameter. We next define the multiplicity parameter. We will prove hardness bounds and devise algorithms with approximation guarantees that are parameterized by the multiplicity. For small multiplicity values, we show that many problem variants are much easier to approximate than the general case.

Given a subset of queries Q', the multiplicity $M_{Q'}(e) = |\{q \in Q' \mid e \in q\}|$ of an element e in Q' is defined as the number of queries in Q' that e appears in. We refer to $M(e) = M_Q(e)$ as the multiplicity of e (i.e. its multiplicity across all input queries). The maximum multiplicity of any element is denoted by $M = \max_e M(e)$. Henceforth, any mention of multiplicity not in the context of a specific item, refers to the maximum multiplicity M.

We can assume henceforth that $M \geq 2$, as for $M = 1$ all queries are disjoint, and therefore the problem is trivial, as one can simply output a solution that consists of a partition of categories that is identical to Q.

2.3 Examples of Problem Instances

We illustrate the *OCT* setting with $r = 1$ via the following toy examples, depicted in Figure 3, that we also described less formally in the Introduction. The figure presents two optimal solutions, computed by brute-force, corresponding to two different *OCT* variants, over the same input, provided on the left side. For convenience, as it is easier to perform arithmetic with integers, we provide integer weights, instead of normalizing into $[0, 1]$, since the normalization of the weights and scores does not affect the complexity of the problem or the performance ratio of the various solutions.

Observe that the overall weight of all four queries is 5, hence this is also an upper bound on the score of any tree for any variant over this input. In addition, observe that, since $r = 1$, we cannot add any branches to either of the depicted trees, without violating the copy-bound constraint.

Example 2.4. The tree T_1, depicted in the middle of the figure, is the optimal solution for the $OCT^1(\mathcal{T}_{0.8,1})$ variant (i.e. precision 0.8 and perfect recall). The categories C_3 and C_4 cover the queries q_2 and q_3, respectively, as they are identical to these queries (and would cover them even for $\alpha = 1$). The category C_1 covers q_1 as its recall score is 1, and 5 out of the 6 items in C_1 are in q_1, hence the precision is $\frac{5}{6} > \alpha$. Note that, we must include f in C_1 since it appears in C_4, and removing f from both categories, would result in C_4 no longer covering q_3. Moreover, there is no incentive to place f elsewhere, since the score, when using a binary function, is not penalized for precision errors if the threshold is exceeded.

As for the category C_2, its addition to the tree is optional, since it does not cover any query, despite all its items belonging to the uncovered query, q_4, as we can no longer achieve perfect recall without the items $\{a, b, f\}$. It is easy to verify that there is no way to cover q_4 by adding a matching category above or below C_1, such that the items $\{a, b, f\}$ would be shared by all categories, without decreasing the precision of other queries to values below the threshold.

Example 2.5. We next discuss, T_2, the optimal solution for $OCT^1(\bar{J}_{0.65})$, the cutoff Jaccard variant with $\delta = 0.65$, depicted on the right side of Figure 3. It overlaps with T_1, except for the item f, which is placed in C_2 instead of C_4 and C_1. In this case, compared to the previously examined variant, since Jaccard variants allow for errors in both precision and recall, and also since we use a lower threshold, it is now possible to cover all queries, albeit with imperfect scores. Indeed, every non-root category in T_2 covers a query, as explained in the figure. Moreover, q_1 is the query of the maximal weight, hence it is not surprising that the optimal tree covers it with a perfect score, at the expense of errors in the covers of less significant queries. We note that, in practice, the same category often covers multiple queries. For instance, if we decrease the threshold from 0.65 to 0.4, then C_1 would also cover q_2, as its precision w.r.t. q_2 is exactly 0.4.

3 PRELIMINARIES

We provide here known results and definitions, that we will use in our hardness proofs. We conclude the section by explaining how proofs are tailored to fit both *OCT* and *OCP* simultaneously, and discuss generalizing a tree to a DAG.

Notation. To simplify the presentation, we use a "tilde-Theta" notation, $\tilde{\Theta}(\cdot)$, to hide sub-polynomial factors. Whenever we state that a variant has inapproximability of $\tilde{\Theta}(n^c)$, for some constant $c \in (0, 1]$, this compact notation implies the more formal argument that, for any $\epsilon > 0$, this variant cannot be approximated within a factor of $O(n^{c-\epsilon})$. We note that a solution of score 1 can always be achieved by producing a single category that equals one of the queries (for differently weighted queries we will specifically select the query of the highest weight). Thus, $\tilde{\Theta}(n)$ is the strictest possible inapproximability factor, using this notation.

Complexity Assumptions. We next define the complexity class *ZPP*, as some of our results use the assumption $ZPP \neq NP$. It is known that $P \subseteq ZPP \subseteq NP$ and that $ZPP \subseteq BPP$, where *BPP* is the class of problems solvable by a randomized PTIME algorithm with a two-sided error.

Definition 3.1. The complexity class *ZPP* contains the problems for which there is a PTIME algorithm that outputs DO NOT KNOW with a probability of less than 1/2, and outputs the correct answer with the remaining probability.

MIS. We leverage reductions from the Maximum Independent Set problem (*MIS*) in *uniform hypergraphs*. In an r-uniform hypergraph, all (hyper)edges are vertex subsets of cardinality r. The special case of $r = 2$ is a graph.

Definition 3.2. In the Maximum Independent Set problem (*MIS*) in uniform hypergraphs, the input is a uniform hypergraph $G = (V, E)$, and the objective is to find a vertex set $S \subseteq V$ of maximum cardinality, subject to the constraint that no edge from E is contained in S.

We have made use of the following known results for *MIS*, where $n = |V|$.

Theorem 3.3. *[13] The MIS problem in r-uniform hypergraphs, for constant $r \geq 2$, cannot be approximated below a $\tilde{\Theta}(n)$ factor, unless ZPP = NP.*

Theorem 3.4. *[33][8] The MIS problem in graphs has inapproximability of $\tilde{\Theta}(n)$, unless P = NP. Moreover, for graphs of sufficiently large constant degree bound d, MIS is hard to approximate below a $\Theta(\frac{d}{\log^2 d})$ factor, unless BPP = NP. Furthermore, MIS is NP-hard even for regular graphs of degree 3.*

Theorem 3.5. *[9] For r-uniform hypergraphs with (not necessarily constant) maximum degree d, there exists a PTIME algorithm producing an independent set of size $\Omega(\frac{n}{d^{\frac{1}{r-1}}})$.*

From Theorem 3.5, we derive the following lemma.

Lemma 3.6. *Given an r-uniform hypergraph $G = (V, E)$, there exists a PTIME algorithm that produces an independent set in G of size $\Omega((\frac{|V|^r}{|E|})^{\frac{1}{r-1}})$.*

Proof of Lemma 3.6. The average degree of G is $\bar{d} = \frac{r|E|}{|V|}$. Let $V_1 \subseteq V$ denote the set of vertices in G whose degree is at most $d = 2\bar{d}$. A simple counting argument (concretely, Markov's inequality) implies that $|V_1| \geq \frac{|V|}{2}$. Consider the sub-hypergraph G_1 of G induced by V_1. Computing G_1 is the first step of the algorithm. Note that G_1 is still r-uniform. In particular, if an edge $e \in E$ is not contained in V_1 but at least one vertex in e is in V_1, then e is not replaced in G_1 by $e \cap V_1$. As follows from Definition 3.2, including all the vertices in $e \cap V_1$ in an *MIS* solution does not violate any *MIS* constraint.

In the second and last step, we apply over G_1 the algorithm from Theorem 3.5, which produces an independent set S. By Theorem 3.5, the size of this independent set is

$$|S| \geq \Omega(\frac{\frac{|V|}{2}}{d^{\frac{1}{r-1}}}) = \Omega(\frac{|V|}{(\frac{r|E|}{|V|})^{\frac{1}{r-1}}}) = \Omega((\frac{|V|^r}{|E|})^{\frac{1}{r-1}}).$$

□

Hard instances of MIS. When reducing from *MIS*, we will restrict ourselves to instances where the optimal solution is of size $\tilde{\Theta}(n)$. The $\tilde{\Theta}(n)$ inapproximability of *MIS* implies that this subset of inputs captures the maximal hardness. Accordingly, in our reductions, assuming this hard set of inputs, we will leverage the fact that one cannot find (in the worst case) an independent set of size $\Omega(n^\epsilon)$ for any $\epsilon > 0$.

Probabilistic Tools. We next define the Hypergeometric and Binomial distributions and present known tail bounds for both. These are useful in the analysis of our randomized reduction.

Definition 3.7. Consider sampling without replacement n uniformly random and independent samples from a set of N elements containing K special elements, and let X denote the number of special elements in the sample. Then, X is a *hypergeometric* random variable, denoted as $X \sim H(N, K, n)$, and its probability mass function is $\Pr(X = i) = \frac{\binom{K}{i}\binom{N-K}{n-i}}{\binom{N}{n}}$.

Definition 3.8. Consider performing n independent experiments with success probability p. Let X denote the number of successful experiments. Then, X is a *binomial* random variable, denoted as $X \sim B(n, p)$, with probability mass function is $\Pr(X = i) = \binom{n}{i} p^i (1-p)^{n-i}$.

We use the following tail-bound for the hypergeometric distribution.

Lemma 3.9. *[25] If $X \sim H(N, K, n)$, as defined in Definition 3.7, and letting $u = \frac{K}{N}$, then, for $t > 0$:*

$$\Pr(X \geq (u+t)n) \leq \left(\left(\frac{u}{u+t}\right)^{u+t}\left(\frac{1-u}{1-u-t}\right)^{1-u-t}\right)^{n}.$$

We also use the following Chernoff bound for the binomial distribution.

Lemma 3.10 (Chernoff Bound). *[28] If $X \sim B(n, p)$, as defined in Definition 3.8, then, denoting the expectation $\mu = np$, for $\delta \geq 1$:*

$$\Pr(X > (1+\delta)\mu) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}.$$

Hardness of OCP and DAGs. Finally, we explain how our hardness reductions for *OCT* were devised to also apply for *OCP*, as well as for the more general problem where one is allowed to produce a rooted DAG with analogous combinatorial constraints. Thus, while the hardness analysis focuses on *OCT*, *the bounds apply for the above two problems as well.* Specifically, all the OCT hardness results, Theorems 4.1, 4.2, 5.1, and 5.2, apply exactly to *OCP* as well. There are two components that, when combined, ensure that our *OCT* hardness results apply to *OCP* as well. First, we will show below that it is straightforward to prove that over any input, the optimal *OCT* solution is of at least the same score as the optimal *OCP* solution. Second, we will construct our hardness reductions such that the optimal scores for both problems are of the same maximum order of $\tilde{\Theta}(n)$. Hence, the ratio between the best PTIME solution and the optimal solution can only be worse for *OCP*.

Loosely speaking, an algorithm that produces a tree has all the capabilities it would need for producing a similar-score partition, along with several additional possibilities to increase the score. Hence, in our analysis, by bounding what is possible for constructing a tree, we also bound what is possible for constructing a partition.

