Parameterized Query Complexity of Hitting Set Using Stability of Sunflowers

Arijit Bishnu
Indian Statistical Institute, Kolkata, India

Arijit Ghosh
The Institute of Mathematical Sciences, Chennai, India

Sudeshna Kolay
Eindhoven University of Technology, Eindhoven, Netherlands

Gopinath Mishra
Indian Statistical Institute, Kolkata, India

Saket Saurabh
The Institute of Mathematical Sciences, Chennai, India

Abstract

In this paper, we study the query complexity of parameterized decision and optimization versions of Hitting-Set. We also investigate the query complexity of Packing. In doing so, we use generalizations to hypergraphs of an earlier query model, known as BIS introduced by Beame et al. in ITCS'18. The query models considered are the GPIS and GPISE oracles. The GPIS and GPISE oracles are used for the decision and optimization versions of the problems, respectively. We use color coding and queries to the oracles to generate subsamples from the hypergraph, that retain some structural properties of the original hypergraph. We use the stability of the sunflowers in a non-trivial way to do so.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability, Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Query complexity, Hitting set, Parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.25

Related Version A full version of the paper is available at [3], https://arxiv.org/abs/1807.06272.

1 Introduction

In query complexity models for graph problems, the aim is to design algorithms that have access to the vertices $V(G)$ of a graph G, but not the edge set $E(G)$. Instead, these algorithms construct local copies by using oracles to probe or infer about a property of a part of the graph. Due to the lack of knowledge about global structures, often it is difficult to design algorithms even for problems that are classically known to have polynomial time algorithms.

A natural optimization question in this model is to minimize the number of queries to the oracle to solve the problem. The most generic approach towards this is to ask as few queries to the oracle before the local copy of the graph is an equivalent sample of the actual graph. This spawns the study of query complexity. The query complexity of the algorithm is the number of queries made to the oracle. Keeping this in mind, several query models have been designed through the years. Let us take the example of the problem of finding a global minimum cut that has led to the introduction of different query models, in order to

achieve a query complexity that is less than the complexity of the actual graph. The query models started from the simple *neighbor query*, but soon people realized that this was not ideal for minimizing the query complexity for most problems [7, 9]. Therefore, in the case of the minimum cut problem, the *cut query* was introduced to achieve subquadratic query complexity [13].

There is a vast literature available on the query complexity of problems with classical polynomial time algorithms (Refer to book [8]). However, there has been almost negligible work on algorithmically hard problems [10, 11, 12]. In this paper, we use ideas of parameterized complexity in order to study the query complexity of NP-hard problems. The Hitting Set and Vertex Cover problems are test problems for all new techniques of parameterized complexity and also in every subarea that parameterized complexity has explored. We continue the tradition and study the query complexity of these problems. We start by defining a generalization of a recently introduced query model [2] to handle hypergraphs.

1.1 The model

A hypergraph is a set system $(U(\mathcal{H}), \mathcal{F}(\mathcal{H}))$, where $U(\mathcal{H})$ is the set of vertices and $\mathcal{F}(\mathcal{H})$ is the set of hyperedges. A hypergraph $\mathcal{H}'$ is a sub-hypergraph of $\mathcal{H}$ if $U(\mathcal{H}') \subseteq U(\mathcal{H})$ and $\mathcal{F}(\mathcal{H}') \subseteq \mathcal{F}(\mathcal{H})$. For a hyperedge $F \in \mathcal{F}(\mathcal{H})$, $U(F)$ or simply F denotes the subset of elements that form the hyperedge. A d-uniform hypergraph has each hyperedge of size d. A *packing* in a hypergraph $\mathcal{H}$ is a family $\mathcal{F}'$ of hyperedges such that for any two hyperedges $F_1, F_2 \in \mathcal{F}'$, $U(F_1) \cap U(F_2) = \emptyset$.

For us "choose a random hash function $h : V \to [N]$", means that each vertex in V is colored with one of the N colors uniformly and independently at random.

In this paper, for a problem instance (I, k) of a parameterized problem Π, a high probability event means that it occurs with probability at least $1 - \frac{1}{k^c}$, where k is the given parameter and c is a constant. The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. For a function $f(k)$, the set of functions $\mathcal{O}(f(k) \cdot \log k)$, is denoted by $\widetilde{\mathcal{O}}(f(k))$.

Motivated by [2] and [11], we consider the following oracles to look at the parameterized query complexity of NP hard graph problems.

Generalized d-partite independent set oracle (GPIS): For a d-uniform hypergraph $\mathcal{H}$, given d pairwise disjoint non-empty subsets $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ as input, a GPIS query oracle answers whether there exists an edge $(u_1, \ldots, u_d) \in \mathcal{F}(\mathcal{H})$ such that $u_i \in A_i$, for each $i \in [d]$.

Generalized d-partite independent set edge oracle (GPISE): For a d-uniform hypergraph $\mathcal{H}$, given d pairwise disjoint non-empty subsets $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ as input, a GPISE query oracle outputs a hyperedge $(u_1, \ldots, u_d) \in \mathcal{F}(\mathcal{H})$ such that $u_i \in A_i$, for each $i \in [d]$; otherwise, the GPISE oracle reports NULL.

For $d = 2$, GPIS oracle is same as Bipartite Independent Set (BIS) oracle introduced by Beame et al. [2]. Similarly, we can define BISE oracle as GPISE oracle for $d = 2$. Notice that BIS is an existence query and is a natural extension of edge existence query. To get a clear motivation behind BIS query, please refer to [2]. BISE (GPISE) is powerful over BIS (GPIS) as BISE (GPISE) can return an edge (a hyperedge) between sets.

As mentioned earlier, queries like *degree query*, *edge existence query*, *neighbor query*, that obtain local information about the graph have its limitation in terms of not being able to achieve *efficient* query costs [7, 9]. This necessitates looking at powerful queries that

goes beyond obtaining local information and generalizes earlier queries. Beame et al. [2] introduced BIS query model and approximately estimated the number of edges in a graph.

In the context of NP-Hard problems, it is not known if any problem can have *efficient* query complexity with conventional query models. So, it is reasonable to study query complexity for parameterized versions of NP-Hard problems. Iwama and Yoshida [11] initiated the study of parameterized version of some NP-Hard problems in the graph property testing framework with the access to standard oracles. We will give the details of their work in Section 1.3 and compare with ours. Now, a natural question to ask is "can we improve the query complexity (of NP-Hard problems) with no assumption on the input by considering a relatively stronger oracle" ? As a first step in this direction, we use GPIS (GPISE) oracles, which are nothing but BIS oracle for hypergraphs, to study parameterized decision (optimization) version of Hitting Set. We believe that these query models will be useful to study the (parameterized) query complexity of other NP-Hard problems.

