Parameterized Query Complexity of Hitting Set using Stability of Sunflowers

Arijit Bishnu [*] Arijit Ghosh [†] Sudeshna Kolay [‡] Gopinath Mishra [*]
Saket Saurabh [†]

July 18, 2018

Abstract

In this paper, we study the query complexity of parameterized decision and optimization versions of Hitting-Set, Vertex Cover, Packing , Matching and Max-Cut. The main focus is the query complexity of Hitting Set. In doing so, we use an oracle known as BIS introduced by Beame et al. [BHR$^+$18] and its generalizations to hypergraphs. The query models considered are the GPIS and GPISE oracles :

(i) the GPIS oracle takes as input d pairwise disjoint non-empty vertex subsets $A_1, \dots, A_d$ in a hypergraph $\mathcal{H}$ and answers whether there is a hyperedge with vertices in each A_i,

(ii) the GPISE oracle takes the same input and returns a hyperedge that has vertices in each A_i; NULL, otherwise.

The GPIS and GPISE oracles are used for the decision and optimization versions of the problems, respectively. For $d = 2$, we refer GPIS and GPISE as BIS and BISE, 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.

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 at hand. The most generic approach towards this is to ask as few queries to the oracle so that the local copy of the graph is an equivalent sample of the actual graph for the problem at hand. This spawns the study of query complexity. In query complexity of a graph problem, the aim is to design algorithms that have access to the vertices of a graph, but not the edge set. Instead, the algorithm has access to an oracle to which queries can be made. 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 that strike a balance between not revealing too much information and revealing enough information per query to reduce the number of queries. Let us take the example of the classical polynomial time problem of finding a global minimum cut that has led to the introduction of different query models, in

[*] Indian Statistical Institute, Kolkata, India
[†] The Institute of Mathematical Sciences, HBNI, India
[‡] Eindhoven University of Technology, Eindhoven, Netherlands

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 [Fei06, GR08]. Therefore, in the case of the minimum cut problem, the *cut query* was introduced to achieve subquadratic query complexity [RSW18].

There is a vast literature available on the query complexity of problems with classical polynomial time algorithms (Refer to book [Gol17]). However, there has been almost negligible work on algorithmically hard problems [IMR$^+$18, IY18, ORRR12]. In this paper, we use ideas of parameterized complexity in order to study the query complexity of NP-hard problems. The HITTING SET and VERTEX COVER problems are test problems for all new techniques of parameterized complexity and also in every subarea that parameterized complexity has explored. We continue the tradition and study the query complexity of these problems. In doing so, we touch upon some other problems like MAX-CUT, MATCHING and d-PACKING. We define a generalization of a recently introduced query model [BHR$^+$18] (details in Section 1.1). In what follows, we describe the relevant model and state our results.

1.1 The model

Given a graph G, the vertex set of G is denoted as $V(G)$ and the edge set is denoted as $E(G)$. For an edge $e \in E(G)$ with endpoints $u, v \in V(G)$, we denote $e = (u, v)$. Given a hypergraph $\mathcal{H}$, the vertex set and hyperedge sets are denoted by $U(\mathcal{H})$ and $\mathcal{F}(\mathcal{H})$, respectively. The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. For a function $f(k)$, the set of functions of the form $\mathcal{O}(f(k) \cdot \log k)$, is denoted by $\widetilde{\mathcal{O}}(f(k))$.

Our goal in this paper is to look at the parameterized query complexity of graph and hypergraph problems. Motivated by Beame et al. [BHR$^+$18] and Iwama et al. [IY18], we consider the following oracles.

Bipartite independent set oracle (*BIS*): For a graph G, given two disjoint non-empty subsets $A, B \subseteq V(G)$ as input, a BIS query oracle answers whether there exists an edge $(u, v) \in E(G)$ such that $u \in A$ and $v \in B$.

Given two vertices $u, v \in V(G)$, the often used *edge existence query* asks for an yes/no answer to the question whether there exists an edge between u and v. The BIS oracle, proposed by Beame et al. [BHR$^+$18], is a generalization over the *edge existence query* in the sense that it asks for the existence of an edge between two disjoint sets of vertices. BIS was used to estimate the number of edges in a graph. We will use BIS oracle to solve decision problems. The following oracle is a generalization of BIS to the hypergraph setting.

Generalized d-partite independent set oracle (*GPIS*): For a d-uniform hypergraph $\mathcal{H}$, given d pairwise disjoint non-empty subsets $A_1, A_2, \ldots, A_d \subseteq U(\mathcal{H})$ 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 $i \in [d]$.

To solve optimization problems, we extend BIS and GPIS to BISE and GPISE, respectively.

Bipartite independent set edge oracle (*BISE*): For a graph G, given two disjoint non-empty subsets of the vertices $A, B \subseteq V(G)$ as input, a BISE query oracle outputs an edge $(u, v) \in E(G)$ such that $u \in A$ and $v \in B$ if such an edge exists; otherwise, the BISE oracle reports NULL.

Generalized d-partite independent set edge oracle (*GPISE*): For a d-uniform hypergraph $\mathcal{H}$, given d pairwise non-empty disjoint subsets $A_1, A_2, \ldots, A_d \subseteq U(\mathcal{H})$ 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 BIS oracle, and GPISE oracle is same as BISE. Both BIS and BISE oracles reveal the same information (that there is no edge between the two disjoint set of vertices) if the answer returned by them turn out to be no. For a yes answer, BISE has more power than BIS in the sense that BISE returns two vertices, one each in the two disjoint sets, between which there is an edge, whereas BIS just lets us know that there is an edge going across the partition with no specificity about the vertices between which there is an edge. This power of BISE allows us to sample edges from a graph. So, BISE is more *powerful* than BIS, and GPISE is more powerful than GPIS. This distinction is crucial and will be evident when we discuss VERTEX-COVER and HITTING SET.

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 [Fei06, GR08]. This necessitates looking at queries that have more power in the sense that it goes beyond obtaining local information and generalizes earlier queries. Beame et al. [BHR$^+$18] introduced BIS query model and gave an approximation algorithm for estimating the number of edges in a graph. To get a clear motivation behind BIS query, please refer to [BHR$^+$18].

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 et al. [IY18] initiated the study of parameterized version of some NP-Hard problems in the graph property testing framework with access to standard oracles. The graph property testing framework implies that the algorithm will be correct only for *stable instances* [1] as input. Note that the query complexity in their paper is exponential in terms of the parameter. 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

(i) BIS (BISE) oracle to study parameterized decision (optimization) version of VERTEX-COVER;

(ii) GPIS (GPISE) oracle to study parameterized decision (optimization) version of HITTING SET;

(iii) BISE (GPISE) oracle to study optimization version of MATCHING (d-PACKING);

(iv) BIS (BISE) oracle to study parameterized decision (optimization) version of MAX-CUT.

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

In our framework, the input consists only of vertices and not the edges or hyperedges. Solving a problem on such an input means making queries to oracles and using the outcomes of the queries to build a reduced instance, on which one can run the traditional (FPT) algorithms. While stating the results, we will bother only about the number of queries required to biuld 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

[1] By a stable instance, we mean that the input graph either satisfies the required propety or we have to modify the existence of at least ϵm many edges to make the graph satisfy the property, where m is the number of edges in the input graph.

probability (Section 2 has the definition of high probability). Our methods use the technique of Color Coding [AYZ16, CFK$^+$15] to restrict the number of queries required to generate a reduced instance of interest. We consider the following problems in this paper.

VERTEX-COVER and d-HITTING-SET The results on VERTEX-COVER and HITTING SET are the main focus of this paper. The VERTEX-COVER and d-HITTING-SET problems are defined as follows.

VERTEX-COVER
Input: The set of vertices $V(G)$ of a graph G, the access to a BISE oracle, and a positive integer k.
Output: A set of vertices of G, of size at most k, that contains at least one end ponit from each edge if such a set exists. Otherwise, we report such a set does not exist.

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.

Note that, in this paper, we consider d as a constant and $VC(G)$ ($HS(\mathcal{H})$) denotes a minimum vertex cover (hitting set) of G ($\mathcal{H}$). The DECISION-VERTEX-COVER and d-DECISION-HITTING-SET problems are the usual decision versions of VERTEX-COVER and d-HITTING-SET, respectively. DECISION-VERTEX-COVER will have oracle access to BIS instead of BISE; and d-DECISION-HITTING-SET will have oracle access to GPIS instead of GPISE. PROMISED-VERTEX-COVER is the promised version of VERTEX-COVER where it is promised that the size of the minimum vertex cover is at most k. Similarly, d-PROMISED-HITTING-SET is the promised version of d-HITTING-SET.

MATCHING and d-PACKING Our solutions to VERTEX-COVER and d-HITTING-SET need us to solve MATCHING and d-PACKING, respectively. The problems are formally stated as follows.

MATCHING
Input: The set of vertices $V(G)$ of a graph G, the access to a BISE oracle, and a positive integer k.
Output: A pairwise disjoint set of at least k edges if such a set of edges exists. Otherwise, we report such a set of edges does not exist.