Concretely, observe that, over any given input, any category partition can be transformed into a category tree of the same score, by connecting all the categories directly to a root. For this reason, over any given input, the optimal score, that can be achieved by a category partition, cannot exceed the optimal score by a category tree. Importantly, in all examined variants, we ensure that our hardness bounds are derived over a subset of inputs for which there exists a category tree whose leaf categories induce a category partition of score $\tilde{\Theta}(n)$, implying that the optimal score of both problems is of at least this order, which is roughly maximal (the score for any input cannot exceed $n = |Q|$). It follows that all our approximation hardness bounds for *OCT* hold for *OCP* as well.

We note that our hardness proofs also apply to the more general problem where instead of a tree, one is allowed to produce any rooted DAG, maintaining the requirements that a category must contain all its descendants and for each element e there are at most r different paths from the root to a semi-leaf w.r.t. e (recall that a semi-leaf w.r.t. e is a most specific node to which e belongs). This follows from the fact the any such DAG can be converted to a valid tree solution, by removing edges, which does not affect the score.

4 HARDNESS OF $OCT^r(\mathcal{T}_{\alpha,\beta})$

In this section, we prove approximation hardness bounds on $OCT^r(\mathcal{T}_{\alpha,\beta})$ for various ranges of the threshold parameters. We first provide a reduction from *MIS* to $OCT^r(\mathcal{T}_{\alpha,\beta})$ where $\alpha = \Theta(1)$ and $\beta > \frac{1}{2}$, proving $\tilde{\Theta}(n^{\frac{1}{r+1}})$ inapproximability (n is the number of queries, and r is the copy-bound), unless $ZPP = NP$. For the special case of $OCT^r(\mathcal{T}_{\alpha,1})$ we improve this bound to $\tilde{\Theta}(n)$. For $r = 1$, we strengthen these bounds by using a weaker theoretical assumption and also provide a bound for the case of queries of bounded size.

To prove that the $\tilde{\Theta}(n^{\frac{1}{r+1}})$ inapproximability extends to the case where $\beta \leq \frac{1}{2}$, we use a more involved randomized reduction, and also provide an analysis that captures sub-constant ranges of the threshold parameters to derive a $\tilde{\Theta}((\alpha^{(r+2)}\beta n)^{\frac{1}{r+1}})$ inapproximability bound.

The remainder of this section consists of two subsections, pertaining to the two reductions. Each subsection is further divided into the reduction from *MIS*, the hardness results it implies, the intuition underlying the proof, and the formal proof. We also explain why different reductions were necessary.

4.1 Special cases with $\beta > \frac{1}{2}$

Before describing the first reduction, we provide a high-level sketch of the proof for the simpler special case of the Exact variant with $\alpha = \beta = 1$ and $r = 1$ to provide intuition. All hardness results in the sequel extend this core reduction.

Concretely, given an input graph, we create an *OCT* instance by replacing every edge with a corresponding *edge-element* and every vertex v with a query q_v containing all the edge elements corresponding to the edges incident to v, and also additional unique *padding elements* to ensure that all queries are of the same size. In any *OCT* solution, a query set consisting of at most one query from every tree branch would correspond to an independent set in the original graph. To see that this is indeed an independent set, assume for the sake of contradiction that there is an edge $\{u, v\}$ in this vertex set. This implies that there is an edge element that is in both q_v and q_u, and due to the perfect recall requirement, it must appear in both the categories covering q_v and q_u. However, since these are on different branches, that would violate the copy-bound constraint, thus proving the claim. It follows that if we can produce a tree that covers categories on many branches, then this would allow us to derive a large independent set, which would violate the *MIS* hardness results. Thus, if we prove that any tree that covers many queries has many different branches that cover a query, then this would imply that an *OCT* algorithm that finds a tree of a high score would also find a large independent set. To prove this claim, we can show that a tree can cover at most one query on every branch. Otherwise, the covering category C on that branch that is closer to the root must contain the elements of both queries (since $\beta = 1$), but since none of the two queries contains the other (they are guaranteed to be the same size), then C violates that precision constraint ($\alpha = 1$), as it contains at least one extra element that is not in the query it covers.

We now formally describe and analyze the first reduction.

Reduction from *MIS*. Given an algorithm for $OCT^r(\mathcal{T}_{\alpha,\beta})$, denoted by A, with a (worst-case) approximation guarantee of γ, we devise an algorithm $R = R_A$ for *MIS* in $(r+1)$-uniform hypergraphs. We compute a lower bound on the size of the independent set (*IS*) that R produces as a function of the approximation guarantee γ. This implies a lower bound on γ, as we will show that when γ exceeds this bound, one can produce an independent set of cardinality $\Omega(n^\epsilon)$ (n is the number of vertices in the hypergraph), for some small constant $\epsilon > 0$, contradicting the hardness of *MIS*.

The algorithm R consists of a sequence of three procedures, R_1, R_2, and R_3:

(1) Given an $(r+1)$-uniform hypergraph, $G = (V, E)$, R_1 transforms it into an instance Q of $OCT^r(\mathcal{T}_{\alpha,\beta})$. The universe of elements for Q consists of three types of elements: an *edge element* for every edge in E, *padding elements*, and $\frac{1-\beta}{2\beta-1}n^r$ *joint elements*. Specifically, for each vertex $v \in V$, we construct a query, q_v, such that $Q = \{q_v \mid v \in V\}$. Every query contains all the joint elements. Moreover, each query, q_v, also contains all the edge elements that correspond to edges incident to v in G. Finally, we add to every query as many unique padding elements as necessary, such that the size of the query is $\frac{n^r}{2\beta-1}$. Every padding element appears in only one query. It follows that every query contains $\frac{1-\beta}{2\beta-1}n^r$ joint elements and $\frac{\beta}{2\beta-1}n^r$ non-joint elements.
(2) R_2 consists simply of running A over Q. Let T denote the category tree A outputs.
(3) R_3 produces an *IS* $S \subseteq V$, as follows. Let $\hat{C}$ denote the set of categories that consists of the lowest (closest to the leaves) covering category of every covering branch in T. Let $\hat{Q}$ denote a set of queries constructed by selecting arbitrarily from every category in $\hat{C}$ a single query that it covers. Observe that $\hat{Q}$ is an r-weak

partition. We denote by $\hat{V} = \{v \in V \mid q_v \in \hat{Q}\}$ the set of vertices that corresponds to the queries in $\hat{Q}$, and denote by $\hat{G}$ the sub-hypergraph of G induced by $\hat{V}$. R_3 computes $\hat{G}$ and applies over it the algorithm from Lemma 3.6, producing an *IS* S, which is the final output.

For simplicity, we ignore rounding issues, as rounding the parameters to the nearest rational fraction has a negligible effect, and the number of vertices, n, can be manipulated by adding vertices that are connected to all other vertices.

Hardness bounds. The construction above implies the following hardness bounds.

Theorem 4.1. *The $OCT^r(\mathcal{T}_{\alpha,\beta})$ problem, with constant $\alpha \in (0,1]$ and $\beta > \frac{1}{2}$, cannot be approximated below a $\tilde{\Theta}(n^{\frac{1}{r+1}})$ factor, unless ZPP = NP. For $OCT^r(\mathcal{T}_{\alpha,1})$ this is improved to $\tilde{\Theta}(n)$. For $r = 1$ these bounds hold for the assumption $P \neq NP$. Moreover, $OCT^1(\mathcal{T}_{\alpha,1})$ with maximum query size $d = \Theta(1)$ is hard to approximate below a $\tilde{\Theta}(\alpha d)$ factor, unless BPP = NP. Lastly, $OCT^1(\mathcal{T}_{1,1})$ is NP-hard even when all queries are of size exactly 3. The bounds for $OCT^1(\mathcal{T}_{\alpha,1})$ hold even when query intersections are of cardinality at most 1.*

We explain below the intuition underlying the reduction and the proof outline, followed by the formal proof.

Intuition. When $\beta = 1$, there are no joint elements, and each query consists of all the relevant edge elements along with padding elements that ensure its size is exactly n^r. In the Exact variant, every covering branch in T covers exactly one query, and this independently-covered set of queries corresponds to a set of vertices that is independent in G. Therefore, if T covers $poly(n)$ queries, we can find an *IS* of the same size in G.

When relaxing the precision threshold, α, it becomes possible for the same branch to cover multiple queries. As we want to select one query from each branch to ensure independence, it may no longer be the case that the number of covering branches is of the same order as the solution. Nevertheless, on every covering branch, the covering category C closest to the root must contain all elements of all covered queries on the same branch. If there are many such queries, then C would not satisfy the precision requirement. Intuitively, a branch can cover no more than $O(\frac{1}{\alpha}) = O(1)$ queries. It follows that the set of independently-covered queries, $\hat{Q}$, is of the same order as the score of the tree in this case as well.

Matters are more complicated when the recall threshold β is also relaxed. It is no longer the case that an independently-covered set of queries corresponds to an *IS*. It is now possible for $(r+1)$ such queries to correspond to an edge in G, as the cover of at least one of these queries can avoid containing that edge-element. Without including joint elements in the reduction, the cover of every query could omit a constant fraction of the edge elements, which amounts to $O(n^{r+1})$ edge elements, such that a large independently-covered set of queries could correspond to even a very dense subgraph in G.

To that end, we show that a cover of a query must include a joint element per every omitted edge element. Since all queries share the joint elements, this hinders the ability of covers in other branches to omit edge elements. It follows that the total number of edges in a subgraph corresponding to an independently-covered set of queries contains at most $O(n^r)$ edges which is the total number of joint elements. Therefore, a tree of high score would correspond to a large vertex set which is also sparse. From this "almost *IS*" we can derive a somewhat smaller, but still polynomial-sized, *IS*, using Lemma 3.6.

Adding joint elements may allow covering more queries on a single branch, as including joint elements in a category contributes to its potential cover of all queries. However, we show that the number of covered queries by a single branch is bounded by a constant.

We ensure that the optimal *OCT* solution is of score $\tilde{\Theta}(n)$. Thus, if the approximation factor of A is low, the eventually derived *IS* is large. In particular, we ensure that the tree contains a category partition of the same score so that all bounds also hold for *OCP*. Observe that the maximum *IS* in G induces the category partition where every category covers a single query pertaining to a vertex in the set, with all covers including all of the non-joint elements and no joint elements. The categories in this partition satisfy the recall condition as narrowly

as possible. Intuitively, this construction means that, while joint elements help an algorithm to an extent, beyond that it must make progress on the *MIS* problem.

Formal proof. We next formalize the intuition described above.

Proof of Theorem 4.1. Let S' denote a maximum independent set in the input graph G. Recall the we assume inputs where $|S'| = \tilde{\Theta}(n)$. For any $v \in V$, let C_v denote the category that consists of all the non-joint elements of the query q_v (as defined in step R_1 of the *MIS* construction, at the beginning of this section). Observe that C_v covers q_v, as the cover precision is 1 and the recall is exactly β. Consider the following set of categories: $P = \{C_v \mid v \in S'\}$. Since S' is independent, we have that every element appears in at most r categories in P, which makes it an r-weak partition. By connecting all categories in P to a root, we get a tree of score $\tilde{\Theta}(n)$, which is also a lower bound on the optimal score over Q. It follows, that the score of the tree T, which A produces, is at least $\tilde{\Theta}(\frac{n}{\gamma})$.