1.2 Problem definition and our results

The d-Hitting-Set problem is defined as follows. Note that d is a constant in this paper and $HS(\mathcal{H})$ denote a minimum hitting set of a hypergraph $\mathcal{H}$.

d-Hitting-Set
Input: The set of vertices $U(\mathcal{H})$ of a d-uniform hypergraph $\mathcal{H}$, the access to a GPISE oracle, and a positive integer k.
Output: A set $HS(\mathcal{H})$ having at most k vertices such that any hyperedge in $\mathcal{H}$ intersects with $HS(\mathcal{H})$ if such a set exists. Otherwise, we report such a set does not exist.

The d-Decision-Hitting-Set problem is the usual decision version of d-Hitting-Set; here the oracle access is to GPIS instead of GPISE.

In our solution framework, we make queries to oracles to build a reduced instance of the problem. On this reduced instance, one can run the traditional (FPT) algorithms. While stating the results, we will bother only about the number of queries required to build the reduced instance. In the query complexity setting, the algorithms are required to make bounded number of queries (good bounds on the total time complexity is not an issue). Our main focus in this paper is to make the query complexity results parameterized, in the sense that they have query complexities bounded by some input parameters of the problem. So, our bounds on the query complexity are not directly comparable with the time complexities of the FPT algorithms in the literature of parameterized complexity. Our results hold with high probability. Our methods use the technique of *color coding* [1, 5] to restrict the number of queries required to generate a reduced instance of interest. The main result of our paper is the following.

▶ **Theorem 1.1.** *d-Hitting-Set can be solved with $\widetilde{\mathcal{O}}(k^{2d})$ GPISE queries and d-Decision-Hitting-Set can be solved with $\widetilde{\mathcal{O}}(k^{2d^2})$ GPIS queries.*

Our solution to d-Hitting-Set needs us to solve another problem of interest termed as d-Packing in a hypergraph, which is a generalization of Matching in a graph. We describe the sketch of our query procedure to solve d-Packing in Section 2. Section 3 has the detailed study on Hitting Set. Table 1 gives the overview of our results.

Table 1 Query complexities for hypergraph problems using GPIS and GPISE oracles. Observe that Vertex Cover results follow by putting $d = 2$ in the above table.

Problems	Query Oracles	
	GPIS	GPISE
d-Hitting-Set	—	$\widetilde{\mathcal{O}}(k^{2d})$
d-Decision-Hitting-Set	$\widetilde{\mathcal{O}}(k^{2d^2})$	$\widetilde{\mathcal{O}}(k^{2d})$
d-Promised-Hitting-Set	—	$\widetilde{\mathcal{O}}(k^{d})$
d-Packing	—	$\widetilde{\mathcal{O}}(k^{2d})$

1.3 Related Works

Several query complexity models have been proposed in the literature to study various problems [7, 9]. The only work prior to ours related to parameterization in the query complexity model was by Iwama and Yoshida [11]. They studied property testing for several parameterized NP optimization problems in the query complexity model. For the query, they could ask for the degree of a vertex, neighbors of a vertex and had an added power of sampling an edge uniformly at random, which is quite unlike in usual query complexity models. To justify the added power of the oracle to sample edges uniformly at random, they have shown that $\Omega(\sqrt{n})$ degree and neighbor queries are required to solve Vertex-Cover. Apart from that, an important assumption in their work is that the algorithms knew the number of edges, which is not what is usually done in query complexity models. Also, the algorithms that are designed gives correct answer only for stable instances. In contrast, our query oracles do not use any randomness, does not know the number of edges, consider all instances, and have a unifying structure. Hence, in this paper, oracles have less power than that of [11] in the context of amount of randomness used by the oracles. Of significance to us, is the vertex cover problem. Their vertex cover algorithm admits a query complexity of $\widetilde{\mathcal{O}}(\frac{2^k}{\epsilon^2})$ and either finds a vertex cover of size at most k or decides that there is no vertex cover of size bounded by k even if we delete ϵm edges, where the number of edges m is known in advance. In contrast, our algorithm uses BISE query for the vertex cover problem; it does not need to estimate the number of edges. Our algorithm admits a query complexity of $\widetilde{\mathcal{O}}(k^4)$ and we either find a vertex cover of size at most k if it exists or decide that there is no vertex cover of size bounded by k. If it is promised that the vertex cover is bounded by k, then we can give an algorithm that makes $\widetilde{\mathcal{O}}(k^2)$ BISE queries. These results on the Vertex Cover are not the main focus of this paper and are mentioned in the full version [3]. The main foucus of this paper is our results on d-Hitting-Set; extension of Vertex Cover to d-Hitting-Set requires a deeper understanding of stability of *sunflowers* under random sampling. Hence, it is evident that GPIS (BIS) and GPISE (BISE) open up a new dimension in the study of *query complexity*. It will be interesting to study what other NP-hard problems can be solved *efficiently* with these oracles.

Recent papers have considered strengthened query complexity models. In [2], the BIS oracle was introduced to design better edge estimation algorithms. In the same work, the IS oracle was also introduced, to estimate the number of edges, where the input to the oracle is a vertex subset $A \subseteq V(G)$ and the output is 1, if the subgraph of G induced by A is an independent set and 0, otherwise. Similarly, in [13], the *cut query* was introduced to obtain better query complexity for minimum cut problem.

2 d-Packing

We first define d-Packing and then design the query procedure.

d-Packing
Input: The set of vertices $U(\mathcal{H})$ of a d-uniform hypergraph $\mathcal{H}$, the access to a GPISE oracle, and a positive integer k.
Output: A pairwise disjoint set of at least k hyperedges if such a set of hyperedges exists. Otherwise, we report such a set of hyperedges does not exist.

As the usual *matching* in a graph (denoted here as Matching) is a special case of d-Packing, we explain the main ideas of the query procedure of d-Packing with Matching. In Matching, our objective is to either report a matching of at least k edges or decide there does not exist a matching of size at least k. We use a hash function to color all the vertices of G. In fixing the number of colors needed, we need to ensure that the endpoints of the matched edges belong to different color classes. If the hash function uses $\mathcal{O}(k^2)$ colors, then with constant probability the endpoints of a k-sized edge set, that certifies the existence of a matching of size at least k, will be in different color classes. For each pair of color classes, we query the BISE oracle and construct a subgraph $\hat{G}$ according to the outputs of BISE queries. We will show that if G has a matching of k edges, then $\hat{G}$ has a matching of k edges. As $\hat{G}$ is a subgraph of G, any matching of $\hat{G}$ is also a matching of G and the size of maximum matching in $\hat{G}$ is less than that of G. So, we report the required answer from the matching of $\hat{G}$. By repeating the query procedure for $\mathcal{O}(\log k)$ times and taking maximum of all the outcomes, we can report the correct answer with high probability. We carry over the above ideas to the hypergraph setting with the oracle being GPISE. Let $\mathsf{Pack}(\mathcal{H})$ denote a maximum packing of $\mathcal{H}$.