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.

CUT In this paper, we show that CUT, as defined below, can be solved deterministically.

CUT
Input: The set of vertices $V(G)$ of a graph G, access to a BISE oracle, and two positive integers t and k.
Output: A t-partition $V_1 \uplus \ldots \uplus V_t$ of $V(G)$ such that the number of edges having endpoints in different parts is at least k if such a partition exists. Otherwise, we report such a set of partition does not exist.

DECISION-CUT denotes the usual decision version of the problem CUT, but the oracle access is to BIS instead BISE. Our results are summarized in Tables 1 and 2.

Problems	Query Oracles	
	BIS	BISE
Vertex-Cover	—	$\widetilde{\mathcal{O}}(k^4)$
Decision-Vertex-Cover	$\widetilde{\mathcal{O}}(k^8)$	$\widetilde{\mathcal{O}}(k^4)$
Promised-Vertex-Cover	—	$\widetilde{\mathcal{O}}(k^2)$
Matching	—	$\widetilde{\mathcal{O}}(k^4)$
Cut	—	$\widetilde{\mathcal{O}}(k^4)$
Decision-Cut	$\widetilde{\mathcal{O}}(k^4)$	$\widetilde{\mathcal{O}}(k^4)$

Table 1: Query complexities for graph problems using BIS and BISE oracles.

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	—	$\widetilde{\mathcal{O}}(k^{2d})$

Table 2: Query complexities for hypergraph problems using GPIS and GPISE oracles.

1.3 Related Works

Several query complexity models have been proposed in literature to study various problems [Fei06, GR08]. The only work prior to ours related to parameterization in the query complexity model was by Iwama and Yoshida [IY18]. 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 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 and have a simple unifying structure in terms of asking for the existence of an edge between two disjoint sets of vertices. 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. The main focus of this paper is our results on Vertex Cover and Hitting Set. Though the two problems look similar, the details of Hitting Set requires more involved and uses the *stablity of sunflowers*. In contrast to the work of Iwama and Yoshida, our algorithm uses BISE query for the vertex cover problem; it neither knows the number of edges, nor estimates it. 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. For the *promised version* of the problem, where it is known that the vertex cover is bounded by k, we can give an algorithm that makes $\widetilde{\mathcal{O}}(k^2)$ BISE queries.

While on one side there has been initial interest in the study of parameterized problems in the query complexity model, on the other hand, recent papers have considered strengthened query complexity models. The BIS model was introduced by Beame et al. [BHR+18] to design better edge estimation algorithms. Note that if the number of edges is bounded by k, then we can extract all the edges using $\mathcal{O}(k^4)$ BISE queries by the same technique we describe for solving MATCHING in Section 3. In the same work [BHR$^+$18], the IS oracle was also introduced, to estimate the number of edges, where the input 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 [RSW18], for designing algorithms with better query complexity for the MIN-CUT and $s-t$-MIN-CUT problems, for a graph G, the oracle took as input a vertex subset $S \subseteq V(G)$ and output the value $c(S)$ of the cut between S and $V(G) \setminus S$. The corresponding query complexities for the MIN-CUT and $s-t$-MIN-CUT problems are $\widetilde{\mathcal{O}}(n)$ and $\widetilde{\mathcal{O}}(n^{5/3})$ [2]; respectively. Thus, oracles are being designed with the allowance to return more information than the earlier query complexity models.

Apart from the problems discussed above, there is a vast literature on query complexity of graph problems. The models considers are mostly *dense graph model*, *bounded degree graph model* and *general graph model*. Some of the problems considered are estimation of the number of edges, testing regularity of a graph, testing bipartiteness, subgraph freeness, testing connectivity etc. To have a detailed view of known results on query complexity of graph problems, refer to book [Gol17].

Organization of the paper In Section 3, we discuss our results on MATCHING and PACKING. In Section 4, we give the algorithms to solve VERTEX-COVER and DECISION-VERTEX-COVER. We describe our algorithms for d-HITTING-SET and d-DECISION-HITTING-SET in Section 5. At the end, in Section 6, we discuss the algorithm for CUT and DECISION-CUT

2 Preliminaries

Given a *graph* G, the vertex set of G is denoted as $V(G)$ and the edge set is denoted as $E(G)$. For an edge $e \in E(G)$ with endpoints $u, v \in V(G)$, we will use the notation $e = (u, v)$. For a vertex $u \in V(G)$, the degree of u is denoted by $\mathsf{deg}_G(u)$. The *neigborhood* $N_G(u)$ denotes the neighbor set $\{v : (u, v) \in E(G)\}$ of u. Given graphs G_1, G_2 defined on the same set of n vertices, the graph $G_1 \cup G_2$ is such that $V(G_1 \cup G_2) = V(G_1) = V(G_2)$ and $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$. A *subgraph* G' of G is said to be induced by vertex set $V \subseteq V(G)$ if $V(G') = V$ and $E(G') = \{(u, v) \in E(G) \mid u, v \in V\}$. A *t-cut* of a graph G corresponds to a partition of $V(G)$ into t parts, and refers to the subset of edges in $E(G)$ that have the two end points in two different parts of the partition. The size of the t-cut is the number of edges that have end points in different partitions.

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. All hyperedges of a *d-uniform hypergraph* are of size d.

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 following observation is important for the analysis of algorithms described in this paper.

Observation 2.1. (i) Let Π be a parameterized maximization (minimization) problem and let (I, k) be an instance of Π. Let $\mathcal{A}$ be a randomized algorithm for Π, with success probability at least p, where $0 < p < 1$ is a constant. Then, if we repeat $\mathcal{A}$ for $C \log k$ times for a suitably large constant C and report the maximum (minimum) sized output over $C \log k$ outcomes, then the event that $\mathcal{A}$ succeeds occurs with high probability. If the

[2] In this particular instance $\widetilde{\mathcal{O}}(f)$ denotes $\mathcal{O}(f \log^c n)$ for some constant c.

query complexity of algorithm $\mathcal{A}$ is q, then the query complexity of the $C \log k$ repetitions of $\mathcal{A}$ is $\widetilde{\mathcal{O}}(q)$.

(ii) Let Π be a parameterized decision problem and let (I, k) be an instance of Π. Let $\mathcal{A}$ be a randomized algorithm for Π, with success probability at least p, where $\frac{1}{2} < p < 1$ is a constant. Then, if we repeat $\mathcal{A}$ for $C \log k$ times for a suitably large constant C and report the *majority* of the $C \log k$ outcomes, then the event that $\mathcal{A}$ succeeds occurs with high probability. If the query complexity of algorithm $\mathcal{A}$ is q, then the query complexity of the $C \log k$ repetitions of $\mathcal{A}$ is $\widetilde{\mathcal{O}}(q)$.

Representative Sets: Let $\mathcal{H}$ be a hypergraph. $\mathcal{F}' \subseteq \mathcal{F}(\mathcal{H})$ is said to be a *k-representative set* corresponding to $\mathcal{H}$ if the following is satisfied. For any $X \subset U(\mathcal{H})$ of size k, if there is a exists $F \in \mathcal{F}(\mathcal{H})$ such that $X \cap F = \emptyset$ then there exists $F' \in \mathcal{F}'$ such that $X \cap F' = \emptyset$.

The following proposition gives a bound on the size of a k- representative set corresponding to a d-uniform hypergraph.

Proposition 2.2 ([BT81]). *If $\mathcal{H}$ is d-uniform hypergraph, then there exists a $\binom{k+d}{d}$ size k-representative set corresponding to $\mathcal{H}$.*

Corollary 2.3 ([CFK+15]). *For a set system $\mathcal{H}$ as above, consider the family $\mathcal{S} = \{U(F) \mid F \in \mathcal{F}(\mathcal{H})\}$ and let $\hat{\mathcal{S}}$ be a k-representative of $\mathcal{S}$ as obtained in Proposition 2.2. Let $\mathcal{H}'$ be the set system where $U(\mathcal{H}') = \bigcup_{S \in \hat{\mathcal{S}}} S$ and $\mathcal{F}(\mathcal{H}') = \{F \in \mathcal{F} \mid U(F) \in \hat{\mathcal{S}}\}$. $(\mathcal{H}, k)$ is a YES instance of d-*DECISION-HITTING-SET *if and only if $(\mathcal{H}', k)$ is a YES instance of* d-DECISION-HITTING-SET.

3 Algorithms for d-PACKING

Recall the definition of MATCHING and d-PACKING defined in Section 1.2. MATCHING in a graph is a special case of d-PACKING, we explain the intuition behind the query procedure 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 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 (see Observation 2.1 in Section 2). 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 3.1. *The d-PACKING problem can be solved with $\widetilde{\mathcal{O}}(k^{2d})$* GPISE *queries.*

Proof. By Observation 2.1, it is enough to give an algorithm that solves d-PACKING with constant probability by using $\mathcal{O}(k^{2d})$ GPISE queries.