We next prove an upper bound on the number of queries that can be covered by a single branch. We use the terminology the *cover of a query* to refer to the set of elements in its covering category. Given a covering branch, let k denote the number of queries it covers, and let C denote its highest covering category, denoting by q one of the queries C covers. We first compute a lower bound on $|C|$. Because C is the highest covering category on the branch, it contains all the elements in the covers of all k queries. Due to the recall condition, the cover of every query contains at least $\frac{\beta}{2\beta-1}n^r$ of its elements. As there are only $\frac{1-\beta}{2\beta-1}n^r$ joint elements, the number of non-joint elements of a query contained in its cover is at least

$$\frac{\beta}{2\beta-1}n^r - \frac{1-\beta}{2\beta-1}n^r = n^r$$

The number of non-joint elements in the union of all k covers may be less than kn^r because the same edge element can be in several covers. However, since a padding element is in only one cover, and an edge element can be in up to $(r+1)$ covers, we have that the number of non-joint elements in C, and, consequently, the total number of elements in C, is at least

$$|C| \geq \frac{kn^r}{r+1}$$

On the other hand, from the precision condition of the cover of q by C, we get:

$$\frac{\frac{n^r}{2\beta-1}}{|C|} \geq \frac{|C \cap q|}{|C|} \geq \alpha$$

From this, we get the upper bound

$$|C| \leq \frac{n^r}{\alpha(2\beta-1)}.$$

Combining both bounds, we get:

$$\frac{kn^r}{r+1} \leq \frac{n^r}{\alpha(2\beta-1)}.$$

Finally, it follows that

$$k \leq \frac{r+1}{\alpha(2\beta-1)} = O(1).$$

Since the number of queries covered by any branch is bounded by a constant, we have that $\hat{Q}$ (the set of independently-covered queries described in step R_3 of the reduction algorithm R), and, consequently, $\hat{V}$ (the vertex set that corresponds to $\hat{Q}$, also defined in step R_3), are of the same order as $S(T)$, which is $\tilde{\Omega}(\frac{n}{\gamma})$.

When $\beta = 1$, we have that $\hat{V}$ is already an independent set, since, to satisfy the recall condition, covers must include all edge elements, and, with $\hat{Q}$ being independently-covered, no edge element can appear in the covers of

$(r+1)$ queries in $\hat{Q}$. Therefore, following Theorem 3.3, we have $\gamma = \tilde{\Theta}(n)$. The improved results for $r = 1$ follow from Theorem 3.4. The bounds for the case of $r = 1$ where queries are also of bounded size d, follow from an analogous proof, where the padding elements ensure that queries are of size d instead of n.

For $\beta < 1$, we make the following observation: for every edge e in $\hat{G}$, for at least one of the $(r+1)$ vertices in e, the cover of its corresponding query in $\hat{Q}$ does not contain the edge element corresponding to e. Another important observation is that the number of joint elements in the cover of every query is at least the number of the query's edge elements not in the cover. This is because a cover consisting of all non-joint elements matches the recall threshold exactly, and removing any edge element from the cover necessitates its replacement by a joint element. It follows that the total number of edge-elements omitted from covers of $\hat{Q}$ is at most the number of joint elements which is $\Theta(n^r)$. Therefore, $\hat{G}$ is a hypergraph of size $\tilde{\Omega}(\frac{n}{\gamma})$ with $O(n^r)$ edges. From Lemma 3.6 (recall that, in our context, G is $(r+1)$-uniform, and not r-uniform), we get that the size of the resulting independent set is

$$|S| = \tilde{\Omega}(\frac{(\frac{n}{\gamma})^{r+1}}{n^r}) = \tilde{\Omega}((\frac{n}{\gamma^{r+1}})^{\frac{1}{r}})$$

The $\tilde{\Omega}(n^{\frac{1}{r+1}})$ bound on γ follows from Theorems 3.3 and 3.4. □

4.2 General threshold parameters

We have examined so far the hardness of various special cases of $OCT^r(\mathcal{T}_{\alpha,\beta})$ where the recall threshold is $\beta > \frac{1}{2}$. In particular, we proved for $\frac{1}{2} < \beta < 1$ inapproximability of $\tilde{\Theta}(n^{\frac{1}{r+1}})$. We next devise a more involved construction to show that this result extends to $\beta \leq \frac{1}{2}$ and also provide more general bounds for polynomially small threshold parameters. We note that since the modified construction is randomized, the bound derived for $r = 1$ does not hold under the weaker assumption of $P \neq NP$, unlike in the first construction.

Modifications. To facilitate a precise discussion, first recall that, given a subset of queries Q', the multiplicity $M_{Q'}(e) = |\{q \in Q' \mid e \in q\}|$ of an element e in Q' is the number of queries in Q' that e appears in. The reduction used for Theorem 4.1 becomes ineffective because in the OCT instance constructed by R, the set of joint elements makes up a $(1-\beta)$-fraction of every query, and for $\beta \leq 1/2$ the set of joint elements becomes large enough, such that a category, that consists exactly of this set, covers all queries, yielding the optimal tree score. To fix this, we need to alter the construction such that joint elements are not shared by all queries. We want to limit the number of joint elements with high multiplicity in any large query set (we will formalize this high-level statement with concrete thresholds in Lemma 4.5), to make it hard for a single branch to cover it while retaining properties essential for the hardness proof.

Concretely, we want any single joint element to be shared by many queries, and for a $(1-\beta)$-fraction of every query to consist of joint elements, so that the tree that corresponds to the optimal MIS solution narrowly exceeds the recall requirements. This requires using more joint elements. However, having more joint elements can make $\hat{G}$ less sparse, reducing the size of the produced IS. To achieve these desired properties while minimally increasing the number of joint elements, we devise a randomized reduction. Moreover, we parameterize it to efficiently capture sub-constant ranges of α and β, to aim for a slower decay in the hardness bound, as these thresholds are decreased.

Generalized reduction from MIS. Our revised MIS algorithm denoted by R' consists of a sequence of three procedures, R'_1, R'_2 and R'_3. To avoid a convoluted presentation, we reuse some of the notation, initially defined in the context of the first algorithm R.

(1) Given an $(r+1)$-uniform hypergraph, $G = (V, E)$, R'_1 transforms it into an instance $Q = \{q_v \mid v \in V\}$ of $OCT^r(\mathcal{T}_{\alpha,\beta})$. Each query, q_v, contains all the edge elements that correspond to edges incident to v in G, and as many unique padding elements as necessary, such that the number of non-joint elements in every

query is exactly n^r. Finally, we distribute $(\frac{1}{\beta}-1)\frac{\log^3 n}{\alpha}n^r$ distinct joint elements to queries via the following randomized scheme. We draw uniformly randomly $(\frac{1}{\beta}-1)n^r$ partitions of Q into $\frac{\log^3 n}{\alpha}$ subsets, each of size $\frac{\alpha n}{\log^3 n}$. Let $\hat{\mathbb{P}}$ denote this set of partitions. In every partition, $\hat{p} \in \hat{\mathbb{P}}$, every set, $\hat{s} \in \hat{p}$, in that partition is assigned a distinct joint element to be included in all queries in the set. Note that the size of each query is now exactly $\frac{n^r}{\beta}$.

(2) The procedure R'_2, same as R_2, runs over Q the given $OCT^r(\mathcal{T}_{\alpha,\beta})$ algorithm A with an approximation guarantee factor of γ. Let T denote the category tree A outputs.

(3) Finally, R'_3 is the same as R_3, except for the following modification: if there is a branch in T that covers more than $\tilde{\Theta}(\frac{1}{\alpha})$ queries, then it outputs DO NOT KNOW, and otherwise proceeds as R_3 to produce an *IS* S.

Generalized hardness bounds. We now state the approximation bounds implied by the revised reduction, followed by the intuition underlying the proof.

Theorem 4.2. *The $OCT^r(\mathcal{T}_{\alpha,\beta})$ problem cannot be approximated below a $\tilde{\Theta}((\alpha^{(r+2)}\beta n)^{\frac{1}{r+1}})$ factor, unless ZPP = NP.*

Intuition. The most significant component in the proof is the following technical Lemma.

Lemma 4.3. *W.p. $1-o(1)$ (over the choices of partitions in $\hat{\mathbb{P}}$) the maximum number of queries in Q a single branch (in any tree) can cover is $\tilde{O}(\frac{1}{\alpha})$.*

We wish to show that precision cannot be maintained past a certain number of covered queries on a branch. We use the term *relevant cover* of a query q, to refer to the intersection of q with its covering category C, with the *relevant cover size* being $|q \cap C|$. One must be careful in selecting the query for which the precision condition is invoked, to derive a tight bound. On the one hand, we aim to select a query covered close to the root, so that its covering category contains the covers of many other queries. On the other hand, we want to select a query whose relevant cover is small. Thus, we first prove that for any branch, there exists a query q covered by C, such that at least a constant fraction of the covered queries on the branch are covered by C or a lower category, and that the average relevant cover size of these queries is smaller than the relevant cover size of q by at most a logarithmic factor.

Lemma 4.4. *Given a branch B that covers k' queries, there exists a query q covered by a category C in B, with the following two properties:*

(1) *the set of queries, Q_k, covered by C or categories below C is of cardinality $k = \Theta(k')$.*

(2) *let $d \in [1, \frac{1}{\beta}]$ denote the value for which the average relevant cover size of queries in Q_k is $n^r d$, then the relevant cover size of q is at most $(2\log n)n^r d$.*

Given q and C as in Lemma 4.4, we derive from the precision condition an upper bound on $|C|$. On the other hand, C contains the union of the k covers of the queries in Q_k, and we show that for $k = \tilde{\omega}(\frac{1}{\alpha})$, the union of the k covers, and thereby C must contain many elements, beyond the upper bound, resulting in a contradiction. The key to proving that C contains many elements is bounding the multiplicity of the joint elements in Q_k. If all elements had constant multiplicity, then an α precision threshold implies that, when the relevant covers are on average of roughly the same size as the relevant cover of q (which is the case following Lemma 4.4), the number of covered queries is $O(\frac{1}{\alpha})$. To that end, we show that the multiplicity of almost every joint element in Q_k does not exceed $\tilde{O}(\frac{k}{\alpha})$.

Lemma 4.5. *For any set Q_k of $k = \omega(\frac{\log^3 n}{\alpha})$ queries, w.p. $1-o(1)$, there are at most $\frac{n^r}{2}$ partitions in $\hat{\mathbb{P}}$ where a joint element is assigned to more than $\theta = \frac{\alpha k}{8\log n}$ queries in Q_k.*

The proof of Lemma 4.5 consists of a combination of probabilistic arguments. We first prove that this θ bound on the number of partitions holds for a uniformly randomly selected set of k queries w.p. $1 - o(n^{-k})$. Then, by using a union bound argument, it will follow that this bound holds for any selection of k queries w.p. $1 - o(1)$.