▶ **Theorem 2.1.** *d-Packing can be solved with $\widetilde{\mathcal{O}}(k^{2d})$ GPISE queries.*

Proof Sketch. Observe that it is enough to give an algorithm that solves d-Packing with probability at least $2/3$ by using $\mathcal{O}(k^{2d})$ GPISE queries. The details are in the full version [3].

We choose a random hash function $h : U(\mathcal{H}) \to [\gamma k^2]$, where $\gamma = 100d^2$. Let $U_i = \{u \in U(\mathcal{H}) : h(u) = i\}$, where $i \in [\gamma k^2]$. Note that $\{U_1, \ldots, U_{\gamma k^2}\}$ form a partition of $U(\mathcal{H})$, where some of the U_i's can be empty. We make a GPISE query with input $(U_{i_1}, \ldots, U_{i_d})$ for each $1 \leq i_1 < \ldots < i_d \leq \gamma k^2$ such that $U_{i_j} \neq \emptyset \; \forall j \in [d]$. Observe that we make $\mathcal{O}(k^{2d})$ queries to the GPISE oracle. Let $\mathcal{F}'$ be the set of hyperedges that are output by the $\mathcal{O}(k^{2d})$ GPISE queries. Now, we can generate a sub-hypergraph $\hat{\mathcal{H}}$ of $\mathcal{H}$ such that $U(\hat{\mathcal{H}}) = U(\mathcal{H})$ and $\mathcal{F}(\hat{\mathcal{H}}) = \mathcal{F}'$. We find $\mathsf{Pack}(\hat{\mathcal{H}})$. If $\left|\mathsf{Pack}(\hat{\mathcal{H}})\right| \geq k$, then we report $\mathsf{Pack}(\hat{\mathcal{H}})$ as $\mathsf{Pack}(\mathcal{H})$. Otherwise, we report there does not exist a packing of size k. The correctness of our query procedure follows from Lemma 2.2 (proof is in the full version [3]) along with the fact that any packing of $\hat{\mathcal{H}}$ is also a packing of $\mathcal{H}$, as $\hat{\mathcal{H}}$ is a sub-hypergraph of $\mathcal{H}$.

▶ **Lemma 2.2.** *If $|\mathsf{Pack}(\mathcal{H})| \geq k$, then $\left|\mathsf{Pack}(\hat{\mathcal{H}})\right| \geq k$ with probability at least $2/3$.* ◀

3 Algorithm for Hitting Set (Theorem 1.1)

3.1 Our ideas in a nutshell

The main ideas explained with Vertex Cover

The d-Hitting-Set problem with $d = 2$ is Vertex-Cover. We first explain the intuition behind our algorithm for d-Hitting-Set with Vertex Cover. The first step is to solve the problem on instances where there is a promise of a Vertex-Cover solution of size

at most k. For this promised version, we use a hash function to color all the vertices of the graph G. We sample a subgraph of G by querying the BISE oracle for each pair of color classes. We sample several such subgraphs of G using the BISE oracle, and finally take the union of these subgraphs to form a single subgraph $\hat{G}$ of G. Finally, we analyse that a minimum vertex cover of $\hat{G}$ is also a minimum vertex cover of G and vice versa. Our analysis is inspired by the analysis of the streaming algorithm for Vertex-Cover [4], and presented in the full version [3]. The non-promised version of Vertex-Cover can be solved by using the algorithm for the promised version along with the algorithm explained for Matching in Section 2. If there exists a matching of size more than k, then the vertex cover is also more than k. Otherwise, the vertex cover is bounded by $2k$. Now we can use our algorithm for the promised version of Vertex Cover to find an exact vertex cover from which we can give final answer to the non-promised Vertex Cover. When we consider the decision version of Vertex-Cover, we only need access to the BIS oracle. We use the fact that the Vertex-Cover problem has an efficient *representative set* of edges [5] associated with it (please refer to the full version [3]) to solve Decision-Vertex-Cover. This helps us to design an algorithm with access to the BIS oracle. This technique also works for d-Decision-Hitting-Set.

Moving from Vertex Cover to d-Hitting-Set

The algorithm for d-Hitting-Set, having a query complexity of $\widetilde{\mathcal{O}}(k^{2d})$ GPISE queries, will use an algorithm admitting query complexity $\widetilde{\mathcal{O}}(k^d)$ for a promised version of this problem. In the promised version, we are guaranteed that the input instance has a hitting set of size at most k. The main idea to solve the promised version is to sample a *suitable* sub-hypergraph having bounded number of hyperedges, using GPISE queries, such that the hitting set of the sampled hypergraph is a hitting set of the original hypergraph and vice versa. We use the stability of *sunflowers* under random sampling. Recall that a hypergraph can be thought of as a *set system*. The core of a sunflower is the pairwise intersection of the hyperedges present in the sunflower, which is formally defined as follows.

▶ **Definition 3.1.** Let $\mathcal{H}$ be a d-uniform hypergraph; $\mathcal{S} = \{F_1, \ldots, F_t\} \subseteq \mathcal{F}(\mathcal{H})$ is a *t-sunflower* in $\mathcal{H}$ if there exists $C \subseteq U(\mathcal{H})$ such that $F_i \cap F_j = C$ for all $1 \leq i < j \leq t$. C is defined to be the *core* of the sunflower $\mathcal{S}$ in $\mathcal{H}$ and $\mathcal{P} = \{F_i \setminus C : i \in [t]\}$ is defined as the set of *petals* of the sunflower $\mathcal{S}$ in $\mathcal{H}$.

The core of a sunflower can be *large*, or *significant*, or *small*; based on the number of hyperedges forming the sunflower. We define large, significant and small in such a way that each large core is significant and each significant (and thus, large) core must intersect with any hitting set. The formal definition of different types of cores is given below.

▶ **Definition 3.2.** Let $S_{\mathcal{H}}(C)$ denote the maximum integer t such that C is the core of a t-sunflower in $\mathcal{H}$. If $S_{\mathcal{H}}(C) > 10dk$, the core C is said to be *large*. If $S_{\mathcal{H}}(C) > k$, core C is said to be *significant*.