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 $O(k^{2d})$ 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 the following Lemma 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 3.2. *If $|\mathsf{Pack}(\mathcal{H})| \geq k$, then $\left|\mathsf{Pack}(\hat{\mathcal{H}})\right| \geq k$ with probability at least $2/3$.*

Proof. Let us fix $\mathcal{P} \subseteq \mathcal{F}(\mathcal{H})$ such that $\mathcal{P}$ is a packing of $\mathcal{H}$ and $|\mathcal{P}| = k$. Let $U_{\mathcal{P}}$ be the set of vertices that intersect with some edge in $\mathcal{P}$. Note that $|U_{\mathcal{P}}| = dk$. Let $\mathcal{E}_1$ be the event that the vertices of $U_{\mathcal{P}}$ are uniquely colored, i.e., $\mathcal{E}_1$: $h(u) = h(v)$ if and only if $u = v$, where $u, v \in U_{\mathcal{P}}$. Now we lower bound the event $\mathcal{E}_1$. Let $\mathcal{E}_1^c$ be the compliment of the event $\mathcal{E}_1$.

$$\mathbb{P}(\mathcal{E}_1^c) \leq \sum_{u,v \in U_{\mathcal{P}}} \mathbb{P}(h(u) = h(v)) \leq \sum_{u,v \in U_{\mathcal{P}}} \frac{1}{\gamma k^2} \leq \frac{|U_{\mathcal{P}}|^2}{\gamma k^2} < \frac{1}{3}.$$

Note that the above bound follows from the fact that $\gamma = 100d^2$.

So, $\mathbb{P}(\mathcal{E}_1) \geq \frac{2}{3}$. Let Prop be the property that for each $F \in \mathcal{P}$, there is an "equivalent" hyperedge in $\mathcal{F}(\hat{\mathcal{H}})$. More specifically, Prop is the following property: For each $(u_1, \ldots, u_d) \in \mathcal{P}$, there is a $(u_1', \ldots, u_2') \in \mathcal{F}(\hat{\mathcal{H}})$ such that $h(u_i) = h(u_i')$ for all $i \in [d]$.

From the definition of the GPISE query oracle, observe that the property Prop is true whenever the event $\mathcal{E}_1$ occurs. If we show that the occurrence of Prop implies our claim, we are done.

For the rest of the proof, assume that Prop holds. We show that there exists a packing $\hat{\mathcal{P}}$ of $\hat{\mathcal{H}}$ such that $\left|\hat{\mathcal{P}}\right| \geq k$. For each $(u_1, \ldots, u_d) \in \mathcal{P}$, we add $(u_1', \ldots, u_2') \in \mathcal{F}(\hat{\mathcal{H}})$ such that $h(u_i) = h(u_i')$ for all $i \in [d]$, to $\hat{\mathcal{P}}$. As Prop holds, observe that $\hat{P}$ is a packing of $\hat{\mathcal{H}}$. □

□

For a universe U, a family of *k-perfect hash functions* is a family of hash functions $h : U \to [\ell]$ with $\ell \geq k$, such that for any subset $S \subseteq U$, with $|S| \leq k$, there is a hash function in the family that maps S injectively into $[\ell]$. Using Proposition 3.3 and suitably many colors, we can make the procedure deterministic.

Proposition 3.3 ([AYZ16])**.** *For a universe U of n elements and a positive integer k, there is a family $\mathcal{B}$ of k-perfect hash functions $h : U \to [k^2]$ of size $\mathcal{O}(k^{\mathcal{O}(1)} \log n)$ and this family can be constructed in $\mathcal{O}(k^{\mathcal{O}(1)} n \log n)$ time.*

Corollary 3.4. *The d-Packing problem can be solved by a deterministic algorithm with $\mathcal{O}\left(k^{2d+\mathcal{O}(1)} \log n\right)$ GPISE queries.*

Proof. As before, let $\mathsf{Pack}(\mathcal{H})$ be a maximum packing of $\mathcal{H}$. With out loss of generality, assume that $|\mathsf{Pack}(\mathcal{H})| \geq k$. Otherwise, no algorithm can report a packing of size at least k. Let $\mathcal{P}$ be a subset of k hyperedges from $\mathsf{Pack}(\mathcal{H})$. Also, let $U_{\mathcal{P}}$ denote the subset of vertices that are incident to some hyperedge of $\mathcal{P}$. Note that $|U_{\mathcal{P}}| \leq dk$. Consider a hash function $h : V(G) \to [100d^2k^2]$ with the property that all the vertices of $U_{\mathcal{P}}$ receive distinct colors. If we have such a hash function, then we have a deterministic algorithm for the d-Packing problem with $\widetilde{\mathcal{O}}(k^{2d})$ GPISE queries. In other words, the only place of randomization in the above algorithm was in choosing a random hash function in the beginning. Now consider the $\mathcal{O}(k^{\mathcal{O}(1)} \log n)$-sized family of dk-perfect hash functions described in Proposition 3.3. For each of the $\mathcal{O}(k^{\mathcal{O}(1)} \log n)$ hash functions of the family, we run the above algorithm. Finally, we output the maximum sized packing over all the hash functions. By definition of this family, there is a hash function h in this family which gives distinct colors to all the vertices of $U_{\mathcal{P}}$ and therefore at least for this hash function, the packing returned is of size at least k. Thus, the algorithm is correct. The query complexity of this algorithm is $\mathcal{O}(k^{\mathcal{O}(1)} \log n) \cdot \mathcal{O}(k^{2d}) = \mathcal{O}\left(k^{2d+\mathcal{O}(1)} \log n\right)$. □

4 Algorithm for Vertex-Cover

In this section, we give algorithms for Vertex-Cover and Decision-Vertex-Cover. We prove the following theorems for Vertex-Cover and Decision-Vertex-Cover.

Theorem 4.1. *There exists an algorithm that makes $\widetilde{\mathcal{O}}(k^4)$ queries to a* BISE *oracle and solves* Vertex-Cover *with high probability.*

Theorem 4.2. *There exists an algorithm that makes $\widetilde{\mathcal{O}}(k^8)$ queries to a* BIS *oracle and solves* Decision-Vertex-Cover *with high probability.*

Before designing the algorithm for Vertex-Cover, we first design an algorithm admitting query complexity of $\widetilde{\mathcal{O}}(k^2)$ for a *promised version* of this problem, where we are guaranteed that the input instance has a vertex cover of size at most k. We use this algorithm as a subroutine to design an algorithm for the non-promised optimization version, that has query complexity $\widetilde{\mathcal{O}}(k^4)$ (See the proof of Theorem 4.1). The main idea of the promised version is to sample a subgraph having bounded number of edges using $\widetilde{\mathcal{O}}(k^2)$ BISE queries, such that the vertex cover of the original graph is a vertex cover of the sampled graph and vice versa. We define a vertex as having either *high* degree or *low* degree. We define high and low degrees in such a way that each high degree vertex must be present in any vertex cover of the original graph. The promise that the vertex cover is bounded by k, will help us

(i) to bound the number of high degree vertices and the number of edges where both end points are low degree vertices,

(ii) to guarantee that all the high degree vertices must be of *sufficient* degree in the sampled graph such that any vertex cover of the sampled graph contains all the high degree vertices with high probability, and

(iii) to guarantee all the edges where both end points are low degree vertices, are present in the sampled graph with high probability.

Observe that if we ensure the above properties, we can report the vertex cover of the sampled graph and we will be correct with high probability. The formal arguments are given in Section 4.1. Our analysis is inspired by the analysis of the streaming algorithm for Vertex-Cover [CCE+16], however our analysis uses simpler arguements.

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 3. 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.

For the algorithm for Decision-Vertex-Cover, we have access to the BIS oracle and obtain an algorithm with query complexity $\widetilde{\mathcal{O}}(k^4)$ (see the proof of Theorem 4.2). The main idea to solve Decision-Vertex-Cover is to use the concept of *representative sets* discussed in Section 2. By Proposition 2.2, the k-representative set corresponding to the vertex cover problem is bounded by $\mathcal{O}(k^2)$. Thus, the number of vertices that are present in the k-representative set is also bounded by $\mathcal{O}(k^2)$. All the $\mathcal{O}(k^2)$ vertices will be uniquely colored with high probability if we color the vertices of the graph by enough number of colors by using a suitable hash function. Then we make BIS queries to extract a sufficient number of edges such that the representative set is *embedded* in the sampled subgraph. The formal arguments are given in Section 4.2.

4.1 Promised-Vertex-Cover

In this part, we are going to design an algorithm for the following problem.

Promised-Vertex-Cover
Input: The set of vertices $V(G)$ of a graph G such that $|VC(G)| \leq k$ and the access to a BISE oracle.
Output: A vertex cover of G of size at most k.

For Promised-Vertex-Cover, we design an algorithm with bounded query complexity.

Theorem 4.3. *There exists an algorithm that makes $\widetilde{\mathcal{O}}(k^2)$ BISE queries and solves Promised-Vertex-Cover with high probability.*

As in the case of d-Promised-Hitting-Set, we first describe a sampling primitive $\mathcal{S}_b$ for the problem, where b is an integer. Let G be a graph whose vertex set $V(G)$ of n vertices is known, but the edge set $E(G)$ is unknown to us.