To prove the bounds of Lemma 4.5 for a randomly selected set Q_k of k queries, observe that a joint element is shared by polylogarithmically less than a $\frac{1}{\alpha}$-fraction of the queries in Q. Therefore, its expected multiplicity in Q_k would constitute the same fraction. To bound the probability of significantly deviating from this expectation, we show that the multiplicity of any joint element in Q_k is a hypergeometric random variable, and use a tail bound. Following a different union bound argument, this bound on the probability is extended over every joint element assigned in a given partition in $\hat{\mathbb{P}}$. Finally, since the partitions in $\hat{\mathbb{P}}$ are chosen independently, we use a Chernoff bound to derive an upper bound, that holds with high probability, on the number of partitions in $\hat{\mathbb{P}}$ in which a joint element with high multiplicity was assigned. We show that if these deviations occur sufficiently rarely, as stated in Lemma 4.5, then the cardinality of C increases as a function of k, deriving the bound $k = \tilde{O}(\frac{1}{\alpha})$.

Formal proofs. We conclude this subsection with the formal proofs of the theorems and lemmas presented above.

Proof of Theorem 4.2. Given Lemma 4.3, the proof of Theorem 4.2 is mostly analogous to the proof of Theorem 4.1. Hence, we only highlight here the modified computations.

First, following the exact same arguments, the score of the tree T is $\tilde{\Omega}(\frac{n}{\gamma})$. When the $\tilde{O}(\frac{1}{\alpha})$ bound on the number of covered queries by a single branch stated in Lemma 4.3 holds, then the number of vertices in $\hat{V}$ is $\tilde{\Omega}(\frac{\alpha n}{\gamma})$. Following the same arguments as for the first reduction, the number of edges in $\hat{G}$ is upper bounded by the total number of joint elements which is $\tilde{O}(\frac{n^r}{\alpha\beta})$. The lower bound on the independent set follows from Lemma 3.6:

$$|S| = \tilde{\Omega}((\frac{(\frac{\alpha n}{\gamma})^{r+1}}{\frac{n^r}{\alpha\beta}})^{\frac{1}{r}}) = \tilde{\Omega}((\frac{\alpha^{r+2}\beta n}{\gamma^{r+1}})^{\frac{1}{r}})$$

From Theorem 3.3, we get $\gamma = \tilde{\Omega}((\alpha^{(r+2)}\beta n)^{\frac{1}{r+1}})$.

Finally, observe that the bound in Lemma 4.3 on every branch in T is a sufficient condition to guarantee the approximation factor we derived for R' as a function of γ. When this bound does not hold for some branch in T, which Lemma 4.3 proves happens with probability $o(1)$, R' can always detect it by examining the set of queries covered by each branch and output DO NOT KNOW. Therefore, R' is a ZPP algorithm. □

Proof of Lemma 4.4. Given a branch B that covers the set $Q_{k'}$ of k' queries, we define d' as the value for which the average relevant cover size of queries in $Q_{k'}$ is $n^r d'$, and q' is defined as the query in $Q_{k'}$ covered by the category C' closest to the root (ties are broken arbitrarily). We use an iterative procedure to find q with the stated properties.

We first set $k_0 = k'$, $d_0 = d'$, $Q_{k_0} = Q_{k'}$, $q_0 = q'$ and $C_0 = C'$. In the i-th iteration, we examine the set $Q_{k_{i-1}}$ of k_{i-1} queries of average relevant cover size $n^r d_{i-1}$. If the relevant cover of q_{i-1} is at most $(2\log n)n^r d_{i-1}$, we set $q = q_{i-1}$ (and, consequently, $C = C_{i-1}$ and $Q_k = Q_{k_{i-1}}$) and we are done (we will promptly prove that Q_k is sufficiently large). Otherwise, we set q_i to be the query covered closest to the root of the queries in $Q_{k_{i-1}}$ whose relevant cover size does not exceed $n^r d_{i-1}\log n$, and C_i is set to be the category that covers q_i. Q_{k_i} is set to be the subset of queries in $Q_{k_{i-1}}$ covered by C_i or categories below it. Accordingly, k_i is the cardinality of Q_{k_i}, and d_i is set such that the average relevant cover size of queries in Q_{k_i} is $n^r d_i$.

Observe that, if the stopping condition is not met, it follows that d_i is smaller than $\frac{d_{i-1}}{2}$. Since the average relevant cover size, due to the recall condition, cannot be lower than n^r and is at most $O(poly(n))$, it follows that there are at most $\log d' = O(\log n)$ iterations before we get to the minimal average relevant cover size, where the stopping condition is necessarily met. Moreover, observe that the number of queries in $Q_{k_{i-1}}$ whose relevant

cover size does exceed $n^r d_{i-1} \log n$ is at most $\frac{k_{i-1}}{\log n}$. Therefore, there are at most $\frac{k_{i-1}}{\log n}$ queries in $Q_{k_{i-1}}$ covered by categories above C_i, implying $k_i \geq k_{i-1}(1 - 1/\log n)$. Putting everything together, it follows that the number of queries in Q_k is at least

$$k \geq k'(1 - 1/\log n)^{O(\log n)} \geq k'(1/e)^{O(1)} = \Theta(k').$$

note that we have used the fact that $(1 - 1/\log n)^{\log n}$ approaches $1/e$ as n tends to infinity. □

Proof of Lemma 4.5. We first show that this bound holds for a uniformly randomly selected set Q_k of k queries with probability at least $1 - o(n^{-k})$. Since there are at most $\binom{n}{k} < n^k$ sets of this cardinality, from the union bound it would follow that this holds for every set of k queries with probability $1 - o(1)$.

We say that an element has high multiplicity when its multiplicity in Q_k exceeds θ. We next bound the probability that a joint element e, which was assigned to any given set $\hat{s} \in \hat{p}$ in any given partition $\hat{p} \in \hat{\mathbb{P}}$ has high multiplicity $M_{Q_k}(e)$ in Q_k, using the tail bound on the hypergeometric distribution from Lemma 3.9. Then, by a union bound argument, we bound the probability of this being the case for any joint element assigned to any set in $\hat{p}$.

Observe that, in general, when proving the θ bound in Lemma 4.5 for a randomly selected Q_k, the probabilities are over the drawing of the partitions as well as the drawing of Q_k. However, due to symmetry, when bounding the multiplicity of a joint element in Q_k, we can first fix the partition $\hat{p}$ and the set of queries $\hat{s}$ to which a joint element e is assigned.

Selecting Q_k uniformly randomly is equivalent to uniformly drawing k queries from Q without replacement, and since all sets in any partition are of size exactly $\frac{\alpha n}{\log^3 n}$, we have that the number of times a query from $\hat{s}$ was drawn, which equals $M_{Q_k}(e)$, is a hypergeometric random variable (Definition 3.7) $X \sim H(n, \frac{\alpha n}{\log^3 n}, k)$.

Hence, we can bound the probability of e having high multiplicity, using Lemma 3.9. We set $u = \frac{\alpha}{\log^3 n}$ and

$$t = \frac{\theta}{k} - u = (\frac{\alpha}{8 \log n} - \frac{\alpha}{\log^3 n}),$$

and assume $k = \omega(\frac{\log^3 n}{\alpha})$. Note that $u = o(t)$, and, in particular, $\frac{u}{t} = \Theta(\frac{1}{\log^2 n})$. Moreover, $kt = \omega(\log^2 n)$. Applying the tail bound, we get:

$$\begin{aligned}
&\Pr(X \geq \theta) = \Pr(X \geq (u+t)k) \\
&\leq ((\frac{u}{u+t})^{u+t}(\frac{1-u}{1-u-t})^{1-u-t})^k \leq ((\frac{u}{u+t})^t(\frac{1-u}{1-u-t}))^k \\
&\leq ((\frac{u}{t})^t(1 + \frac{t}{1-u-t}))^k = ((\frac{u}{t})^t(1 + \frac{t}{1-u-t})^{\frac{2t}{2t}})^k \\
&\leq ((\frac{u}{t})(1+2t)^{\frac{2}{2t}})^{kt} = \Theta(\frac{u}{t}e^2)^{kt} \leq \Theta(\frac{1}{\log^2 n})^{kt} \\
&= o(\frac{1}{\log^2 n})^{\log^2 n} = o(\frac{\alpha}{n^4})
\end{aligned}$$

Note that we have used the fact that as n tends to infinity, $2t$ tends to 0, and, thus, $(1+2t)^{\frac{1}{2t}})$ approaches e.

From the union bound, we get that the probability, p, of the event where at least one of the $\tilde{\Theta}(\frac{1}{\alpha})$ joint elements assigned to the sets in $\hat{p}$ has high multiplicity is $p = o(\frac{1}{n^4})$.

Let $l = (\frac{1}{\beta} - 1)n^r < \frac{n^r}{\beta}$ denote the number of partitions in $\hat{\mathbb{P}}$. The number of partitions, where an element of high multiplicity in Q_k is assigned, is a binomial random variable (Definition 3.8) $Y \sim B(l, p)$. The expectation of

Y is

$$\mu = lp = o(\frac{n^r}{\beta} \cdot \frac{1}{n^4}) = o(\frac{n^{r-4}}{\beta}).$$

By applying the Chernoff bound (Lemma 3.10), and setting

$$\delta = \frac{n^r}{2\mu} - 1 = \omega(\beta n^4),$$

we get a bound on the probability of this high multiplicity event occurring for more than $n^r/2$ partitions:

$$\begin{aligned}\Pr(Y > \frac{n^r}{2}) &= \Pr(Y > (1+\delta)\mu) \\ &< (\frac{e^\delta}{(1+\delta)^{1+\delta}})^\mu = O((\frac{e}{\delta})^{\delta\mu}) = O((\frac{e}{\delta})^{\frac{n^r}{2}}) \\ &= o((\frac{e}{\beta n^4})^{\frac{n^r}{2}}) = o((\frac{1}{n^2})^{\frac{n^r}{2}}) = o(n^{-n^r}) = o(n^{-k})\end{aligned}$$

□

Proof of Lemma 4.3. We assume here, for consistency, the same notation and definitions as in Lemmas 4.5 and 4.4. Given any branch B that covers k' queries, we invoke the precision condition of a query q, covered by a category C, with the properties stated in Lemma 4.4. The precision condition implies

$$\frac{(2\log n)n^r d}{|C|} \geq \frac{|C \cap q|}{|C|} \geq \alpha,$$

from which it follows that

$$|C| \leq \frac{2\log n}{\alpha} n^r d.$$

On the other hand, C contains the union of the covers of the $k = \Theta(k')$ queries, Q_k, covered by it, or by categories below it. From Lemma 4.4, we have that the sum of the relevant covers sizes of Q_k is $kn^r d$. This sum of the covers is also the sum of the multiplicities of all the elements in the union of the covers.