The promise that the hitting set is bounded by k, will help us (i) to bound the number of hyperedges that do not contain any large core as a subset, (ii) to guarantee that all the large cores, that do not contain any significant cores as subsets in the original hypergraph, are significant in the sampled hypergraph with high probability, and hence will intersect any hitting set of the sampled hypergraph, (iii) to guarantee that all the hyperedges that do not contain any large core as a subset, are present in the sampled hypergraph with high probability. Using the above properties, we can prove that reporting the hitting set of the sampled hypergraph as the hitting set of the original graph is correct with high probability. The formal definitions and arguments are given in Section 3.3.

In this Section we also give algorithm for d-Decision-Hitting-Set, where we have access to the GPIS oracle and obtain an algorithm with query complexity $\widetilde{\mathcal{O}}(k^{2d^2})$. The main idea to solve d-Decision-Hitting-Set is to use the concept of representative sets [5] (For details see the full version [3]). The size of a k-representative set corresponding to a hypergraph is bounded by $\mathcal{O}(k^d)$. Thus, the number of vertices that are present in the k-representative set is also bounded by $\mathcal{O}(dk^d)$. All the $\mathcal{O}(dk^d)$ vertices will be uniquely colored with high probability if enough number of colors are used for the hash function. Then we make GPIS queries to extract a sufficient number of hyperedges such that the hyperedges corresponding to the representative set are *embedded* in the sampled sub-hypergraph. The formal arguments are given in Section 3.3.

3.2 d-Promised-Hitting-Set

In this part, we study the following problem.

d-Promised-Hitting-Set
Input: The set of vertices $U(\mathcal{H})$ of a d-uniform hypergraph $\mathcal{H}$ such that $|HS(\mathcal{H})| \leq k$ and the access to a GPISE oracle, and a positive integer k.
Output: A hitting set of $\mathcal{H}$ that is of size at most k.

For d-Promised-Hitting-Set, we design an algorithm with query complexity $\widetilde{\mathcal{O}}(k^d)$.

▶ **Theorem 3.3.** *There exists an algorithm that makes $\widetilde{\mathcal{O}}(k^d)$ GPISE queries and solves d-Promised-Hitting-Set with high probability.*

Here, we give an outline of the algorithm. The first step of designing this algorithm involves, for a positive integer b, a sampling primitive $\mathcal{S}_b$ for the problem. Let $\mathcal{H}$ be the d-uniform hypergraph whose vertex set $U(\mathcal{H})$ is known and hyperedge set $\mathcal{F}(\mathcal{H})$ is unknown to us. Let $h : U(\mathcal{H}) \to [b]$ be a random hash function. Let $U_i = \{u \in U(\mathcal{H}) : h(u) = i\}$, where $i \in [b]$. Note that $U_1, \ldots, U_b$ form a partition of $U(\mathcal{H})$, some of the U_i's can be empty. We make a GPISE query with input $(U_{i_1}, \ldots, U_{i_d})$ for each $1 \leq i_1 < \ldots < i_d \leq b$ such that $U_{i_j} \neq \emptyset \ \forall j \in [d]$. Observe that we make $\mathcal{O}(b^d)$ queries to the oracle. Let $\mathcal{F}'$ be the set of hyperedges that are output by the $\mathcal{O}(b^d)$ GPISE queries. Now, we can generate a sub-hypergraph $\mathcal{H}^h$ of $\mathcal{H}$ such that $U(\mathcal{H}^h) = U(\mathcal{H})$ and $\mathcal{F}(\mathcal{H}^h) = \mathcal{F}'$.

Henceforth, the term edge and graph would essentially mean a hyperedge and a d-uniform hypergraph, respectively.

We find $\alpha \log k$ samples by calling the sampling primitive $\mathcal{S}_{\beta k}$ for $\alpha \log k$ times, where $\alpha = 100d^2$ and $\beta = 100d^3 2^{d+5}$. Let the subgraphs resulting from the sampling be $\mathcal{H}_1, \ldots, \mathcal{H}_{\alpha \log k}$. Let $\hat{\mathcal{H}} = \mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{\alpha \log k}$. Note that we can construct $\hat{\mathcal{H}}$ by making $\widetilde{\mathcal{O}}(k^d)$ GPISE queries. Observe that if we prove the following lemma, then we are done with the proof of Theorem 3.3 (the detailed proof is in the full version [3]).

▶ **Lemma 3.4.** *If $|HS(\mathcal{H})| \leq k$, then $HS(\mathcal{H}) = HS(\hat{\mathcal{H}})$ with high probability.*

To prove Lemma 3.4, we need some intermediate results. We state the following proposition and then define some sets, which will be needed for our analysis.

▶ **Proposition 3.5** ([6])**.** *Let $\mathcal{H}$ be a d-uniform hypergraph. If $|\mathcal{F}(\mathcal{H})| > d!k^d$, then there exists a $(k+1)$-sunflower in $\mathcal{H}$.*

▶ **Definition 3.6.** In the hypergraph $\mathcal{H}$, $\mathcal{C}$ is the set of *large* cores; $\mathcal{F}_s$ is the family of edges that do not contain any *large* core; $\mathcal{C}'$ is the family of *large* cores none of which contain a *significant* core as a proper subset.

The following two results (Lemma 3.7 and 3.8) give useful bounds with respect to the input instances of d-Promised-Hitting-Set.

▶ **Lemma 3.7.** *If $|HS(\mathcal{H})| \leq k$, then $|\mathcal{F}_s| \leq d!(10dk)^d$.*

Proof. If $|\mathcal{F}_s| > d!(10dk)^d$, then there exists a $(10dk+1)$-sunflower $\mathcal{S}$ in $\mathcal{H}$ by Proposition 3.5 such that each edge in $\mathcal{S}$ belongs to $\mathcal{F}_s$. First, since $HS(\mathcal{H}) \leq k$, the core $C_{\mathcal{H}}(\mathcal{S})$ of $\mathcal{S}$ must be non-empty. Note that $C_{\mathcal{H}}(\mathcal{S})$ is a large core and $C_{\mathcal{H}}(\mathcal{S})$ is contained in every edge in $\mathcal{S}$. Observe that we arrived at a contradiction, because any edge in $\mathcal{S}$ is also an edge in $\mathcal{F}_s$ and any edge in $\mathcal{F}_s$ does not contain a large core by definition. Hence, $|\mathcal{F}_s| \leq d!(10dk)^d$. ◀

▶ **Lemma 3.8.** *If $|HS(\mathcal{H})| \leq k$, then $|\mathcal{C}'| \leq (d-1)!k^{d-1}$.*

Proof. Let us consider the set system of all cores in $\mathcal{C}'$. Note that the number of elements present in each core in $\mathcal{C}'$ is at most $d-1$. If $|\mathcal{C}'| > (d-1)! \cdot k^{d-1}$, then there exists a $k+1$-sunflower $\mathcal{S}'$. Let $C_1, \ldots, C_{k+1}$ be the sets present in the sunflower $\mathcal{S}'$ and let $C_{\mathcal{S}'}$ be the core of $\mathcal{S}'$. Observe that if $C_{\mathcal{S}'} = C_1 \cap \ldots \cap C_{k+1} = \emptyset$, then $|HS(\mathcal{H})| > k$.