For a given integer b, let $h : V(G) \to [b]$ be a random hash function. Let $V_i = \{v \in V(G) : h(v) = i\}$, where $i \in [b]$. Note that $\{V_1, \ldots, V_b\}$ form a partition of V, where some of the V_i's can be empty. For each $i \neq j$ such that $V_i, V_j \neq \emptyset$, if we make a BISE query with input (V_i, V_j), then observe that we make $\mathcal{O}(b^2)$ queries to the oracle. Let E' be the set of edges that are ouput of $\mathcal{O}(b^2)$ BISE queries. Now, we can generate a subgraph G^h of G due to the sampling primitive $\mathcal{S}_b$.

Let $VC(G)$ denotes the minimum vertex cover of G. We find $100 \log k$ samples with the primitive $\mathcal{S}_{1000k}$. Let these sample subgraphs be $G_1, \ldots, G_{100 \log k}$. Let $\hat{G} = G_1 \cup \ldots \cup G_{100 \log k}$. The following is the main lemma to prove Theorem 4.3.

Lemma 4.4. *If $|VC(G)| \leq k$, then $VC(G) = VC(\hat{G})$ with high probability.*

To prove the above lemma we need some intermediate results.

We say that a vertex of $V(G)$ has *high degree* if its degree is at least $20k$. Otherwise, the vertex is said to have *low* degree. Let V_h denote the set of high degree vertices of $V(G)$, while V_ℓ denotes the set of low degree vertices of $V(G)$. Note that $V_h \uplus V_\ell = V(G)$. Also, let $E_\ell = \{(u, v) \in E(G) \mid u, v \in V_\ell\}$.

Observation 4.5. If $|VC(G)| \leq k$, then $|V_h| \leq k$ and $|E_\ell| \leq 20k^2$.

Proof. First, we prove that $|V_h| \leq k$. Notice that if there is a vertex $u \in V_h \setminus VC(G)$, then it must be the case that $N_G(u) \subseteq VC(G)$. Since $|VC(G)| \leq k$ and $|N_G(u)| \geq 20k$, no such vertex u exists and $V_h \subseteq VC(G)$. That is, $|V_h| \leq |VC(G)| \leq k$.

Next, we prove that $|E_\ell| \leq 20k^2$. Each edge in E_ℓ must be covered by some vertex of V_ℓ. As $|VC(G)| \leq k$, there must exist a set $V' \subseteq V_\ell$ of at most k vertices such that V' covers E_ℓ. Note that each vertex in V_ℓ can cover at most $20k$ edges. Hence, $|E_\ell| \leq 20k^2$. □

Lemma 4.6. *Let $\hat{G} = G_1 \cup G_2 \cup \ldots G_{100 \log k}$, where each G_i is a result of sampling from $\mathcal{S}_{1000k}$. If $VC(G) \leq k$, the following two conditions hold with high probability: (i) $E_\ell \subseteq E(\hat{G})$ and (ii) for each $u \in V_h$, $\mathsf{deg}(u) \geq 2k$.*

Proof. First consider the two claims stated below. We will prove these claims later.

Claim 4.7. *For $i \in [100 \log k]$, $\mathbb{P}(e \in E(G_i) \mid e \in E_\ell) \geq \frac{1}{2}$.*

Claim 4.8. *For $i \in [100 \log k]$, $\mathbb{P}(\mathsf{deg}_{G_i}(u) \geq 2k \mid u \in V_h) \geq \frac{1}{2}$.*

Recall that $\hat{G} = G_1 \cup \ldots \cup G_{100 \log k}$. Using Claims 4.7 and 4.8, we get

$$\mathbb{P}(e \notin E(\hat{G}) \mid e \in E_\ell) \leq \left(1 - \frac{1}{2}\right)^{100 \log k} \leq \frac{1}{k^{100}}$$

and

$$\mathbb{P}(\mathsf{deg}_{\hat{G}}(u) < 2k \mid u \in V_h) \leq \left(1 - \frac{1}{2}\right)^{100 \log k} \leq \frac{1}{k^{100}}.$$

Using the union bound on probabilities of events together with Observation 4.5, we can deduce the following

$$\mathbb{P}(E_\ell \nsubseteq E(\hat{G})) \leq \sum_{e \in E_\ell} \mathbb{P}(e \notin E(\hat{G}) \mid e \in E_\ell) \leq \frac{20}{k^{98}}$$

and

$$\mathbb{P}(\exists\ u \in V_h \text{ such that } \mathsf{deg}_{\hat{G}}(u) < 2k) \leq \sum_{u \in V_h} \mathbb{P}(\mathsf{deg}_{\hat{G}}(u) < 2k \mid u \in V_h) \leq \frac{1}{k^{99}}.$$

Hence, by the union bound

$$\mathbb{P}(E_\ell \nsubseteq E(\hat{G}) \text{ or } \exists\ u \in V_h \text{ such that } \mathsf{deg}_{\hat{G}}(u) < 2k) \leq \frac{21}{k^{98}}.$$

This implies that with high probability, $E_\ell \subseteq E(\hat{G})$ and for each $u \in V_h$, $\mathsf{deg}_{\hat{G}}(u) \geq 2k$. □

Proof of Lemma 4.4. First, since $\hat{G}$ is a subgraph of G, a minimum vertex cover of G is also a vertex cover of $\hat{G}$. To prove the statement of Lemma 4.4, it remains to show that when $|VC(G)| \leq k$, then a minimum vertex cover of $\hat{G}$ is also a vertex cover of G. Recall that $\hat{G} = G_1 \cup \ldots \cup G_{100 \log k}$. By Lemma 4.6, when $|VC(G)| \leq k$, with high probability it is true that $E_\ell \subseteq E(\hat{G})$ and for each $u \in V_h$, $\mathsf{deg}_{G'}(u) \geq 2k$. It is enough to show that when $E_\ell \subseteq E(\hat{G})$ and $deg_{G'}(u) \geq 2k$ for each $u \in V_h$, then a minimum vertex cover of $\hat{G}$ is also a vertex cover for G. From Observation 4.5, $V_h \subseteq VC(G)$. By the conditions that we have assumed, $\mathsf{deg}_{\hat{G}}(u) \geq 2k$ for each $u \in V_h$. Therefore, by arguments similar to Observation 4.5, every vertex of V_h must be in any vertex cover of $\hat{G}$.

As V_h is a subset of any vertex cover of G, all the edges that have one vertex in V_h, are covered by any vertex cover of G. Note that the edges that cannot be covered by vertices from V_h are the edges in E_ℓ. For a minimum vertex cover in G, the edges of E_ℓ remain to be covered. However, by the conditions above, $E_\ell \subseteq E(\hat{G})$. Thus, any vertex cover of $\hat{G}$ must also cover each edge in E_ℓ. Hence, a minimum vertex cover of $\hat{G}$ is also a vertex cover for G.

Thus, when $|VC(G)| \leq k$, $VC(G) = VC(\hat{G})$. □

Proof of Claim 4.7. Without loss of generality, we will prove the statement for the graph G_1. Let $h : V(G) \to [1000k]$ be the random hash function used in the sampling of G_1.

Let $e = (u, v)$. Observe that by the construction of G_1, $e \in E(G_1)$ if the following two conditions hold.

- $h(u) \neq h(v)$.
- There does not exist any edge $e' \neq e$ such that one end point of e' is mapped to $h(u)$ and the other end point is mapped to $h(v)$.

Hence,

$$\mathbb{P}(e \notin E(G_1) | e \in E_\ell) \leq \mathbb{P}(h(u) = h(v)) + \mathbb{P}(\mathcal{E}_1),$$

where $\mathcal{E}_1$ is the event defined as follows.

$\mathcal{E}_1$: $\exists$ an edge $e' \neq e$ such that one end point of e' is mapped to $h(u)$ and the other end point is mapped to $h(v)$.

Let us define another event $\mathcal{E}_2$, which is a superset of $\mathcal{E}_1$. For this, consider the set $S = (VC(G) \cup N_G(u) \cup N_G(v)) \setminus \{u, v\}$.

$\mathcal{E}_2$: $\exists\ z \in S$ such that $h(z) = h(u)$ or $h(z) = h(v)$.

Observe that $\mathcal{E}_1 \subseteq \mathcal{E}_2$ as any edge $e' \neq e$ must have an end point in S. By definition of E_ℓ, $|S| \leq k + 20k + 20k = 41k$. So, $\mathbb{P}(\mathcal{E}_2) \leq 2 \cdot \frac{|S|}{1000k} < \frac{1}{10}$.

Putting everything together,

$$\mathbb{P}(e \notin E(G_1) | e \in E_\ell) \leq \mathbb{P}(h(u) = h(v)) + \mathbb{P}(\mathcal{E}_1) \leq \frac{1}{1000k} + \mathbb{P}(\mathcal{E}_2) \leq \frac{1}{1000k} + \frac{1}{10} < \frac{1}{2}.$$

□

Proof of Claim 4.8. Without loss of generality, we will prove the statement for the graph G_1. Let $h : V \to [1000k]$ be the random hash function used in the sampling of G_1.