Assume, for the sake of contradiction, that $k = \omega(\frac{\log^3 n}{\alpha})$. Then, by Lemma 4.5, we have that, with probability $1 - o(1)$, there are at most $\frac{n^r}{2} \leq \frac{n^r d}{2}$ partitions in $\hat{\mathbb{P}}$ in which there is an element has high multiplicity in Q_k. In all those cases we can assume the worst-case where there is a joint element with multiplicity k (observe that for any given partition the sum of multiplicities of the corresponding joint elements cannot exceed k). Therefore, when excluding all the elements with high multiplicity, we have that the sum of all k covers, and the sum of the multiplicities, is at least

$$kn^r d - \frac{n^r d}{2} k = \frac{kn^r d}{2}.$$

For any of the remaining elements we have that its multiplicity is at most $\theta = \frac{\alpha k}{8\log n}$. This multiplicity bound, of course, extends to edge and padding elements that have constant multiplicity which is $o(\theta)$. Therefore, the number of (distinct) elements in C is at least

$$|C| \geq \frac{kn^r d}{2\theta} = \frac{4\log n}{\alpha} n^r d.$$

This contradicts the upper bound of $\frac{2\log n}{\alpha} n^r d$, and along with the fact that k' is of the same order as k, implies that, with probability at least $1 - o(1)$ over the random partitions in $\hat{\mathbb{P}}$, there can be no branch on a category tree for Q that covers $\tilde{\omega}(\frac{1}{\alpha})$ queries. □

5 HARDNESS OF OTHER VARIANTS

So far we have proven hardness of $OCT^r(\mathcal{T}_{\alpha,\beta})$. In this section, we provide approximation hardness bounds for the remaining OCT variants, via reductions from $OCT^r(\mathcal{T}_{\alpha,\beta})$ and Theorems 4.1 and 4.2.

We first show that the $\tilde{\Theta}(n^{\frac{1}{r+1}})$ bound of $OCT^r(\mathcal{T}_{\alpha,\beta})$ with constant thresholds extends to the threshold versions of Jaccard and F_1 scores, with similar inapproximability for sub-constant thresholds as well. We then use these results to derive bounds for the cutoff versions of these functions, which only differ for $\delta = o(1)$.

We formulate our proofs schematically, such that they may be applied to threshold and cutoff variants of other functions.

For threshold functions, we derive the following bounds.

Theorem 5.1. *The variants $OCT^r(\hat{J}_\delta)$ and $OCT^r(\hat{F}_{1(\delta)})$ cannot be approximated below a $\tilde{\Theta}((\delta^{r+3}n)^{\frac{1}{r+1}})$ factor, unless ZPP = NP. For $r = 1$, we have $\tilde{\Theta}(\sqrt{n})$ inapproximability, assuming $P \neq NP$, for $OCT^r(\hat{J}_\delta)$ with $\delta > \frac{1}{2}$ and $OCT^r(\hat{F}_{1(\delta)})$ with $\delta > \frac{2}{3}$.*

Proof. Let $\mathcal{S}$ denote any given function in $\{J, F_1\}$ and let $\hat{\mathcal{S}}_\delta$ denote its corresponding threshold version in $\{\hat{J}_\delta, \hat{F}_{1(\delta)}\}$. We define I_α as the set of all query-category pairs, q and C, such that $r(q, C) = 1$ and $p(q, C) = \alpha$, and analogously define I_β as the set of all query-category pairs such that $p(q, C) = 1$ and $r(q, C) = \beta$. Let $M_\alpha(\mathcal{S})$ denote the score $\mathcal{S}(q, C)$ of any $(q, C) \in I_\alpha$, and let $M_\beta(\mathcal{S})$ denote the score $\mathcal{S}(q, C)$ in the latter case (we will promptly show that this score is well defined and uniform across all pairs in the same set). Finally, given the parameter values α and β, let $M(\mathcal{S}) = M_{\alpha,\beta}(\mathcal{S}) = \max\{M_\alpha(\mathcal{S}), M_\beta(\mathcal{S})\}$.

For both variants, we use the same reduction from $OCT^r(\mathcal{T}_{\alpha,\beta})$ where the input Q is not modified at all. We set, however, different threshold parameters for the original $OCT^r(\mathcal{T}_{\alpha,\beta})$ instance, depending on which variant we reduce to.

The proof for each variant consists of two arguments, and relies on ensuring that $M(\mathcal{S}) = M_\beta(\mathcal{S}) = \delta$. For the two particular functions examined here, we will also ensure that $M_\alpha(\mathcal{S}) = \delta$, which is, in general, preferable, as we want to use the highest possible value of α ($M_\alpha(\mathcal{S})$ is a monotonically increasing function of α), such that the bound for the set of $OCT^r(\mathcal{T}_{\alpha,\beta})$ inputs we reduce from is stricter.

The first argument is that optimal score over the input Q w.r.t. $\hat{\mathcal{S}}_\delta$ is of at least the same order as the optimal score w.r.t. $\mathcal{T}_{\alpha,\beta}$. To that end, when reducing from $OCT^r(\mathcal{T}_{\alpha,\beta})$, we restrict ourselves to the hard set of inputs of $OCT^r(\mathcal{T}_{\alpha,\beta})$ mapped to by our reduction from hard inputs of MIS in the proof of Theorems 4.1 and 4.2. We showed that for any such input, Q, there exists a category tree (not necessarily optimal), we denote here by T'_Q, whose leaves induce a category partition covering $\tilde{\Theta}(n)$ queries, each covered with precision 1 and recall β. It follows that the score of T'_Q, w.r.t. $\mathcal{S}$, for each query is $M_\beta(\mathcal{S})$. We will show that, for our choice of threshold parameters, $M_\beta(\mathcal{S})\} = \delta$, and hence T'_Q is also of score $\tilde{\Theta}(n)$, w.r.t. $\hat{\mathcal{S}}_\delta$. Note that leveraging the fact that the covers are of precision 1 is essential since in general the same score could have hypothetically been achieved over the $OCT^r(\mathcal{T}_{\alpha,\beta})$ instance, such that in every cover both the precision and recall equaled the threshold values, however, this would not imply that the covers are each of score $\mathcal{S}(q, C) \geq \delta$.

The second argument is that the score of any tree, w.r.t. $\hat{\mathcal{S}}_\delta$, cannot exceed its score w.r.t. $\mathcal{T}_{\alpha,\beta}$. This, along with the first argument, would imply the hardness bound. To that end, observe, that, since $\mathcal{S}$ is a monotonically increasing function of both the precision and the recall, for a given query-category pair, q and C, the highest score $\mathcal{S}(q, C)$ that can be achieved, such that $\mathcal{T}_{\alpha,\beta} = 0$, occurs when either the precision is 1 and the recall is infinitesimally smaller than β, or the recall is 1 and the precision is infinitesimally smaller than α. Therefore, for any such case, the score $\mathcal{S}(q, C)$ would be below $M(\mathcal{S})$. Ensuring that $M(\mathcal{S}) = \delta$, implies the argument.

Specifically, we have, when $r(q,C) = 1$ and $p(q,C) = \alpha$, that $q \cup C = C$. Therefore, in this case, for $\mathcal{S} = J$, it follows that

$$M_\alpha = J(q,C) = \frac{|q \cap C|}{|q \cup C|} = \frac{|q \cap C|}{|C|} = p(q,C) = \alpha.$$

Similarly, when $p(q,C) = 1$ and $r(q,C) = \beta$, then $q \cup C = q$, and

$$M_\beta = \frac{|q \cap C|}{|q|} = r(q,C) = \beta.$$

Therefore, we reduce to $OCT^r(\hat{J}_\delta)$ from $OCT^r(\mathcal{T}_{\delta,\delta})$, which implies, $M(J) = M_\beta(J) = \delta$, as required. Following Theorem 4.2, the inapproximability factor for $OCT^r(\hat{J}_\delta)$ is

$$\tilde{\Theta}((\alpha^{(r+2)}\beta n)^{\frac{1}{r+1}}) = \tilde{\Theta}((\delta^{r+3} n)^{\frac{1}{r+1}})$$

Similarly, for $\mathcal{S} = F_1$, when $r(q,C) = 1$ and $p(q,C) = \alpha$, we have

$$M_\alpha = F_1(q,C) = 2\frac{\alpha}{1+\alpha}.$$

We also have, analogously, that $M_\beta = 2\frac{\beta}{1+\beta}$.

We, therefore, reduce to $OCT^r(\hat{F}_{1(\delta)})$ from $OCT^r(\mathcal{T}_{\delta',\delta'})$, where $\delta' = \frac{\delta}{2-\delta}$, which implies, $M(F_1) = M_\beta(F_1) = \delta$, as required. For any constant $\epsilon \in (0, 0.5)$ we have that when $\delta \in [\epsilon, 1-\epsilon]$ then $\frac{\epsilon}{2} < \delta' < \frac{1}{1+\epsilon}$, thus $\delta' = \Theta(\delta)$. Moreover, for $\delta = o(1)$, we have $\delta' \approx \frac{\delta}{2} = \Theta(\delta)$, as well. Hence, the inapproximability bound of $OCT^r(\hat{F}_{1(\delta)})$ is also $\tilde{\Theta}((\delta^{r+3} n)^{\frac{1}{r+1}})$.

The improved hardness for $r = 1$ and a sufficiently high δ parameter follows from Theorem 4.1. □

Finally, we provide bounds for cutoff functions, that follow from Theorem 5.1.

Theorem 5.2. *The variants $OCT^r(\bar{J}_\delta)$ and $OCT^r(\bar{F}_{1(\delta)})$, with $\delta \in [0,1)$, have $\tilde{\Theta}((\delta^{2r+4} n)^{\frac{1}{r+1}})$ inapproximability, unless ZPP = NP. For $r = 1$, we have $\tilde{\Theta}(\sqrt{n})$ inapproximability, assuming $P \neq NP$, for $OCT^r(\bar{J}_\delta)$ with $\delta > \frac{1}{2}$ and $OCT^r(\bar{F}_{1(\delta)})$ with $\delta > \frac{2}{3}$.*

Proof of Theorem 5.2. For both cutoff functions, we use a reduction from its corresponding threshold variant, with the same threshold parameter δ, where we do not modify the input. By definition, the score of any tree in the cutoff variant cannot exceed its score in the threshold variant. On the other hand, the score of any tree for the cutoff instance is at least a δ-fraction of its score for the threshold instance. In particular, while the score of the optimal solution for the cutoff instance can be lower, it is, nevertheless, at least a δ-fraction of the score of the optimal solution for the threshold instance. Therefore, the approximation factor can be lower by at most a δ factor, yielding the stated bound. □

6 ALGORITHMS

We next present algorithms, that despite the harsh hardness bounds of the general case, can provide much improved guarantees for various special cases, based on the values of more granular problem parameters. For the Exact variant with $r = 1$, we also provide an algorithm, that we showed to solve the problem optimally on all examined inputs, as detailed in [5].

Concretely, we first describe procedures that, for the cost of logarithmic factors, allow one to assume that the input is unweighted and that all queries are of roughly the same cardinality. These assumptions are used in some of our devised algorithms.

We then provide the algorithms for the Exact variant that have the same performance ratio as the underlying *MIS* algorithm they leverage. This both allows to use *MIS* exact solvers that are shown in practice to run efficiently, but also, on the theoretical side, to guarantee an approximation factor, that may be much improved compared to the general case, based on a parameter related to the intersections of the queries.

Finally, we provide approximation algorithms for multiple variants, that are parameterized by the maximum multiplicity of the instance. Concretely, we show that one can achieve an approximation factor of roughly the same order as the multiplicity, by an efficient randomized algorithm.

We note that all similarity threshold parameters are assumed to be constants.

6.1 Reducing to unweighted instances of near-uniform query size

We first describe two standard procedures, that each, for the cost of a logarithmic factor, can reduce any problem instance to an unweighted instance and to an instance where all queries are of roughly the same cardinality, respectively. We will use these procedures in some, but not all, of our algorithms.