Now consider the following observation for the case when $C_{\mathcal{S}'}$ is non-empty.

▶ **Observation 3.9.** *If $C_{\mathcal{S}'}$ is non-empty, then $C_{\mathcal{S}'}$ is the pair-wise intersection of a family of $k+1$ edges in $\mathcal{H}$.*

Proof. Let A_i be the set of at least $10dk$ edges that form a sunflower with core C_i, where $i \in [k+1]$. Observe that this is possible as each C_i is a large core. Before proceeding further, note that $C_i \cap C_j = C_{\mathcal{S}'}$ and $(C_i \setminus C_{\mathcal{S}'}) \cap (C_j \setminus C_{\mathcal{S}'}) = \emptyset$ for all $i, j \in [k+1]$ and $i \neq j$.

Consider $B_i \subseteq A_i$ such that for each $F \in B_i$, $F \cap C_j = C_{\mathcal{S}'}$ $\forall j \neq i$ and $|B_i| \geq 9dk$. First, we argue that B_i exists for each $i \in [k+1]$. Recall that for each $j \in [k+1]$, $|C_j| \leq d-1$. Also, for any pair of edges $F_1, F_2 \in A_i$, $(F_1 \setminus C_i) \cap (F_2 \setminus C_i) = \emptyset$. Thus, using the fact that $C_i \cap C_j = C_{\mathcal{S}'}$ for $i \neq j$, a vertex in $C_j \setminus C_{\mathcal{S}'}$ can belong to at most one edge in A_i. This implies that there are at most $(d-1)k < dk$ sets F in A_i such that $F \cap C_j \neq C_{\mathcal{S}'}$ for some $j \neq i \in [k+1]$. We can safely assume that $k+1 \geq d$ and therefore, the number of edges $F \in A_i$ such that $F \cap C_j = C_{\mathcal{S}'}$ $\forall j \neq i \in [k+1]$ is at least $10dk - dk = 9dk$. Next, we argue that there exists $k+1$ edges $F_1, \ldots, F_{k+1}$ such that $F_i \in B_i$ $\forall i \in [k+1]$ and $F_i \cap F_j = C_{\mathcal{S}'}$ for all $i, j \in [k+1]$ and $i \neq j$. We show the existence of the F_i's inductively. For the base case, take any arbitrary edge in B_1 as F_1. Assume that we have chosen $F_1, \ldots, F_p$, where $1 \leq p \leq k$, such that the required conditions hold. We will show that there exists $F_{p+1} \in B_{p+1}$ such that $F_i \cap F_{p+1} = C_{\mathcal{S}'}$ for each $i \in [p]$. By construction of B_i's, no edge in B_{p+1} intersects with $C_i \setminus C_{\mathcal{S}'}, i \leq p$; but every edge in B_{p+1} contains $C_{\mathcal{S}'}$. Also, none of the chosen edges out of $F_1, \ldots, F_p$, intersects $C_{p+1} \setminus C_{\mathcal{S}'}$. So, if we can select an edge $F \in B_{p+1}$ such that $F \setminus C_{p+1}$ is disjoint from $F_i \setminus C_i$ $\forall i \in [p]$, then we are done. Note that for two edges $F', F'' \in B_{p+1}$, $F' \setminus C_{p+1}$ and $F'' \setminus C_{p+1}$ are disjoint. Consider the set $B'_{p+1} \subseteq B_{p+1}$ such that each edge $F \in B'_{p+1}$ intersects with at least one out of $\{F_1 \setminus C_1, \ldots, F_p \setminus C_p\}$. $\left|B'_{p+1}\right| \leq dp \leq dk$, because $(F_i \setminus C_i) \cap (F_j \setminus C_j) = \emptyset$ $\forall i \neq j \in [p]$ and $|F_i| \leq d, i \in [p]$. As $|B_{p+1}| \geq 9dk$, we select any edge in $B_{p+1} \setminus B'_{p+1}$ as F_{p+1}. ◀

The above observation implies the following. If $C_{\mathcal{S}'}$ is non-empty, then there exists a $(k+1)$-sunflower in $\mathcal{H}$. So, $S_{\mathcal{H}}(C_{\mathcal{S}'}) > k$ or equivalently $C_{\mathcal{S}'}$ is a significant core. Note that each C_i contains $C_{\mathcal{S}'}$, which is a significant core; which contradicts the definition of $\mathcal{C}'$. Hence, $|\mathcal{C}'| \leq (d-1)!k^{d-1}$. ◀

The following Lemma provides insight into the structure of $\hat{\mathcal{H}}$ and thereby is the most important part of proving Lemma 3.4.

▶ **Lemma 3.10.** *Let $\hat{\mathcal{H}} = \mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{\alpha \log k}$. If $|HS(\mathcal{H})| \leq k$, then (a) $\mathcal{F}_s \subseteq \mathcal{F}(\hat{\mathcal{H}})$, and (b) $\forall C \in \mathcal{C}', S_{\hat{\mathcal{H}}}(C) > k$ hold with high probability.*

Proof Sketch. First, consider the two claims stated below.

▶ **Claim 3.11.** $\forall i \in [\alpha \log k]$, $\mathbb{P}(F \in \mathcal{F}(\mathcal{H}_i) \mid F \in \mathcal{F}_s) \geq \frac{1}{2}$.

▶ **Claim 3.12.** $\forall i \in [\alpha \log k]$, $\mathbb{P}(S_{\mathcal{H}_i}(C) > k \mid C \in \mathcal{C}') \geq \frac{1}{2}$.

The proofs of Claims 3.11 and 3.12 are involved which we prove below. Observe that Lemma 3.10 follows from Claim 3.11 and 3.12. The detailed proof of Lemma 3.10 is in the full version [3]. ◀

Proof of Claim 3.11. Without loss of generality, we will prove the statement for the graph $\mathcal{H}_1$. Let $h : U(\mathcal{H}) \to [\beta k]$ be the random hash function used in the sampling of $\mathcal{H}_1$. Observe that by the construction of $\mathcal{H}_1$, $F \in \mathcal{F}(\mathcal{H}_1)$ if the following two conditions hold.