Let $N'_G(u) \subseteq N_G(u) \setminus VC(G)$ be such that $|N'_G(u)| = 19k$. Note that such an $N'_G(u)$ exists as $N_G(u) \geq 20k$ and $VC(G) \leq k$.

For $v \in N'_G(u)$, let X_v be the indicator random variable that takes value 1 if and only if there exists $w \in N_G(u)$ such that $(u, w) \in E(G_1)$ and $h(v) = h(w)$. Define $X = \sum\limits_{v \in N'_G(u)} X_v$. Observe that $\deg_{G_1}(u)$ is a random variable such that $\deg_{G_1}(u) \geq X$. Recall that we have to prove $\mathbb{P}(\mathsf{deg}_{G_1}(u) \geq 2k \mid u \in V_h) \geq \frac{1}{2}$. So, if we can show $\mathbb{P}(X < 2k) < \frac{1}{2}$, then we are done.

Observe that $X_v = 1$ if the following two conditions hold.

- $h(u) \neq h(v)$.
- $h(u), h(v) \neq h(z)\ \forall z \in VC(G) \setminus \{u\}$.

So,

$$\mathbb{P}(X_v = 1) \quad \geq \quad \mathbb{P}(h(u) \neq h(v) \text{ and } h(u), h(v) \neq h(z)\ \forall z \in VC(G) \setminus \{u\})$$

and

$$\begin{aligned} \mathbb{P}(X_v = 0) \quad &\leq \quad \mathbb{P}(h(u) = h(v)) + \sum_{z \in VC(G)\setminus\{u\}} \mathbb{P}(h(u) = h(z)) + \sum_{z \in VC(G)\setminus\{u\}} \mathbb{P}(h(v) = h(z)) \\ &\leq \quad \frac{1}{1000k} + \frac{|VC(G)|}{1000k} + \frac{|VC(G)|}{1000k} \quad < \quad \frac{1}{200}. \end{aligned}$$

The last inequality follows from the fact that $|VC(G)| \leq k$. Hence, $\mathbb{E}[X] = \sum\limits_{v \in N'_G(u)} \mathbb{P}(X_v = 1) \geq 19k \cdot \frac{199}{200} > 18k$.

As $|N'_G(u)| = 19k$, we have

$$\begin{aligned} \mathbb{P}(X < 2k) \quad &\leq \quad \mathbb{P}(\left|N'_G(u)\right| - X \ \geq \ 16k) \\ &\leq \quad \frac{\mathbb{E}[|N'_G(u)| - X]}{16k} && \text{(By Markov's inequality)} \\ &< \quad \frac{k}{16k} && (\because \mathbb{E}[X] > 18k) \\ &\leq \quad \frac{1}{16} < \frac{1}{2}. \end{aligned}$$

□

Proof of Theorem 4.3. Our query procedure will be as follows. We make $100 \log k$ samples from the sampling primitive $\mathcal{S}_{1000k}$. This results in subgraphs $G_1, \ldots, G_{100 \log k}$. Let $\hat{G} = G_1 \cup \ldots \cup G_{100 \log k}$. We find a minimum vertex cover of $\hat{G}$. We report $VC(\hat{G})$ as $VC(G)$. The correctness of the algorithm follows from Lemma 4.4. The query complexity of the algorithm is $\widetilde{\mathcal{O}}(k^2)$, which is evident from the sampling primitive described at the beginning of this section. □

Remark 1. Let $\mathsf{Match}(G)$ denote a maximum matching of graph G. Given the vertices graph G, we can find $\mathsf{Match(G)}$ with high probability by making $\widetilde{\mathcal{O}}(k^2)$ BISE queries if $|\mathsf{Match(G)}| \leq k$. Let $\hat{G} = G_1 \cup \ldots \cup G_{200 \log k}$ where each G_i is a subgraph of G resulting from a sample from $\mathcal{S}_{2000 \log k}$. Note that the number of queries we make to construct $\hat{G}$, is $\widetilde{\mathcal{O}}(k^2)$. The following is the main claim.

Lemma 4.9. *If* $|\mathsf{Match}(G)| \leq k$*, then* $\mathsf{Match}(G) = \mathsf{Match}(\hat{\mathsf{G}})$ *with high probability.*

Proof. Observe that $|VC(G)| \leq 2k$. Using arguments similar to that in the proof of Lemma 4.6, we can show that the following properties hold with high probability. $E_\ell \subseteq E(\hat{G})$ and $\mathsf{deg}_{\hat{G}}(u) \geq 5k$ for each $u \in V_h$. Note that V_h is the set of high degree vertices and E_ℓ is the edges where both the end points are low degree vertices. We can also show that $|V_h| \leq 2k$, by arguments similar Observation 4.5.

As $\hat{G}$ is a subgraph of G, a matching in $\hat{G}$ is a matching of G. If we can show that $|\mathsf{Match}(G)| \leq \left|\mathsf{Match}(\hat{G})\right|$, then as $\hat{G}$ is a subgraph of G, we are done. Consider a fixed maximum matching M in G, we show that a matching $\hat{M}$ in $\hat{G}$, can be constructed incrementally from M such that $|M| \leq \left|\hat{M}\right|$. Note that $E_\ell \subseteq E(\hat{G})$. We initialize $\hat{M}$ with all the edges in $E_\ell \cap M$. Each edge in $M \setminus E_\ell$ must have a vertex in V_h as one of its end point and no two edges in M can share a vertex. Observe that $|M \setminus E_\ell| \leq V_h$. So, $|M| \leq |E_\ell \cap M| + |V_h|$. Consider the vertices in V_h one by one. We argue that we will be able to add an edge to $\hat{M}$ corresponding to each $u \in V_h$. Using the fact that $\mathsf{deg}_{\hat{G}}(u) \geq 5k$, $|\mathsf{Match}(G)| \leq k$ and $|V_h| \leq 2k$, we can say that we have k edges incident to u such that none of them belongs to $\hat{M}$ and none of them has an end point in $V_h \setminus \{u\}$. So, we can add an edge to $\hat{M}$ corresponding to each $u \in V_h$. That is $\left|\hat{M}\right| \geq |E_\ell \cap M| + |V_h|$ But we have already shown that $|M| \leq |E_\ell \cap M| + |V_h|$. Hence, $|\mathsf{Match}(G)| \leq \left|\mathsf{Match}(\hat{G})\right|$.

Putting everything together, $\mathsf{Match}(G) = \mathsf{Match}(\hat{G})$. □

4.2 Algorithms for Vertex-Cover and Decision-Vertex-Cover

Let G be a given graph and $VC(G)$ denotes some minimum vertex cover of G. In Section 4.1, we have given an algorithm to solve Promised-Vertex-Cover with high probability using $\widetilde{\mathcal{O}}(k^2)$ BISE queries. In this section, we give algorithms for Vertex-Cover and Decision-Vertex-Cover given query access to a BISE oracle and a BIS oracle, respectively. Thus we prove Theorems 4.1 and 4.2.

Proof of Theorem 4.1. Let $\mathsf{Match}(G)$ denotes some maximum matching of graph G. By Theorem 3.1, with high probability, we can find a $\mathsf{Match}(G)$ if $|\mathsf{Match}(G)| \geq k+1$ or decide that there does not exist any matching of size at least $k+1$; by making $\widetilde{\mathcal{O}}(k^4)$ BISE queries.

If $|\mathsf{Match}(G)| \geq k+1$, then $|VC(G)| \geq k+1$. So, in this case we report that there does not exist any vertex cover of size at most k. Otherwise, if $|\mathsf{Match}(G)| \leq k$, then $|VC(G)| \leq 2k$. As $|VC(G)| \leq 2k$, $VC(G)$ can be found using our algorithm for Promised-Vertex-Cover by making $\widetilde{\mathcal{O}}(k^2)$ BISE queries. If $|VC(G)| \leq k$, we output $VC(G)$ and if $|VC(G)| > k$, we report that there does not exist a vertex cover of size at most k.

The total query complexity is $\widetilde{\mathcal{O}}(k^4)$. □

Proof of Theorem 4.2. By Observation 2.1, it is enough to give an algorithm that solves Decision-Vertex-Cover with probability at least $\frac{2}{3}$ by using $\mathcal{O}(k^8)$ BIS queries.

We choose a random hash function $h : V(G) \to [100k^4]$. Let $V_i = \{v \in V(G) : h(v) = i\}$, where $i \in [100k^4]$. Note that $\{V_1, \ldots, V_{1000k^4}\}$ form a partition of V, where some of the V_i's can be empty. For each $i \neq j$ such that $V_i, V_j \neq \emptyset$, we make a BIS query with input (V_i, V_j). Note that the output of a BIS query is Yes or No. We create a graph $\hat{G}$ where we create a vertex corresponding to each part V_i, $i \in [100k^4]$. Abusing notation, we denote $V(\hat{G}) = \{V_1, \ldots, V_{100k^4}\}$ and $E(\hat{G}) = \{(V_i, V_j) : \text{BIS oracle answers yes when given } (V_i, V_j) \text{ as input}\}$. Observe that we make $\mathcal{O}(k^8)$ queries to the BIS oracle. We find $VC(\hat{G})$ and report $|VC(G)| \leq k$ if and only if $\left|VC(\hat{G})\right| \leq k$.