We first explain how to eliminate weights at the cost of a $O(\log n)$ factor. Let q' denote the query of the highest weight, w_{max}. At the cost of a constant factor, we can ignore all queries whose weight is less than $w_{low} = \frac{w_{max}}{2n}$, since if the total weight of these queries is a constant fraction of the weight covered by the optimal solution, then a solution that covers only q' (by having a single category that is identical to q') achieves constant approximation. Next, for the remaining queries, we can partition these into sets based on weight ranges that increase exponentially. Specifically, consider all the sets S_i that contain all queries of weights in the range $[w_{low} * 2^i, w_{low} * 2^{i+1})$ for $0 \leq i \leq O(\log n)$ (once the range exceeds w_{max} there are no longer non-empty sets). By a simple counting argument, at least one of these sets contains a $1/O(\log n)$-fraction of the queries covered by the optimal solution (note that not necessarily all queries can be covered by an optimal solution, as that would imply based on the proof of Theorem 4.1 that a large independent set always exists) . Therefore, we can solve the problem separately for each S_i, and we are guaranteed that for at least one of these instances, the optimal weight that can be covered is at least a $1/O(\log n)$-fraction of the optimal solution to the original instance. Finally, we can treat each separate instance as unweighted, as this will cost at most an additional factor of 2 (since in each instance the maximum weight ratio of two queries is bounded by 2).

To reduce to instances where all queries are of roughly the same cardinality, we use a slightly generalized variant of the same method. Concretely, for any constant $\epsilon > 0$, we partition the queries into the sets S_i that contain all queries of cardinality in the range $[(1+\epsilon)^i, (1+\epsilon)^{i+1})$, for $0 \leq i \leq O(\log n)$. Since we assume in our model that the size of the universe is $O(poly(n))$, $O(\log n)$ sets are indeed sufficient to account for all queries. Once again, solving the problem separately over each set S_i and selecting the solution of the highest value would lose at most a $O(\log n)$ factor. Note that, in each instance, all queries are of the same size up to a $(1+\epsilon)$ factor.

6.2 Approximation algorithm for the Exact variant

We next provide PTIME approximation algorithms for the Exact variant of (weighted) $OCT^1(\mathcal{T}_{1,1})$ and $OCP^r(\mathcal{T}_{1,1})$. Note that for *OCT* the algorithm applies when $r = 1$, while for *OCP* it applies to any constant r. The Exact variant is of special interest because it is a special case of all variants pertaining to all examined similarity functions, where the error threshold is $\delta = 1$ (or $\alpha = \beta = 1$ for $\mathcal{T}_{\alpha,\beta}$). We note that in [5] we show empirically that this algorithm can solve real-world instances optimally, where $r = 1$, using as a subroutine a modern weighted *MIS* exact solver [17]. Moreover, as mentioned, the requirement $r = 1$ is the most prevalent in practice.

In Theorem 4.1 we prove that it is *NP*-hard to approximate this variant below a $\tilde{\Theta}(n)$ factor. Hence, one cannot provide any non-trivial approximation guarantees for the general case. Nevertheless, we devise an algorithm with an optimal approximation guarantee of $\tilde{O}(\bar{D})$, where $\bar{D} \in [0, n]$, referred to as the *average (weighted) degree* of the input, is a parameter relating to the number of intersections among the queries. Formally, we define a *conflict*

as any pair of queries that intersect and neither is a subset of the other (for *OCP* only the former condition is relevant). The *degree* $d(q)$ of a query $q \in Q$ is defined as the number of conflicts in the input that contain q. The average degree is the weighted average of all query degrees in the input. That is, $\bar{D} = \frac{\sum_{q \in Q} W(q) \cdot d(q)}{\sum_{q \in Q} W(q)}$.

Algorithm 1: A_T Algorithm

Input: Q - queries, W - query weights, A_{WMIS} - WMIS Algorithm
Output: T - category tree

1 Construct a weighted graph G where the nodes are Q (with the same weights) and an edge connects every two queries that intersect without one containing the other;
2 Run A_{WMIS} over G producing a query set S;
3 Construct a tree T where the categories are S and a *root* (that contains the union of the queries in S), and the parent of every category C in S is the smallest category in $S \cup root$ that contains C;
4 **return** T;

Algorithm 2: A_P^r Algorithm

Input: Q - queries, W - query weights, A_{WMIS} - WMIS Algorithm
Output: S - category partition

1 Construct a weighted hypergraph G where the nodes are Q (with the same weights W) and the hyperedges are all the sets of $(r+1)$ queries whose intersection is not empty;
2 Run A_{WMIS} over G producing a query set S;
3 **return** S;

6.2.1 Algorithms for $OCT^1(\mathcal{T}_{1,1})$ and $OCP^r(\mathcal{T}_{1,1})$. We first describe the algorithm A_T for $OCT^1(\mathcal{T}_{1,1})$, depicted in Algorithm 1 and 2, respectively. Note that both algorithms receive as input an underlying algorithm for the weighted *MIS* problem (or the unweighted problem, if the *OCT* or *OCP* instance is unweighted as well). The first step in A_T is to construct a *conflict graph* G, which is a weighted graph whose vertices are the queries (with the weight of the vertex being the weight of the query), and the edges are the conflicts. We next run over G the approximation algorithm for the Weighted *MIS* (*WMIS*) problem in [2]. The *WMIS* problem is a generalization of the *MIS* problem, where the vertices are weighted, and the goal is to produce the independent set of the highest total weight. Note that the degree of a vertex in the G is the degree of the query and the average (weighted) degree in the graph, which is defined as the weighted average of the vertex degrees, is $\bar{D}$, hence the terminology and the values are the same. The algorithm in [2] is based on semidefinite programming, and provides an $O(\bar{D}\frac{\log\log\bar{D}}{\log\bar{D}})$ approximation guarantee. There also exist simple greedy $\tilde{O}(\bar{D})$-approximation algorithms, as mentioned in [2], and for the special case of unweighted graphs, there is an improved SDP algorithm in [6] with a $\tilde{O}(\frac{d}{\log^2 d})$ approximation guarantee, where d is the maximum degree.

Let S denote the independent set produced by the above algorithm over the conflict hypergraph. For any query $q \in S$ that is contained in at least one other query in S, we denote by $P(q) \in S$ the query that contains q and is not contained in any other query that contains q. This is well defined because there is exactly one such query. If there were (at least) two such queries, then they necessarily intersect as both contain the elements of q, and since by definition of $P(q)$ neither of the two contains the other, it follows that these two queries conflict, which is a contradiction since S is an independent set in the conflict graph.

The last step is to build the category tree T. Besides the root, which contains all elements in U, the categories in T are made up of one category per every query in S, which contains exactly the elements of the query. The root is the parent of every category that corresponds to a query that is not contained in any other query in S. Whereas, for any other category that corresponds to a query q, its parent is the category corresponding to $P(q)$.

The algorithm A_P^r for $OCP^r(\mathcal{T}_{1,1})$ is analogous, yet simpler. Concretely, a conflict is defined as any subset of $(r+1)$ of queries whose (collective) intersection is not empty. Thus, the resulting conflict graph for $r > 1$ is in fact a hypergraph (once again, the hyperedges are the conflicts). Since $r = \Theta(1)$, checking all conflicts can be done in PTIME. Importantly, the *WMIS* algorithm in [2] also applies for hypergraphs, with the same performance guarantee. Once the *WMIS* algorithm produces an independent set S, one simply outputs the category partition that consists of one category per each query in S, containing exactly the elements of that query.

It is important to note that the same generalization does not work for *OCT*, as one can show simple counterexamples of independent sets in a conflict hypergraph that cannot be transformed into a tree that covers the entire set.

Lastly, we note that, as explained above, for an additional loss of a $O(\log n)$ factor, one can reduce the problem to unweighted instances, and for $r > 1$ we show that the approximation of *OCP* can then be improved.

Theorem 6.1. *The A_T and A_P^r algorithms, with the underlying WMIS algorithm being the algorithm for solving WMIS in [2], provide an $O(\bar{D})$-approximation for $OCT^1(\mathcal{T}_{1,1})$ and $OCP^r(\mathcal{T}_{1,1})$, respectively. This factor is optimal (up to negligible factors) for $r = 1$, unless $P = NP$.*

Proof. We focus here on the proof for A_T, the algorithm for *OCT*, as the proof for *OCP* is analogous.

To prove the stated approximation factor, we show that for any independent set S in the conflict graph G of total weight $W(S)$ there exists a category tree T that covers exactly the queries of S (and thus has a score of $W(S)$ as well), and in the other direction, we show that every tree T', in which the set of covered queries is S', the set S' is also an independent set in G. This one-to-one correspondence implies that the approximation factor of A_T for *OCT* is the same as the guarantee of the *WMIS* algorithm, proving the stated factor.

The first direction is trivial, as the algorithm A_T contains the procedure that turns every independent set S into a category tree that covers S. As for the other direction, assume for the sake of contraction that a set of queries S' that is covered by some category tree T' is not an independent set in G. That means that there exist two queries $q_1, q_2 \in S'$ that conflict. Let e denote some element in their intersection. Since in the Exact variant any cover requires perfect precision and perfect recall, it follows that both queries are covered by different categories, and since neither query contains the other (otherwise it is not a conflict), it also follows that the two covering categories are on different branches. The perfect recall also implies that e appears in both covering categories. However, this implies that e appears in two different branches, which violates the copy-bound restriction, and yields a contradiction, proving the claim.

It remains to prove that this factor is optimal (for *OCP* this is true only for $r = 1$). To that end, recall that in Theorem 4.1 we proved that when all queries are of size at most $d = \Theta(1)$ it is *NP*-hard to approximate $OCT^1(\mathcal{T}_{1,1})$ below a $\tilde{\Theta}(d)$ factor. Moreover, in the proof of this claim in Theorem 4.1 these hard *OCT* instances were reduced from *MIS* instances where the maximum degree is d, and each element in the *OCT* instance either pertains to an edge in the *MIS* instance, and thus appears only in 2 queries, or it is a unique padding element and appears only in one query (to ensures every query is of size exactly d). Since in these *OCT* instances in each query there are exactly d elements, and each such element appears in at most one other query, it follows that the degree (number of conflicts) of each query is at most d, and thus also the average degree $\bar{D}$ is at most d. We have proven these instances are *NP*-hard to approximate below a $\tilde{\Theta}(\bar{D})$ factor, which proves the optimality claim.

□

The result above shows that the approximation hardness of the Exact variant is strongly dependent on the average number of intersections between queries.

6.3 Approximation algorithms for variants with bounded multiplicity

We next provide approximation algorithms parameterized by the maximum multiplicity M.

Reduction to MIS. We first show that, for small values of M, with a slightly diminished guarantee, the A_P^r algorithm, based on the MIS solver, described above for the Exact *OCP* problem, can also be applied with similar success to variants that require perfect recall but relax the precision threshold. More importantly, while we prove in Section 4 that for variants where $\beta = 1$, the problem has $\tilde{\Theta}(n)$ worst-case inapproximability even for $M = 2$, we later show that for variants where $\beta < 1$, a solution that covers at least a $\Theta(1/M)$-fraction of the total weight of the input queries can be found by a randomized PTIME algorithm.