- $h(u) = h(v)$ if and only if $u = v$, where $u, v \in F$.
- For any $F' \neq F$ and $F' \in \mathcal{F}(\mathcal{H})$, F' and F differ in the color of at least one vertex.

Hence, $\mathbb{P}(F \notin \mathcal{F}(\mathcal{H}_1) \mid F \in \mathcal{F}_s) \leq \sum_{u,v \in F : u \neq v} \mathbb{P}(h(u) = h(v)) + \mathbb{P}(\mathcal{E}_1)$, where

$\mathcal{E}_1 :$ $\exists$ an edge $F' \in \mathcal{F}(\mathcal{H})$ such that $F' \neq F$ and $\{h(z) : z \in F\} = \{h(z) : z \in F'\}$.

Before we bound the probability of the occurrence of $\mathcal{E}_1$, we show the existence of a set $D \subseteq U(\mathcal{H}) \setminus F$ of bounded cardinality such that each edge in $\mathcal{F}(\mathcal{H}) \setminus \{F\}$ intersects with D.

▶ **Observation 3.13.** *Let $F \in \mathcal{F}_s$. Then there exists a set $D \subseteq U(\mathcal{H}) \setminus F$ such that each edge in $\mathcal{F}(\mathcal{H}) \setminus \{F\}$ intersects with D and $|D| \leq 2^{d+5} d^2 k$.*

Proof. For each $C \subset F$, consider the hypergraph $\mathcal{H}_C$ such that $U(\mathcal{H}_C) = U(\mathcal{H}) \setminus C$ and $\mathcal{F}(\mathcal{H}_c) = \{F' \setminus C : F' \in \mathcal{F}(\mathcal{H})$ and $F' \cap F = C\}$. First, we prove that the size of $HS(\mathcal{H}_C)$ is at most $dS_{\mathcal{H}}(C)$. For the sake of contradiction, assume that $|HS(\mathcal{H}_C)| > dS_{\mathcal{H}}(C)$. Then we argue that there exists $\mathcal{F}' \subseteq \mathcal{F}(\mathcal{H}_C)$ such that each pair of hyperedges in $\mathcal{F}'$ are vertex disjoint and $|\mathcal{F}'| > S_{\mathcal{H}}(C)$. If $|\mathcal{F}'| \leq S_{\mathcal{H}}(C)$, then the vertex set $\{w : w \in F', F' \in \mathcal{F}'\}$ is a hitting set of $\mathcal{H}_c$ and it has size at most $dS_{\mathcal{H}}(C)$, which is a contradiction. Therefore, there is a $\mathcal{F}' \subseteq \mathcal{F}(\mathcal{H}_C)$ such that each pair of hyperedges in $\mathcal{F}'$ is vertex disjoint and $|\mathcal{F}'| > S_{\mathcal{H}}(C)$. Observe that the set of edges $\{F'' \cup C : F'' \in \mathcal{F}'\}$ forms a t-sunflower in $\mathcal{H}$, where $t > S_{\mathcal{H}}(C)$; which contradicts the definition of $S_{\mathcal{H}}(C)$.

The required set D is $(HS(\mathcal{H}) \setminus F) \cup \bigcup_{C \subset F} HS(\mathcal{H}_C)$.

If a hyperedge F^* in $\mathcal{F}(\mathcal{H}) \setminus \{F\}$ intesects with F, then it must intersect with $HS(\mathcal{H}_C)$ for some $C \subset F$; otherwise F^* intersects with $HS(\mathcal{H}) \setminus F$. So, each hyperedge in $\mathcal{F}(\mathcal{H}) \setminus \{F\}$, intersects with D. Now, we bound the size of D.

$$\begin{aligned}
|D| &\leq |HS(\mathcal{H})| + \left| \bigcup_{C \subset F} HS(\mathcal{H}_C) \right| \\
&\leq k + \sum_{C \subset F} dS_{\mathcal{H}}(C) && (\because |HS(\mathcal{H})| \leq k \text{ and } |HS(\mathcal{H}_C)| \leq dS_{\mathcal{H}}(C)) \\
&\leq k + 2^d \cdot d \cdot 10dk && (\because F \text{ does not contain any large core}) \\
&\leq 2^{d+5} d^2 k
\end{aligned}$$

◀

With respect to the set D, we define another event $\mathcal{E}_2$ such that $\mathcal{E}_2 \supseteq \mathcal{E}_1$

$\mathcal{E}_2 :$ $\exists\ z \in D$ such that $h(z) = h(y)$ for some $y \in F$.

We will now bound $\mathbb{P}(\mathcal{E}_2)$.

So, $\mathbb{P}(\mathcal{E}_2) \leq d\frac{|D|}{\beta k} = \frac{d \cdot 2^{d+5} d^2 k}{\beta k} = \frac{d^3 2^{d+5}}{\beta} < \frac{1}{10}$. Putting everything together,

$$\begin{aligned}\mathbb{P}(F \notin \mathcal{F}(\mathcal{H}_1) | F \in \mathcal{F}_s) &\leq \sum_{u,v \in F: u \neq v} \mathbb{P}(h(u) = h(v)) + \mathbb{P}(\mathcal{E}_1) \\ &\leq \frac{d^2}{\beta k} + \mathbb{P}(\mathcal{E}_2) \leq \frac{d^2}{\beta k} + \frac{1}{10} < \frac{1}{2}.\end{aligned}$$

◀

Proof of Claim 3.12. Without loss of generality, we will prove the statement for the graph $\mathcal{H}_1$. Let $h : U(\mathcal{H}) \to [\beta k]$ be the random hash function used in the sampling of $\mathcal{H}_1$.

Let $\mathcal{S}$ be the sunflower with core C and $\mathcal{F}'$ be the set of edges corresponding to sunflower $\mathcal{S}$. Note that $|\mathcal{F}'| > 10dk$. Let $\mathcal{F}'' \subseteq \mathcal{F}'$ be such that $\forall F \in \mathcal{F}'', (F \setminus C) \cap HS(\mathcal{H}) = \emptyset$, and $|\mathcal{F}''| = (10d-1)k$. Note that such an $\mathcal{F}''$ exists as $|\mathcal{F}''| > 10dk$ and $HS(\mathcal{H}) \leq k$.