For a vertex cover $VC(G)$ consider the set $S' = \{V_i \mid \exists u \in VC(G), h(u) = i\}$. Observe that S' is a vertex cover for $\hat{G}$. So, $\left|VC(\hat{G})\right| \leq |VC(G)|$, and if $|VC(G)| \leq k$, then $\left|VC(\hat{G})\right| \leq k$. Now, the correctness of our query procedure follows from the following Lemma

Lemma 4.10. *If $\left|VC(\hat{G})\right| \leq k$, then $|VC(G)| \leq k$ with probability at least $2/3$.*

Proof. Let $\mathcal{R}$ be a fixed k-representative set corresponding to G, which is a 2-uniform set system, obtained from Proposition 2.2 and let G' be the graph obtained from $\mathcal{R}$ as described in Corollary 2.3. Note that $|E(G')| \leq k(k+1)+1$ and $|V(G')| \leq 2(k(k+1)+1)$. Let us define an events $\mathcal{E}_1$ as follows.

$\mathcal{E}_1$: $h(u) = h(v)$ if and only if $u = v$, where $u, v \in V(G')$.
We lower bound the event $\mathcal{E}_1$. Let $\mathcal{E}_1^c$ be the compliment of the event $\mathcal{E}_1$. Then

$$\mathbb{P}(\mathcal{E}_1^c) \leq \sum_{u,v \in V(G')} \mathbb{P}(h(u) = h(v)) \leq \sum_{u,v \in V(G')} \frac{1}{100k^4} \leq \frac{|V(G')|^2}{100k^4} < \frac{1}{3}$$

Therefore, $\mathbb{P}(\mathcal{E}_1) \geq \frac{2}{3}$.

Now consider the property Prop: For each $(u, v) \in E(G')$, $(V_{h(u)}, V_{h(v)}) \in E(\hat{G})$. By definition of the BIS oracle, Prop holds true when $\mathcal{E}_1$ occurs. If we can show that the occurrence of Prop implies our claim, we are done.

For the rest of the proof assume that Prop holds. Let us define a function $f : V(\hat{G}) \to V(G') \cup \{\psi\}$ as follows. For each $i \in [100k^4]$, $f(V_i) = u$ if $h(u) = i$ and $u \in V(G')$, and $f(V_i) = \psi$, otherwise.

Let $\left|VC(\hat{G})\right| = k' \leq k$. Let $VC(\hat{G}) = \{X_1, \ldots, X_{k'}\} \subseteq V(\hat{G})$. Consider the vertex set $V' = \{f(X_i) \mid i \in [k'], f(X_i) \neq \psi\} \subseteq V(G')$. As $VC(\hat{G})$ is a vertex cover of $\hat{G}$, V' covers all the edges present in $E(G')$ and is of size at most k. By Corollary 2.3, $|VC(G)| \leq k$. Thus we are done. □

□

5 Algorithm for d-Hitting-Set

In this section, we prove the following results for d-Hitting-Set and d-Decision-Hitting-Set.

Theorem 5.1. *d-Hitting-Set can be solved with $\widetilde{\mathcal{O}}(k^{2d})$ GPISE queries.*

Theorem 5.2. *d-Decision-Hitting-Set can be solved with $\widetilde{\mathcal{O}}(k^{2d^2})$ GPIS queries.*

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 where 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 structure of a *sunflower* in a hypergraph. The core of a sunflower is the pairwise intersection of the hyperedges present in the sunflower, which is formally defined as follows.

Definition 5.3. 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 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 $\mathcal{H}$.

Based on the number of hyperedges forming the sunflower, the core of a sunflower can be *large*, *significant*, or *small*. We take them in such a way that each large core is significant and each significant core (and hence, large core also) must intersect with any hitting set. The formal definition follows.

Definition 5.4. Let $S_{\mathcal{H}}(C)$ denote the maximum integer t such that C is the core of a t-sunflower in $\mathcal{H}$. If $S_{\mathcal{H}}(C) > 10dk$, C is *large*. If $S_{\mathcal{H}}(C) > k$, C is *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, and

(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 observations, we can prove that the hitting set of the sampled hypergraph is the hitting set of the original graph with high probability. The formal definitions and arguments are given in Section 5.1.

The non-promised version of d-Hitting Set can be solved by using the algorithm for the promised version along with the algorithm explained for d-Packing in Section 3. If there exists a packing of size more than k, then the hitting set is also more than k. Otherwise, the hitting set is bounded by $2k$. Now we can use our algorithm for the promised version of d-Hitting Set to find an exact vertex cover from which we can give final answer to the non-promised d-Hitting Set.

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 is to use the concept of representative sets discussed in Section 2. By Proposition 2.2, 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 5.2.

5.1 Algorithm for 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 5.5. *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 which 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 $h : U(\mathcal{H}) \rightarrow [b]$ 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 $\mathcal{F}(\mathcal{H}^h) = \mathcal{F}'$.

In the rest of this section, we abuse the standard graph theoretic terminology by calling a d-uniform hypergraph a graph and a hyperedge an edge.

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 5.5. For completeness, the detailed proof of Theorem 5.5 is given at the end of Section 5.1. Let $HS(\mathcal{H})$ denote a minimum hitting set of $\mathcal{H}$.

Lemma 5.6. *If $|HS(\mathcal{H})| \leq k$, then $HS(\mathcal{H}) = HS(\hat{\mathcal{H}})$ with high probability.*

To prove Lemma 5.6, we need some intermediate results. We state the following proposition and then define some sets, which will be needed for our analysis.

Proposition 5.7 ([ER60])**.** *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 5.8. 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 subset.

The following two results (Lemma 5.9 and 5.10) give useful bounds with respect to the input instances of d-Promised-Hitting-Set.

Lemma 5.9. *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 5.7 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 5.10. *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 $|HS(\mathcal{H})| > k$.

To complete the proof, we consider the following observation when $C_{\mathcal{S}'}$ is non-empty.

Observation 5.11. 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 5.6.

Lemma 5.12. *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) $S_{\hat{\mathcal{H}}}(C) > k$, $\forall C \in \mathcal{C}'$ hold with high probability.*

Proof. First, consider the two claims stated below.

Claim 5.13. $\forall i \in [\alpha \log k]$, $\mathbb{P}(F \in \mathcal{F}(\mathcal{H}_i) \mid F \in \mathcal{F}_s) \geq \frac{1}{2}$.

Claim 5.14. $\forall i \in [\alpha \log k]$, $\mathbb{P}(S_{\mathcal{H}_i}(C) > k \mid C \in \mathcal{C}') \geq \frac{1}{2}$.

The proofs of Claims 5.13 and 5.14 are involved which we prove below.

Recall that $\hat{\mathcal{H}} = \mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{\alpha \log k}$. Using Claims 5.13 and 5.14, we get

$$\mathbb{P}(F \notin \mathcal{F}(\hat{\mathcal{H}}) \mid F \in \mathcal{F}_s) \leq \left(1 - \frac{1}{2}\right)^{\alpha \log k} \leq \frac{1}{k^\alpha}$$

and

$$\mathbb{P}(S_{\hat{\mathcal{H}}}(C) \leq k \mid C \in \mathcal{C}') \leq \left(1 - \frac{1}{2}\right)^{\alpha \log k} \leq \frac{1}{k^\alpha}$$

Using the union bound together with Lemma 5.9, we can deduce the following

$$\mathbb{P}(\mathcal{F}_s \nsubseteq \mathcal{F}(\hat{\mathcal{H}})) \leq \sum_{F \in \mathcal{F}_s} \mathbb{P}(F \notin \mathcal{F}(\hat{\mathcal{H}}) \mid F \in \mathcal{F}_s) \leq \frac{d!(10k)^d}{k^\alpha} \leq \frac{1}{k^{98}}$$

and

$$\mathbb{P}(\exists\ C \in \mathcal{C}' \text{ such that } S_{\hat{\mathcal{H}}}(C) \leq k) \leq \sum_{C \in \mathcal{C}'} \mathbb{P}(S_{\hat{\mathcal{H}}}(C) \leq k \mid C \in \mathcal{C}') \leq \frac{(d-1)!k^{d-1}}{k^\alpha} \leq \frac{1}{k^{99}}.$$

Hence,

$$\mathbb{P}(\mathcal{F}_s \not\subseteq \mathcal{F}(\hat{\mathcal{H}}) \text{ or } \exists\, C \in \mathcal{C}' \text{ such that } S_{\hat{\mathcal{H}}}(C) \leq k) \leq \frac{2}{k^{98}}.$$

This implies that with high probability, $\mathcal{F}_s \subseteq \mathcal{F}(\hat{\mathcal{H}})$ and $S_{\hat{\mathcal{H}}}(C) > k$, $\forall C \in \mathcal{C}'$ □

Proof of Claim 5.13. 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, $\mathbb{P}(F \notin \mathcal{F}(\mathcal{H}_1) \mid F \in \mathcal{F}_s) \leq \sum\limits_{u,v \in F : u \neq v} \mathbb{P}(h(u) = h(v)) + \mathbb{P}(\mathcal{E}_1)$, where $\mathcal{E}_1$ is the event defined as follows

$\mathcal{E}_1$: $\exists$ 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 5.15. Let $F \in \mathcal{F}_s$. 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 $\mathcal{F}(\mathcal{H}_c)$ $= \{F' \setminus C : F' \in \mathcal{F}(\mathcal{H}) \text{ and } F' \cap F = C\}$. 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 defined as $D = (HS(\mathcal{H}) \setminus F) \cup \bigcup\limits_{C \subset F} HS(\mathcal{H}_C)$.