Starting with the case where $\beta = 1$, by slight abuse of terminology and notation, we henceforth refer to a conflict according to the definition given above for the Exact variant of *OCP*, rather than *OCT*, reusing the same notation as well. That is, a conflict is any pair of queries that intersect. This is due to the fact that we prove that the A_P^r algorithm, described above for the Exact variant of *OCP*, provides the same guarantees up to a factor of M, and then we prove that, for bounded multiplicity, the optimal solution for *OCT* must contain a partition of categories that cover queries constituting at least a $\Omega(1/\log n)$-fraction of the total covered weight. Therefore, the A_P^r *OCP* algorithm works for *OCT* as well, with an extra logarithmic loss factor in the guarantee.

Note that for $M = O(1)$ the guarantees are roughly the same, up to negligible factors. Moreover, another advantage of our approach is that the queries that are guaranteed to be covered are covered such that both the precision and recall scores are perfect (the solution may also cover more queries beyond the approximation guarantee, but only a fraction of the queries that corresponds to the worst-case approximation factor is guaranteed to be covered with by identical categories with perfect recall and precision).

Theorem 6.2. *The A_P^r algorithm, with the underlying WMIS algorithm in [2], provide $O(\bar{D}M \log n)$- and $O(\bar{D}M)$-approximation for $OCT^1(\mathcal{T}_{\alpha,1})$ and $OCP^r(\mathcal{T}_{\alpha,1})$, respectively. Moreover, these guaranteed fractions of queries are covered by the above algorithms with both precision and recall scores of* 1.

Proof. The proof for *OCP* is based on the following key claim: the maximum number of queries that can be covered by a single category is $O(M)$. This implies that there exists a solution whose score is a $\Omega(1/M)$-fraction of the optimal score where each category is identical to some query. Specifically, this solution corresponds to selecting only the query of the highest weight that is covered by each category and removing from the category all the items that are not in this query. Note that each such query is now covered with perfect precision (as well as perfect recall), therefore we can apply the algorithm for the Exact variant (for which the optimal solution is at least as good as the solution proven above to exist), and we lose only an $O(M)$ factor in the approximation guarantee, compared to all the factors stated in Theorem 6.1.

To prove that at most $O(M)$ queries can be covered by the same category, let S_k of cardinality k denote the set of queries covered by a given category. When $\beta = 1$ the category must contain the union of all the covered queries in S_k. Let $q' \in S_k$ denote the query of the lowest cardinality, $|q'| = d'$. The sum of cardinalities of all queries in S_k is at least kd'. Since each element can appear in at most M queries the cardinality of the union of all queries in S_k is at least kd'/M. However, due to the precision requirement of covering q', we get that the size of the covering category is bounded by $d'\alpha$. Therefore, $k \leq M/\alpha$, implying the result.

To get the bound for OCT, we first prove the following similar, yet more general, claim: If C covers a query q of size d, then, for any constant $\epsilon > 0$, the number of queries of size at least ϵq covered in the subtree rooted on C is $O(M)$. To prove this claim, let S_k of size k denote the set of queries covered in the subtree rooted on C. Following, the same argument as for the claim above, the number of elements in C (which contains all the elements in all

queries in S_k) is at least $\epsilon kd/M$. And also as in the proof of the claim above, due to the precision requirement on covering q, the number of elements in C is at most d/α, implying that $K \leq M/(\epsilon\alpha)$.

To see how the above claim implies the bound for OCT, consider a partition of all the queries covered by any given branch in the optimal *OCT* solution, into $O(\log n)$ sets, $\{S_i\}$, of exponentially growing cardinality ranges, exactly as described in Subsection 6.1. From a simple counting argument, at least one of these sets accounts for at least a $\Omega(1/\log n)$-fraction of the total weight covered by the optimal solution. Let S_1 denote this set. Next, consider the set S_2 of queries in S_1 such that no queries in S_1 are covered above them (i.e. by a category closer to the root). By definition, for every query in S_1 there is a subtree rooted in a category that covers a query in S_2. And since all queries in S_1 are of the same cardinality up to a constant factor, there are at most $O(M)$ queries covered in each such subtree. Finally, consider the set S_3 consisting of the query of the highest weight in each of the subtrees described above. This set of queries accounts for at least a $\Omega(1/M)$-fraction of the total weight of S_1, and thus a $\Omega(1/M \log n)$-fraction of the total weight covered by the optimal solution. It follows that the set of categories that cover S_3 is an *OCP* solution (since these categories belong to disjoint subtrees) covering at least a $\Omega(1/M \log n)$-fraction of the optimal *OCT* solution. □

Queries of constant cardinality. We next show that when all query cardinalities are upper bounded by a constant, then, for *any OCP* or *OCT* (with $r = 1$) problem variant with constant, the A_P^r algorithm covers at least an $\tilde{\Omega}(1/M)$ fraction of the total weight of Q, with perfect recall and precision (i.e., via an identical category). Note that, in particular, the algorithm guarantees an $\tilde{O}(M)$-approximation. We will later also prove that, by using a different algorithm, the same guarantee can be provided for all variants that do not require perfect recall, even without assuming any bounds on the query cardinalities.

Concretely, when the query sizes are bounded by a constant, we can simply use the A_P^r algorithm (depicted in Algorithm 2) with the MIS algorithm in Theorem 3.5 as the underlying *MIS* algorithm. We next show that the number of conflicts is guaranteed to be small (when M is small) which implies the approximation guarantee.

Theorem 6.3. *The A_P^r algorithm produces a solution that covers an $\Omega(\frac{1}{M \log n})$-fraction of the total weight of the input queries for both $OCT^1(\mathcal{T}_{\alpha,\beta})$ and $OCP^r(\mathcal{T}_{\alpha,\beta})$ instances, where the query size is bounded by a constant. For unweighted variants, this guarantee improves to a $\Omega(\frac{1}{M})$-fraction. Moreover, this guaranteed fraction of queries are covered with both precision and recall scores of 1. Lastly, this guarantee also applies to variants using the Jaccard or F_1 similarity functions.*

Proof. We prove the result below for unweighted inputs, as we showed in Subsection 6.1 how one transforms weighted instances into unweighted instances at the cost of a $\log n$ factor. Let $d = O(1)$ denote the maximum cardinality of an input query. Since each element appears in at most M queries, the number of queries each given query intersects with cannot exceed dM. Therefore, the average degree in the conflict graph is at most $O(M)$, and by using the *MIS* solver in Theorem 3.5, we get the desired result. Note that since A_P^r covers the queries produced by the *MIS* algorithm with identical categories, these covers have perfect precision and recall and apply to all variants. □

Non-perfect recall. We next show that, for variants where $r = 1$ and perfect recall is not required, a simple randomized algorithm, A_{rand}, depicted in Algorithm 3, can with high probability produce a solution that covers at least an $\Omega(1/M)$-fraction of the total weight of the input queries. Note the stark contrast to variants with perfect recall, discussed above, where even for $M = 2$ one cannot guarantee anything significantly above a trivial result (whereas, in the cases below for constant M we get a constant approximation). We assume below that all query cardinalities exceed a sufficiently large constant, d. This loses at most a constant factor of 2 in the approximation guarantee, as one can partition the input into two instances - one consisting of all queries of cardinality at most d, and one of the remaining queries. For the former, we can use the A_P^r algorithm described above, and for the

latter - the A_{rand} algorithm we describe below with the same coverage guarantee. At least one of the two inputs must contain at least half of the total weight of the queries in the original input, proving that we lose at most a factor of 2 (however, note that for weighted cases with queries of bounded size the guaranteed fraction is smaller by an additional $\log n$ factor, as described in Theorem 6.3).

We next describe the randomized algorithm, and then state and prove its performance guarantee. In the first stage, it selects a sample $S_1 \subseteq Q$ where every query is selected independently with probability $\frac{1}{cM}$) where c is a sufficiently large constant (whose value will be determined later in the probabilistic analysis). It then computes for each sampled query q, its subset q' of elements that only appear in one sampled query (i.e., only in q). Then, for each sampled query q, if q' contains at least a β-fraction of its elements, it is added to a set denoted by S_2. Lastly, the set categories in the final output consists, for every $q \in S_2$, of the corresponding subset q'. Clearly, since every element in each category has a multiplicity of 1 in S_1, the output is a valid category partition. As before for *OCT*, we use the standard adaption of the *OCP* solution of adding a root as the direct ancestor of all the categories in the partition. Also note that the set of covered queries is S_2, where each query is covered with perfect precision and the recall is at least β.

Algorithm 3: A_{rand} Algorithm

Input: Q - queries, W - query weights, β - recall threshold
Output: S - category partition/tree

1 Select a sample $S_1 \subseteq Q$ where every query is selected independently with probability $\Theta(1/M)$.;
2 For each sampled query q compute its subset q' of elements that only appear in one sampled query ;
3 Compute the set $S_2 \subseteq S_1$ consisting of every query q such that $|q'| \geq \beta|q|$;
4 Compute $S = \cup_{q \in S_2} q'$;
5 **return** *S (for OCT also add a root that connects to all categories in S)*;

Theorem 6.4. *The A_{rand} algorithm covers an $\Omega(1/M)$-fraction of the total weight of Q for $OCT^1(\mathcal{T}_{\alpha,\beta})$ and $OCP^1(\mathcal{T}_{\alpha,\beta})$ variants where $\beta < 1$, with an arbitrarily small constant probabilistic error. The same guarantee also applies to the $OCT^1(\hat{J}_\delta)$ and $OCP^1(\hat{J}_\delta)$ variants, where $\delta < 1$.*

Proof. To simplify the presentation, we will prove the statement for unweighted instances, and then describe the simple generalization of several arguments, so that the proof applies analogously to weighted instances as well. Also note that since, as discussed above, we only handle the case where all queries are sufficiently large, we can ignore any rounding issues when discussing constant fractions of cardinalities of queries.

We will show that for any constant δ (we also assume that $\delta < 1/2$ as this only makes the argument stronger), one can adjust the parameter c (recall that in the first phase of the algorithm every query is sampled with probability $\frac{1}{cM}$)), such that, with probability at least $(1-\delta)$ the total cardinality of the set S_2, which is covered by the final solution, is at least $\Omega(n/M)$.

In the probabilistic analysis below, we condition on the event that the total number of sampled queries is at least a $\frac{1}{cM}$) $-c'$, where c' is an arbitrarily small constant (i.e. the size of the sample is at least negligibly less than the expectation). A straightforward application of the Chernoff bound (Lemma 3.10 implies that this happens with probability $1 - o(1)$. Since this probabilistic error is negligibly small, for simplicity of presentation, we ignore it in the subsequent analysis. It, therefore, remains to prove that the size of S_2 is with high probability a constant fraction of S_1.

We call every element whose multiplicity in the sample is 1 (i.e. it appears in only one query) as a *good* element, whereas all other elements are *bad* elements. We similarly call every sampled query in S_2 a *good* query, whereas all other sampled queries are *bad*. We next analyze the number of good elements and queries in S_1.