For $F \in \mathcal{F}''$, let X_F be the indicator random variable that takes value 1 if and only if there exists $F' \in \mathcal{F}'$ such that $F' \in \mathcal{F}(\mathcal{H}_1)$ and $\{h(v) \mid v \in F\} = \{h(v) \mid v \in F'\}$. Define $X = \sum_{F \in \mathcal{F}''} X_F$. Observe that $S_{\mathcal{H}_1}(C)$ is a random variable such that $S_{\mathcal{H}_1}(C) \geq X$. Recall that we have to prove $\mathbb{P}(S_{\mathcal{H}_1}(C) > k \mid C \in \mathcal{C}') \geq \frac{1}{2}$. So, if we can show $\mathbb{P}(X \leq k) < \frac{1}{2}$, then we are done. Observe that $X_F = 1$ if the following events occur.

$\mathcal{E}_1$: $h(u) = h(v)$ if and only if $u = v$, where $u, v \in F$.

$\mathcal{E}_2$: There does not exist $y \in F$ and $z \in HS(\mathcal{H}) \setminus C$ such that $h(y) = h(z)$.

So, $\mathbb{P}(X_F = 1) \geq \mathbb{P}(\mathcal{E}_1 \text{ and } \mathcal{E}_2)$ and using the fact that $|HS(\mathcal{H})| \leq k$, we have

$$\begin{aligned}\mathbb{P}(X_F = 0) &\leq \sum_{u,v \in F; u \neq v} \mathbb{P}(h(u) = h(v)) + \sum_{y \in F} \sum_{z \in HS(\mathcal{H}) \setminus \{u\}} \mathbb{P}(h(y) = h(z)) \\ &\leq \frac{d^2}{\beta k} + d \cdot \frac{|HS(\mathcal{H})|}{\beta k} < \frac{1}{200}\end{aligned}$$

Hence, $\mathbb{E}[X] = \sum_{F \in \mathcal{F}''} \mathbb{P}(X_F = 1) \geq (10d-1)k \cdot \frac{199}{200} > 9dk$.

$$\begin{aligned}\mathbb{P}(X \leq k) &\leq \mathbb{P}(\left|\mathcal{F}''\right| - X \geq (10d-2)k) \ (\because\ |\mathcal{F}''| = (10d-1)k) \\ &\leq \frac{\mathbb{E}[\left|\mathcal{F}''\right| - X]}{(10d-2)k} < \frac{(10d-1)k - 9dk}{(10d-2)k} \leq \frac{d-1}{10d-2} < \frac{1}{2}.\end{aligned}$$

The first inequality is by Markov and second one is due to $\mathbb{E}[X] > 9dk$. ◀

Now, we have all the ingredients to prove Lemma 3.4.

Proof of Lemma 3.4. First, since $\hat{\mathcal{H}}$ is a subgraph of $\mathcal{H}$, a minimum hitting set of $\mathcal{H}$ is also a hitting set of $\hat{\mathcal{H}}$. To prove this Lemma, it remains to show that when $|HS(\mathcal{H})| \leq k$, then a minimum hitting set of $\hat{\mathcal{H}}$ is also a hitting set of $\mathcal{H}$. By Lemma 3.10, it is true that with high probability $\mathcal{F}_s \subseteq \mathcal{F}(\hat{\mathcal{H}})$ and $S_{\hat{\mathcal{H}}}(C) > k$ if $C \in \mathcal{C}'$. It is enough to show that when $\mathcal{F}_s \subseteq \mathcal{F}(\hat{\mathcal{H}})$ and $S_{\hat{\mathcal{H}}}(C) > k$, $\forall C \in \mathcal{C}'$, then a minimum hitting set of $\hat{\mathcal{H}}$ is also a minimum hitting set of $\mathcal{H}$.

First we show that each significant core intersects with $HS(\mathcal{H})$. Suppose there exists a significant core C that does not intersect with $HS(\mathcal{H})$. Let $\mathcal{S}$ be a t-sunflower in $\mathcal{H}$, $t > k$, such that C is the core of $\mathcal{S}$. Then each of the t petals of $\mathcal{S}$ must intersect with $HS(\mathcal{H})$.

But the petals of any sunflower are disjoint. This implies $HS(\mathcal{H}) \geq t > k$, which is a contradiction. So, each significant core intersects with $HS(\mathcal{H})$. As large cores are significant, each large core also intersects with $HS(\mathcal{H})$.

Let us consider a subhypergraph of $\mathcal{H}$, say $\tilde{\mathcal{H}}_1$, with the following definition. Take a large core C_1 in $\mathcal{H}$ that contains a significant core C_2 as a subset. Let $\mathcal{S}_1$ be a sunflower with core C_1. Let $\mathcal{S}_2$ be a sunflower with core C_2 that has more than k petals. Note that there can be at most one hyperedge F_1 of $\mathcal{S}_1$ that is also present in $\mathcal{S}_2$. We delete all hyperedges participating in $\mathcal{S}_1$ except F_1. The remaining hyperedges remain the same as in $\mathcal{H}$. Notice that a hitting set of $\tilde{\mathcal{H}}_1$ is also a hitting set of $\mathcal{H}$; the significant core C_2 remains significant in $\tilde{\mathcal{H}}_1$. Thus, any hitting set of $\tilde{\mathcal{H}}_1$ must intersect with C and therefore, must hit all the hyperedges of $\mathcal{S}_1$. We can think of this as a reduction rule, where the input hypergraph and the output hypergraph have the same sized minimum hitting sets. Let $\tilde{\mathcal{H}}$ be a hypergraph obtained after applying the above reduction rule exhaustively on $\mathcal{H}$. The following properties must hold for $\tilde{\mathcal{H}}$: (i) $HS(\mathcal{H}) = HS(\tilde{\mathcal{H}})$, (ii) all large cores in $\tilde{\mathcal{H}}$ do not contain significant cores as subsets, (iii) all hyperedges of $\mathcal{F}_s$ in $\mathcal{H}$ are still present in $\tilde{\mathcal{H}}$.

By Lemma 3.10, it is also true with high probability that $S_{\tilde{\mathcal{H}}}(C) > k$ when C is a large core of $\tilde{\mathcal{H}}$ that does not contain any significant core as a subset. Note that the arguments in Lemma 3.10 can be made for such large cores without significant cores in $\tilde{\mathcal{H}}$. Thus, we continue the arguments with the assumption that $S_{\hat{\mathcal{H}}}(C) > k$ when C is a large core of $\tilde{\mathcal{H}}$ that does not contain any significant core as a subset.