If a hyperedge F^* in $\mathcal{F}(\mathcal{H}) \setminus \{F\}$ intersects 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. Since $|HS(\mathcal{H})| \leq k$ and $|HS(\mathcal{H}_C)| \leq dS_{\mathcal{H}}(C))$, we have

$$|D| \leq |HS(\mathcal{H})| + \left| \bigcup_{C \subset F} HS(\mathcal{H}_C) \right| \leq k + \sum_{C \subset F} dS_{\mathcal{H}}(C) \leq k + 2^d \cdot d \cdot 10dk \leq 2^{d+5} d^2 k.$$

The last inequality follows from the fact that F does not contain any large core. □

With respect to the set D, we define another event $\mathcal{E}_2 \supseteq \mathcal{E}_1$ and we bound $\mathbb{P}(\mathcal{E}_2)$. Let

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

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 5.14. 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 $(F \setminus C) \cap HS(\mathcal{H}) = \emptyset\ \forall F \in \mathcal{F}''$, 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 $h(y) = h(z)$.

Since $\mathbb{P}(X_F = 1) \geq \mathbb{P}(\mathcal{E}_1 \ \wedge \ \mathcal{E}_2)$, we have So,

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

The last inequality follows from the fact that $|HS(\mathcal{H})| \leq k$. Hence, $\mathbb{E}[X] = \sum_{F\in\mathcal{F}''} \mathbb{P}(X_F = 1) \geq (10d-1)k \cdot \frac{199}{200} > 9dk$.

Since $|\mathcal{F}^{''}| = (10d-1)k$, we have

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

□

Now, we have all the ingredients to prove Lemma 5.6.

Proof of Lemma 5.6. 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 5.12, 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 5.12, it is also true with high probability 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. Note that the arguments in Lemma 5.12 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 edge 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, $HS(\mathcal{H}) = HS(\hat{\mathcal{H}})$. □

Now, we give the formal proof for Theorem 5.5.

Proof of Theorem 5.5. Our query procedure will be as follows. We find $\alpha \log k$ samples using the primitive $\mathcal{S}^d_{\beta k}$, where $\alpha = 100d^2$ and $\beta = 100d^3 2^{d+5}$. Let those subgraphs be $\mathcal{H}_1, \ldots, \mathcal{H}_{\alpha \log k}$. Let $\hat{\mathcal{H}} = \mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{\alpha \log k}$. We find a minimum hitting set of $\hat{\mathcal{H}}$. We report $HS(\hat{\mathcal{H}})$ as $HS(\mathcal{H})$. The correctness of the algorithm follows from Lemma 5.6. The query complexity of the algorithm is $\widetilde{\mathcal{O}}(k^d)$, which is evident from the sampling primitive described at the beginning of this section. □

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

In this section, we finally come to the proof of Theorem 5.1. Let $\mathcal{H}$ be a given hypergraph and let $HS(\mathcal{H})$ denote a minimum hitting set of $\mathcal{H}$.

Proof of Theorem 5.1. Let $\mathsf{Pack}(\mathcal{H})$ denote a maximum packing of hypergraph $\mathcal{H}$. By Theorem 3.1, 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 $|HS(G)| \leq k$, 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 $\widetilde{\mathcal{O}}(k^{2d})$. □

Proof of Theorem 5.2. By Observation 2.1, 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.

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 $(U_{i_1}, \ldots, U_{i_d})$ 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 $\left|HS(\hat{\mathcal{H}})\right| \leq k$.

For the hitting set $HS(\mathcal{H})$, consider the set $S' = \{U_i \mid \exists u \in HS(\mathcal{H}), h(u) = i\}$. Then S' is a hitting set for $\hat{\mathcal{H}}$. So, $\left|HS(\hat{\mathcal{H}})\right| \leq |HS(\mathcal{H})|$, and if $|HS(\mathcal{H})| \leq k$, then $\left|HS(\hat{\mathcal{H}})\right| \leq k$. Now, the correctness of our query procedure follows from the following Lemma.

Lemma 5.16. *If $\left|HS(\hat{\mathcal{H}})\right| \leq k$, then $|HS(\mathcal{H})| \leq k$ with probability at least 2/3.*

Proof. Let $\mathcal{R}$ be a fixed k-representative set corresponding to $\mathcal{H}$ obtained from Proposition 2.2 and let $\mathcal{H}'$ be a set system obtained from $\mathcal{R}$ as described in Corollary 2.3. Consider the set $U(\mathcal{H}')$. Note that $|\mathcal{F}(\mathcal{H}')| \leq \binom{k+d}{d}$ and $|U(\mathcal{H}')| \leq d \cdot \binom{k+d}{d}$. Let $\mathcal{E}_1$ be the event that all the vertices in $U(\mathcal{H}')$ are uniquely colored, i.e., $\mathcal{E}_1$: $h(u) = h(v)$ if and only if $u = v$, where $u, v \in U(\mathcal{H}')$.

Now we lower bound the event $\mathcal{E}_1$. Let $\mathcal{E}_1^c$ denote the compliment of the event $\mathcal{E}_1$. Therefore,

$$\mathbb{P}(\mathcal{E}_1^c) \leq \sum_{u,v \in U(\mathcal{H}')} \mathbb{P}(h(u) = h(v)) \leq \sum_{u,v \in U(\mathcal{H}')} \frac{1}{\gamma k^{2d}} \leq \frac{|U(\mathcal{H}')|^2}{\gamma k^{2d}} < \frac{1}{3}$$

.

So, $\mathbb{P}(\mathcal{E}_1) \geq \frac{2}{3}$. Let Prop be the property that for each $F \in \mathcal{F}(\mathcal{H}')$, there is an "equivalent" hyperedge in $\mathcal{F}(\hat{\mathcal{H}})$. More specifically, Prop is the following property: For each $(u_1, \ldots, u_d) \in \mathcal{F}(\mathcal{H}')$, the hyperedge $(U_{h(u_1)}, \ldots, U_{h(u_d)})$ belongs to $\mathcal{F}(\hat{\mathcal{H}})$ for all $i \in [d]$.

From the definition of the GPIS query oracle, observe that the property Prop is true whenever the event $\mathcal{E}_1$ occurs.

If we show that the occurrence of Prop implies that $|HS(\mathcal{H})| \leq k$ if and only if $\left|HS(\hat{\mathcal{H}})\right| \leq k$, we are done.

For the rest of the proof, assume that Prop holds. Let us define a function $f : U(\hat{\mathcal{H}}) \to U(\mathcal{H}') \cup \{\psi\}$ as follows. For each $i \in [\gamma k^{2d}]$; if $h(u) = i$ and $u \in U(\mathcal{H}')$, then $f(U_i) = u$. Otherwise, $f(U_i) = \psi$.

Let $\left|HS(\hat{\mathcal{H}})\right| = k' \leq k$. Let $HS(\hat{\mathcal{H}}) = \{X_1, \ldots, X_{k'}\} \subseteq U(\hat{\mathcal{H}})$. Consider the vertex set $U' = \{f(X_i) : i \in [k'], f(X_i) \neq \psi\} \subseteq U(\mathcal{H}')$ which is of size at most k. As $HS(\hat{\mathcal{H}})$ is a hitting set of $\hat{\mathcal{H}}$, U' covers all the hyperedges present in $\mathcal{F}(\mathcal{H}')$. Hence by Corollary 2.3, $|HS(\mathcal{H})| \leq k$. □

□

6 Algorithms for Cut

In this Section, we prove the following Theorem by designing algorithms for the Cut and Decision-Cut.

Theorem 6.1. Decision-Cut *(*Cut*) can be solved using* $\widetilde{\mathcal{O}}(k^4)$ BIS *(*BISE*) queries.*

The algorithm uses the following simple idea. If the given graph has a cut of size k, then the set of vertices involved in the cut is at most $2k$. Thus, if we color the vertices of the graph by enough colors using a suitable hash function, the set of vertices involved in the cut will be colored with unique colors with high probability. Then we make BISE queries to extract the edges of a cut of size bounded by k. Finally, we describe a deterministic algorithm with slightly worse query complexity. Let $\mathsf{Cut}_t(G)$ denote a t-cut of G having the maximum number of edges.