For any given query $q \in S_1$, each of its elements is good only if all other queries that contain it are not sampled. Since there are at most $M-1$ such queries, the probability that an element is good is $1-(1-1/(cM))^{M-1}$. Applying Bernoulli's inequality [21] we get that

$$1-(1-1/(cM))^{M-1} \geq 1-\frac{M-1}{cM} \geq 1-1/c.$$

Hence, the probability that the element is bad is at least $1/c$. We can choose c sufficiently large (yet still a constant) such that the probability of an element being bad is $\frac{1-\beta}{\delta/2}$. From the linearity of expectation, it would then follow that the expected fraction of bad elements is $\frac{1-\beta}{\delta^2}$. Then a simple counting argument (concretely, Markov's inequality), implies that the probability that the fraction of bad elements exceeds $(1-\beta)$, i.e., that the query is bad, is at most δ^2. It follows that the expected fraction of bad queries is at most δ^2. From the same Markov's inequality argument, the probability that the fraction of bad queries exceeds $2\delta^2/\delta = 2\delta$ is at most δ. Therefore, with probability at least $\delta/2$, the number of good queries is at least a $(1-2\delta)$-fraction of the queries in S_1 (recall that we assumed that $\delta < 1/2$). And since we conditioned on the high probability event that the cardinality of S_1 is $\Omega(n/M)$, this concludes the proof.

To generalize the proof for the weighted instances, we only need to nominally generalize the three arguments regarding the number of bad queries. Concretely, the same bound for the expected fraction of the number of bad queries of the total sampled queries applies to the fraction of the total expected weight of the covered queries out of the total weight of the sampled queries. This is also the case for the expected fraction of the weight of the sampled queries which generalizes the fraction of the sampled queries out of n. Moreover, the same exact counting argument, based on the Markov bound, that implies that the number of bad queries with high probability does not exceed its expectation, applies for the total weight of the bad queries not exceeding its expectation by much. The rest of the proof is exactly the same (in the above arguments we ignored degenerate cases where a constant fraction of the total weight is concentrated in a few queries, as one can in *PTIME* test all small solutions and find this set).

Lastly, the result also applies to the analogous Jaccard variants, since if the Jaccard similarity is at least δ then the recall is at least δ as well (in fact, when the precision is 1, then the recall equals the Jaccard similarity. □

7 RELATED WORK

The construction of category trees/taxonomies has been studied in multiple domains, including e-commerce, document management, and question answering [11, 14, 29]. Many algorithms have been devised for automating taxonomy construction [11, 22, 23] and maintenance, [27, 29, 32] employing different clustering approaches [11, 22], as well as crowdsourcing [26].

In the lines of work specified above, the quality of the resulting taxonomy is assessed along the following two dimensions.

The first dimension of quality assessment is user-study [11, 22], an evaluation which we incorporate w.r.t. our model in the complementary empirical work [5]. This evaluation is naturally entirely subjective.

The second dimension, which is the focus of the present paper, is the similarity of the resulting category tree to a given (combinatorially unrestricted) ground-truth set of items/documents. For example, the F_1 score used in [11, 22, 23] is a variant (without a threshold) of our corresponding F_1 measure for $r = 1$. Similarly, [27] computes recall and F_1 scores for the resulting trees, also with $r = 1$.

To our knowledge, however, no previous work investigates the theoretical complexity of the optimization problem of computing the tree of the highest score. The score is only used as an evaluation measure, to which the algorithm is oblivious. This approach, to an extent, is loosely justified by our worst-case bounds. Nevertheless, we show in [5] and [4], that leveraging the relation we outlined in Section 6 to the weighted *MIS* problem, allows solving well (and, in some cases, optimally) real-world problem instances, via extensively studied *MIS* solvers.

Our model differs from clustering models [15, 24] that typically focus on item-similarity, optimizing the similarity within each cluster or the dissimilarity across clusters. Moreover, these models are commonly defined by pairwise similarities, while our model also considers relations of a higher order. Thus, closest to our work in this domain is the field of hypergraph partitioning (clustering) [16, 19]. Specifically, the *OCP* problem with copy-bound $r = 1$ corresponds to seeking a partition of the vertices that maximizes the weight of (hyper)edges for which there is a similar set in the partition. Relaxing the copy-bound corresponds to overlapping clusters. Importantly, this relation between hypergraph clustering and our model is different from the more artificial relation leveraged in our reductions, where we cluster the hypergraph edges, instead of the vertices. Nevertheless, our proposed framework differs from existing models in several aspects. Notably, hypergraph clustering typically studies a multi-way cut problem, intending to minimize the weight of the cut edges. Recently [19] suggested that there is a benefit in quantifying how an edge is cut, in terms of which subsets of its vertices are clustered together. Our work is relevant in that respect, as we quantify how similar these subsets are to the original edge.

A work resembling ours in a different aspect is [31], where the objective is to maximize the edge weights inside the cluster (we also maximize the covered "demand", instead of the less natural minimization of uncovered demand). However, the models of [19] and [31] (and many others [10, 18]) are easier to approximate, due to principal technical differences (e.g., bounds on the size and number of clusters), and we are not aware of clustering research that resembles our model or bounds.

8 CONCLUSION

In this paper, we studied the hardness of computing categorizations with a bounded number of possible repetitions, that best capture a given collection of item sets. We defined a model that captures various practical settings and proved inapproximability results for multiple variants and special cases. We also provided an algorithm for the Exact variant with an approximation guarantee that depends on finer input parameters, along with algorithms, with much improved guarantees compared to the worst case, for various special cases where the cardinality of the input sets or the number of sets each item belongs to are upper bounded.

An interesting direction for future work would be to identify more special cases that admit improved performance. Another intriguing avenue of exploration is determining for cases where we showed $\tilde{\Theta}(\sqrt{n})$ hardness, whether one can devise algorithms with matching approximation guarantees or prove stronger bounds.

REFERENCES

[1] [n. d.]. https://export.ebay.com/en/start-sell/selling-basics/seller-fees/fees-optional-listing-upgrades/.

[2] Geir Agnarsson, Magnús M Halldórsson, and Elena Losievskaja. 2013. SDP-based algorithms for maximum independent set problems on hypergraphs. *Theoretical Computer Science* 470 (2013), 1–9.

[3] Rakesh Agrawal, Amit Somani, and Yirong Xu. 2001. Storage and querying of e-commerce data. In *VLDB*. 149–158.

[4] Uri Avron, Shay Gershtein, Ido Guy, Tova Milo, and Slava Novgorodov. 2021. ConCaT: Construction of Category Trees from Search Queries in E-Commerce. In *ICDE*.

[5] Uri Avron, Shay Gershtein, Ido Guy, Tova Milo, and Slava Novgorodov. 2022. Automated Category Tree Construction in E-Commerce. In *SIGMOD*. 1770–1783.

[6] Nikhil Bansal, Anupam Gupta, and Guru Guruganesh. 2015. On the Lovász Theta function for Independent Sets in Sparse Graphs. *CoRR* abs/1504.04767 (2015).

[7] Slobodan Beliga, Ana Meštrović, and Sanda Martinčić-Ipšić. 2015. An overview of graph-based keyword extraction methods and approaches. *JIOS* 39, 1 (2015), 1–20.

[8] Amey Bhangale and Subhash Khot. 2019. UG-hardness to NP-hardness by Losing Half. In *CCC*.

[9] Yair Caro and Zsolt Tuza. 1991. Improved lower bounds on k-independence. *Journal of Graph Theory* 15, 1 (1991), 99–107.

[10] Karthekeyan Chandrasekaran, Chao Xu, and Xilin Yu. 2019. Hypergraph k-cut in randomized polynomial time. *Mathematical Programming* (2019), 1–29.

[11] Shui-Lung Chuang and Lee-Feng Chien. 2004. A Practical Web-Based Approach to Generating Topic Hierarchy for Text Segments. In *CIKM*. 127–136.
[12] Idan Hasson, Slava Novgorodov, Gilad Fuchs, and Yoni Acriche. 2021. Category Recognition in E-Commerce using Sequence-to-Sequence Hierarchical Classification. In *WSDM*.
[13] Thomas Hofmeister and Hanno Lefmann. 1998. Approximating maximum independent sets in uniform hypergraphs. In *Proc. of MFCS*. 562–570.
[14] Yi-Hsiang Hsieh, Shih-Hung Wu, Liang-Pu Chen, and Ping-Che Yang. 2017. Constructing Hierarchical Product Categories for E-Commerce by Word Embedding and Clustering. In *IRI*. 397–402.
[15] Anna Huang. 2008. Similarity measures for text document clustering. In *NZCSRSC*, Vol. 4. 9–56.
[16] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1999. Multilevel hypergraph partitioning: applications in VLSI domain. *VLSI* 7, 1 (1999), 69–79.
[17] Sebastian Lamm, Christian Schulz, Darren Strash, Robert Williger, and Huashuo Zhang. 2019. Exactly solving the maximum weight independent set problem on large real-world graphs. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 144–158.
[18] Tom Leighton, Fillia Makedon, and SG Tragoudas. 1990. Approximation algorithms for VLSI partition problems. In *ISCAS*. 2865–2868.
[19] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous hypergraph clustering with applications. In *NIPS*. 2308–2318.
[20] Xueqing Liu, Yangqiu Song, Shixia Liu, and Haixun Wang. 2012. Automatic taxonomy construction from keywords. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1433–1441.
[21] Dragoslav S Mitrinovic and Petar M Vasic. 1970. *Analytic inequalities*. Vol. 1. Springer.
[22] Kunal Punera, Suju Rajan, and Joydeep Ghosh. 2005. Automatically Learning Document Taxonomies for Hierarchical Classification. In *Proc. of WWW*.
[23] Cécile Robin, James O'Neill, and Paul Buitelaar. 2017. Automatic Taxonomy Generation - A Use-Case in the Legal Domain. In *LCT*.
[24] Lior Rokach and Oded Maimon. 2005. Clustering methods. In *Data mining and knowledge discovery handbook*. Springer, 321–352.
[25] Matthew Skala. 2013. Hypergeometric tail inequalities: ending the insanity. arXiv:1311.5939 [math.PR]
[26] Yuyin Sun, Adish Singla, Dieter Fox, and Andreas Krause. 2015. Building Hierarchies of Concepts via Crowdsourcing. In *IJCAI 2015*.
[27] Lei Tang, Jianping Zhang, and Huan Liu. 2006. Acclimatizing taxonomic semantics for hierarchical content classification. In *Proc. of KDD*. 384–393.
[28] Eli Upfal. 2005. *Probability and computing: randomized algorithms and probabilistic analysis*. Cambridge university press.
[29] Quan Yuan, Gao Cong, Aixin Sun, Chin-Yew Lin, and Nadia Magnenat Thalmann. 2012. Category hierarchy maintenance: a data-driven approach. In *SIGIR*. 791–800.
[30] Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. 2014. Taxonomy discovery for personalized recommendation. In *WSDM*. 243–252.
[31] Wenxing Zhu and Chuanyin Guo. 2010. Local Search Approximation Algorithms for the Complement of the Min-k-Cut Problems. *Research Report, Fuzhou University* (2010).
[32] Hai Zhuge and Lei He. 2017. Automatic maintenance of category hierarchy. *Future Generation Computer Systems* 67 (2017), 1 – 12.
[33] David Zuckerman. 2006. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. of STOC*. 681–690.

Received 27 February 2023; revised 2 October 2023; accepted 2 April 2024