Now we show that when $HS(\mathcal{H}) \leq k$, $HS(\tilde{\mathcal{H}}) = HS(\hat{\mathcal{H}})$. We know that $\mathcal{F}_s \subseteq \mathcal{F}(\tilde{\mathcal{H}})$. That is, any hyperedge that does not contain any large core as a subset, is present in $\tilde{\mathcal{H}}$. Each hyperedge in $\mathcal{F}_s$ must be covered by any hitting set of $\mathcal{H}$ as well as any hitting set of $\tilde{\mathcal{H}}$ and $\hat{\mathcal{H}}$. Now, it is enough to argue that an hyperedge $F \in \mathcal{F}(\tilde{\mathcal{H}}) \setminus \mathcal{F}_s$, must be covered by any hitting set of $\hat{\mathcal{H}}$. Note that each $F \in \mathcal{F}(\tilde{\mathcal{H}}) \setminus \mathcal{F}_s$ contains a large core, say $\hat{C}$, which does not contain a significant core as a subset. By our assumption, $\hat{C}$ is a significant core in $\hat{\mathcal{H}}$ and therefore, must be hit by any hitting set of $\hat{\mathcal{H}}$.

Putting everything together, when $|HS(\mathcal{H}|) \leq k$, each edge in $\mathcal{H}$ is covered by any hitting set of $\hat{\mathcal{H}}$. Thus, $HS(\mathcal{H}) = HS(\hat{\mathcal{H}})$. ◀

3.3 Algorithms for d-Hitting-Set and d-Decision-Hitting-Set

Now, we explain the algorithms for d-Hitting-Set and d-Decision-Hitting-Set.

▶ **Theorem 3.14.** *d-Hitting-Set can be solved with $\widetilde{\mathcal{O}}(k^{2d})$ GPISE queries.*

Proof. Let $\mathsf{Pack}(\mathcal{H})$ denote a maximum packing of hypergraph $\mathcal{H}$. By Theorem 2.1, with high probability, we can find $\mathsf{Pack}(\mathcal{H})$ if $|\mathsf{Pack}(\mathcal{H})| \geq k+1$ or decide that there does not exist any packing of size $k+1$, by making $\widetilde{\mathcal{O}}(k^{2d})$ GPISE queries.

If $|\mathsf{Pack}(\mathcal{H})| \geq k+1$, then $|HS(\mathcal{H})| \geq k+1$. So, in this case we report that there does not exist any hitting set of size at most k. Otherwise, if $|\mathsf{Pack}(\mathcal{H})| \leq k$, then $|HS(\mathcal{H})| \leq dk$. As $|HS(\mathcal{H})| \leq dk$, $HS(\mathcal{H})$ can be found using our algorithm for d-Promised-Hitting-Set by making $\widetilde{\mathcal{O}}(k^d)$ GPISE queries. If $|HS(\mathcal{H})| \leq k$, with high probability we output $HS(\mathcal{H})$ and if $|HS(\mathcal{H})| > k$, we report there does not exist a hitting set of size at most k. The total query complexity is $\widetilde{\mathcal{O}}(k^{2d})$. ◀

▶ **Theorem 3.15.** *d-Decision-Hitting-Set can be solved with $\widetilde{\mathcal{O}}(k^{2d^2})$ GPIS queries.*

Proof. Observe that, it is enough to give an algorithm that solves d-Decision-Hitting-Set with probability at least $2/3$ by using $\mathcal{O}(k^{2d^2})$ GPIS queries. The details are in the full version [3].

We choose a random hash function $h : U(\mathcal{H}) \to [\gamma k^{2d}]$, where $\gamma = 1009^d d^2$. Let $U_i = \{u \in U(\mathcal{H}) : h(u) = i\}$, where $i \in [\gamma k^{2d}]$. Note that U_i's form a partition of $U(\mathcal{H})$, where some of the U_i's can be empty. We make a GPIS query with input $(U_{i_1}, \ldots, U_{i_d})$ for each $1 \leq i_1 < \ldots < i_d \leq \gamma k^{2d}$ such that $U_{i_j} \neq \emptyset \; \forall j \in [d]$. Recall that the output of a GPIS query is Yes or No. We create a hypergraph $\hat{\mathcal{H}}$ where we create a vertex for each part U_i, $i \in [\gamma k^{2d}]$. We abuse notation and denote $U(\hat{H}) = \{U_1, \ldots, U_{\gamma k^{2d}}\}$ and $\mathcal{F}(\hat{\mathcal{H}}) = \{(U_{i_1}, \ldots, U_{i_d}) :$ GPIS oracle answers yes when given $(U_{i_1}, \ldots, U_{i_d})$ as input$\}$. Observe that we make $\mathcal{O}(k^{2d^2})$ queries to the GPIS oracle. We find $HS(\hat{\mathcal{H}})$ and report $|HS(\mathcal{H})| \leq k$ if and only if $\left|HS(\hat{\mathcal{H}})\right| \leq k$. The correctness of our query procedure follows from the following Lemma (proof is in the full version [3]).

▶ **Lemma 3.16.** *If* $\left|HS(\hat{\mathcal{H}})\right| \leq k$*, then* $|HS(\mathcal{H})| \leq k$ *with probability at least* $2/3$. ◀

References

1. Noga Alon, Raphael Yuster, and Uri Zwick. Color Coding. In *Encyclopedia of Alg.*, pages 335–338. Springer, 2016.
2. Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. In *ITCS*, pages 38:1–38:21, 2018.
3. Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Parameterized Query Complexity of Hitting Set using Stability of Sunflowers. *CoRR*, abs/1807.06272, 2018. `arXiv:1807.06272`.
4. Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via Sampling with Applications to Finding Matchings and Related Problems in Dynamic Graph Streams. In *SODA*, pages 1326–1344, 2016.
5. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Parameterized Algorithms. *Springer*, 2015.
6. P. Erdős and R. Rado. Intersection Theorems for Systems of Sets. *Journal of the London Mathematical Society*, s1-35(1):85–90, 1960.
7. Uriel Feige. On Sums of Independent Random Variables with Unbounded Variance and Estimating the Average Degree in a Graph. *SIAM J. Comput.*, 35(4):964–984, 2006.
8. Oded Goldreich. Introduction to Property Testing. *Cambridge University Press*, 2017.
9. Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.
10. Piotr Indyk, Sepideh Mahabadi, Ronitt Rubinfeld, Ali Vakilian, and Anak Yodpinyanee. Set Cover in Sub-linear Time. In *SODA*, pages 2467–2486, 2018.
11. Kazuo Iwama and Yuichi Yoshida. Parameterized Testability. *TOCT*, 9(4):16:1–16:16, 2018.
12. Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *SODA*, pages 1123–1131, 2012.
13. Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In *ITCS*, pages 39:1–39:16, 2018.