Proof of Theorem 6.1. We show the optimization version of Cut using BISE queries. The claim for BIS query can be proved similarly.

By Observation 2.1, it is enough to give an algorithm that solves Cut with constant probability by using $\mathcal{O}(k^4)$ BISE queries. Choose a random hash function $h : V(G) \to [100k^2]$. Let $V_i = \{v \in V(G) : h(v) = i\}$ be the i-th color class, where $i \in [100k^2]$. Note that $\{V_1, \ldots, V_{100k^2}\}$ form a partition of $V(G)$ such that some of the V_i's can be empty. Let $\{V_1, \ldots, V_p\}$ be the partition of $V(G)$ such that none of the V_i's is empty, where $i \in [p]$ and $p \leq 100k^2$. For each $1 \leq i < j \leq p$ and non-empty V_i, V_j, we make a BISE query with input (V_i, V_j). Observe that we make $\mathcal{O}(k^4)$ queries to the BISE oracle. Let E' be the set of edges that are output due to the $\mathcal{O}(k^4)$ BISE queries. This results in a subgraph $\hat{G}$ of G. As $\hat{G}$ is a subgraph of G, $\left|\mathsf{Cut}_t(\hat{G})\right| \leq |\mathsf{Cut}_t(G)|$. So, if $\left|\mathsf{Cut}_t(\hat{G})\right| \geq k$, then $|\mathsf{Cut}_t(G)| \geq k$. Now consider the following Lemma.

Lemma 6.2. *If* $|\mathsf{Cut}_t(G)| \geq k$*, then* $\left|\mathsf{Cut}_t(\hat{G})\right| \geq k$ *with probability at least* $2/3$*.*

Proof. let $\mathcal{C}$ be an arbitrary set of k edges of a particular t-cut of G. Let $V_\mathcal{C}$ be the set of vertices that are incident to some edge in $\mathcal{C}$. Note that $|V_\mathcal{C}| \leq 2k$. Let $\mathcal{E}_1$ be the event that the vertices in $V_\mathcal{C}$ are uniquely colored, i.e., $\mathcal{E}_1$: $h(u) = h(v)$ if and only if $u = v$, where $u, v \in V_\mathcal{C}$. Now we lower bound the event $\mathcal{E}_1$. Let $\mathcal{E}_1^c$ be the complement of the event $\mathcal{E}_1$.

$$\mathbb{P}(\mathcal{E}_1^c) \leq \sum_{u,v \in V_\mathcal{C}} \mathbb{P}(h(u) = h(v)) \leq \sum_{u,v \in V_\mathcal{C}} \frac{1}{100k^2} \leq \frac{|V_\mathcal{C}|^2}{100k^2} < \frac{1}{3}.$$

So, $\mathbb{P}(\mathcal{E}_1) \geq \frac{2}{3}$. Let Prop be the property that for an edge (u, v) in $\mathcal{C}$, there is an "equivalent" edge (u', v') in $\hat{G}$; specifically Prop is the property where for each $(u, v) \in \mathcal{C}$, there is a $(u', v') \in E(\hat{G})$ such that $h(u) = h(u')$ and $h(v) = h(v')$. From the definition of the BISE query oracle, observe that the property Prop always holds whenever the event $\mathcal{E}_1$ occurs. If we can show that the occurrence of property Prop implies our claim, we are done.

For the rest of the proof assume that Prop holds. First, we show that there exists a p-cut $\mathcal{C}'$ of $\hat{G}$ such that $|\mathcal{C}'| \geq k$. For each $(u, v) \in \mathcal{C}$, add $(u', v') \in E(\hat{G})$ to $\mathcal{C}'$ such that $h(u) = h(u')$ and $h(v) = h(v')$. As Prop holds, observe that the set of edges in $\mathcal{C}'$, is a p-cut of $\hat{G}$. Hence, $|\mathcal{C}'| \geq |\mathcal{C}|$. Observe that if there is p-cut of a graph of size k, then there exists a $p + 1$ cut of size k. So, we are done if $t \geq p$. For $t < p$, We find all possible t-partitions of $V(\hat{G})$ such that all vertices of a particular $V_i, i \in [p]$, belong to the same part. For each t-partition of vertices, we compute the size of the t-cut, i.e., the number of edges that have endpoints in different parts. Observe that we get a t partition such that the number of edges that have endpoints in different parts is at least k. □

Hence, we report $\left|\mathsf{Cut}_t(\hat{G})\right| \geq k$ if and only if $|\mathsf{Cut}_t(G)| \geq k$. □

Using Proposition 3.3 and suitably many colors, we obtain the following deterministic algorithm.

Corollary 6.3. Decision-Cut *(*Cut*) can be solved by a deterministic algorithm with* $\mathcal{O}(k^{\mathcal{O}(1)} \log n)$ BIS *(*BISE*) queries.*

Proof. As before, let $\mathsf{Cut}_t(G)$ be a t-cut of G having the maximum number of edges and let $\mathcal{C}$ be a subset of k edges from $\mathsf{Cut}_t(G)$. Also, let $V_{\mathcal{C}}$ denote the subset of vertices that are incident to some edge of $\mathcal{C}$. Note that $|V_{\mathcal{C}}| \leq 2k$. Consider a hash function $h : V(G) \to [4k^2]$ with the property that all the vertices of $V_{\mathcal{C}}$ receive distinct colors. If we have such a hash function, then we have a deterministic algorithm for the Decision-Cut (Cut) problem with $\widetilde{\mathcal{O}}(k^4)$ BIS (BISE) queries. In other words, the only place of randomization in the above algorithm was in choosing a random hash function in the beginning. Now consider the $\mathcal{O}(k^{\mathcal{O}(1)} \log n)$-sized family of $2k$-perfect hash functions described in Proposition 3.3. For each of the $\mathcal{O}(k^{\mathcal{O}(1)} \log n)$ hash functions of the family, we run the above algorithm. Finally, we output the maximum sized cut that satisfied all the properties of the problem, over all the hash functions. By definition of this family, there is a hash function h in this family which gives distinct colors to all the vertices of $V_{\mathcal{C}}$ and therefore at least for this hash function the algorithm returns a required cut that has size at least k. Thus, the algorithm is correct. The query complexity of this algorithm is $\mathcal{O}(k^{\mathcal{O}(1)} \log n) \cdot \widetilde{\mathcal{O}}(k^4) = \mathcal{O}(k^{\mathcal{O}(1)} \log n)$. □

7 Open Problems

Deterministic algorithms for the various versions of the Hitting Set problem considered in this paper are still open. As stated before, we would also like to study other hard problems with respect to the query models used in this paper and develop other feasible query models due to which NP-hard problems have query complexity less than the complexity of the whole graph.

References

[AYZ16] N. Alon, R. Yuster, and U. Zwick. Color Coding. In *Encyclopedia of Algorithms*, pages 335–338. 2016.

[BHR+18] P. Beame, S. Har-Peled, S. N. Ramamoorthy, C. Rashtchian, and M. Sinha. Edge Estimation with Independent Set Oracles. In *Proc. of ITCS*, pages 38:1–38:21, 2018.

[BT81] B. Bollobás and C. Thomassen. The Size of Connected Hypergraphs with Prescribed Covering Number. *Journal of Combinatorial Theory, Series B*, 31(2):150 – 155, 1981.

[CCE+16] R. Chitnis, G. Cormode, H. Esfandiari, M.-T. Hajiaghayi, A. McGregor, M. Monemizadeh, and S. Vorotnikova. Kernelization via Sampling with Applications to Finding Matchings and Related Problems in Dynamic Graph Streams. In *Proc. of SODA*, pages 1326–1344, 2016.

[CFK+15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Parameterized Algorithms. *Springer*, 2015.

[ER60] P. Erdős and R. Rado. Intersection Theorems for Systems of Sets. *Journal of the London Mathematical Society*, s1-35(1):85–90, 1960.

[Fei06] U. 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.

[Gol17] O. Goldreich. Introduction to Property Testing. *Cambridge University Press*, 2017.

[GR08] O. Goldreich and D. Ron. Approximating Average Parameters of Graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.

[IMR+18] P. Indyk, S. Mahabadi, R. Rubinfeld, A. Vakilian, and A. Yodpinyanee. Set Cover in Sub-linear Time. In *Proc. of SODA*, pages 2467–2486, 2018.

[IY18] K. Iwama and Y. Yoshida. Parameterized Testability. *TOCT*, 9(4):16:1–16:16, 2018.

[ORRR12] K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A Near-Optimal Sublinear-Time Algorithm for Approximating the Minimum Vertex Cover Size. In *Proc. of SODA*, pages 1123–1131, 2012.

[RSW18] A. Rubinstein, T. Schramm, and S. M. Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In *Proc. of ITCS*, pages 39:1–39:16, 2018.