Faster Counting and Sampling Algorithms Using Colorful Decision Oracle

ANUP BHATTACHARYA, National Institute of Science Education and Research, Bhubaneswar, India
ARIJIT BISHNU, Indian Statistical Institute, Kolkata, India
ARIJIT GHOSH, Indian Statistical Institute, Kolkata, India
GOPINATH MISHRA, National University of Singapore - Kent Ridge Campus, Singapore, Singapore

In this work, we consider d-Hyperedge Estimation and d-Hyperedge Sample problems that deal with estimation and uniform sampling of hyperedges in a hypergraph $\mathcal{H}(U(\mathcal{H}), \mathcal{F}(\mathcal{H}))$ in the query complexity framework, where $U(\mathcal{H})$ denotes the set of vertices and $\mathcal{F}(\mathcal{H})$ denotes the set of hyperedges. The oracle access to the hypergraph is called Colorful Independence Oracle (CID), which takes d (non-empty) pairwise disjoint subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ as input and answers whether there exists a hyperedge in $\mathcal{H}$ having exactly one vertex in each A_i for all $i \in \{1, 2, \ldots, d\}$. Apart from the fact that d-Hyperedge Estimation and d-Hyperedge Sample problems with CID oracle access seem to be nice combinatorial problems, Dell et al. [SODA'20 & SICOMP'22] established that *decision* vs. *counting* complexities of a number of combinatorial optimization problems can be abstracted out as d-Hyperedge Estimation problem with a CID oracle access.

The main technical contribution of this article is an algorithm that estimates $m = |\mathcal{F}(\mathcal{H})|$ with $\widehat{m}$ such that

$$\frac{1}{C_d \log^{d-1} n} \leq \frac{\widehat{m}}{m} \leq C_d \log^{d-1} n$$

by using at most $C_d \log^{d+2} n$ CID queries, where n denotes the number of vertices in the hypergraph $\mathcal{H}$ and C_d is a constant that depends only on d. Our result, when coupled with the framework proposed by Dell et al. (SODA'20 & SICOMP'22), leads to implies improved bounds for $(1 \pm \varepsilon)$-approximation (where $\varepsilon \in (0, 1)$) for the following fundamental problems:

Edge Estimation using the Bipartite Independent Set (BIS) query. We improve the bound obtained by Beame et al. (ITCS'18 & TALG'20).

Triangle Estimation using the Tripartite Independent Set (TIS) query. Currently, Dell et al.'s result gives the best bound for the case of triangle estimation in general graphs (SODA'20 & SICOMP'22). The previous best bound for the case of graphs with low *co-degree* (co-degree of a graph is the maximum number of triangles incident over any edge of the graph) was due to Bhattacharya et al. (ISAAC'19 & TOCS'21). We improve both of these bounds.

Hyperedge Estimation & Sampling using Colorful Independence Oracle (CID). We give an improvement over the bounds obtained by Dell et al. (SODA'20 & SICOMP'22).

CCS Concepts: • **Theory of computation → Streaming, sublinear and near linear time algorithms;**

Additional Key Words and Phrases: Group query, hyperedge estimation, query complexity

ACM 1942-3454/2024/06-ART12
https://doi.org/10.1145/3657605

ACM Reference Format:
Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. 2024. Faster Counting and Sampling Algorithms Using Colorful Decision Oracle. *ACM Trans. Comput. Theory* 16, 2, Article 12 (June 2024), 19 pages. https://doi.org/10.1145/3657605

1 INTRODUCTION

Estimating different combinatorial structures such as edges, triangles, and cliques in an unknown graph that can be accessed only through *query oracles* is a fundamental area of research in *sublinear algorithms* [14–17]. Different query oracles provide unique ways of looking at the same graph. Beame et al. [2, 3] introduced an independent set-based subset query oracle, named Bipartite Independent Set (BIS) query, to estimate the number of edges in a graph using polylogarithmic queries. The BIS query answers a YES/NO question on the existence of an edge between two disjoint subsets of vertices of a graph G. The next natural question in this line of research were problems of estimation and uniform sampling of hyperedges in hypergraphs [4, 5, 10, 11]. In this article, we will be focusing on these two fundamental questions, and in doing so, we will improve all the previous results [2–5, 10, 11].

1.1 Our Query Oracle, Results and the Context

A hypergraph $\mathcal{H}$ is a *set system* $(U(\mathcal{H}), \mathcal{F}(\mathcal{H}))$, where $U(\mathcal{H})$ denotes a set of n vertices and $\mathcal{F}(\mathcal{H})$, a set of subsets of $U(\mathcal{H})$, denotes the set of hyperedges. A hypergraph $\mathcal{H}$ is said to be d-*uniform* if every hyperedge in $\mathcal{H}$ consists of exactly d vertices. The cardinality of the hyperedge set is denoted as $m(\mathcal{H}) = |\mathcal{F}(\mathcal{H})|$. We will access the hypergraph using the following oracle [6, 7][1]:

*Definition 1.1 (Colorful Independence Oracle (*CID*)).* Given d pairwise disjoint subsets $A_1, \ldots, A_d$ of vertices of $U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ as input, CID query answers Yes if $m(A_1, \ldots, A_d) \neq 0$, where $m(A_1, \ldots, A_d)$ denotes the number of hyperedges in $\mathcal{H}$ having exactly one vertex in each A_i, where $i \in \{1, 2, \ldots, d\}$. Otherwise, the oracle answers No.

Note that the earlier mentioned BIS is a special case of CID when $d = 2$. The CID query has a *transversal* nature to it. A *transversal* [18] of a hypergraph $\mathcal{H} = (U(\mathcal{H}), \mathcal{F}(\mathcal{H}))$ is a subset $T \subseteq U(\mathcal{H})$ that intersects all sets of $\mathcal{F}(\mathcal{H})$. One can see a CID query as a *transversal query*, as it answers if there exists a transversal for the disjoint subsets $A_1, \ldots, A_d$ as in Definition 1.1. With this query oracle access, we solve the following two problems:

d-Hyperedge-Estimation
Input: Vertex set $U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ with n vertices, a CID oracle access to $\mathcal{H}$, and $\varepsilon \in (0, 1)$.
Output: A $(1 \pm \varepsilon)$-approximation $\widehat{m}$ to $m(\mathcal{H})$ with probability $1 - 1/n^{\Omega(d)}$.

Note that Edge Estimation problem is a special case of d-Hyperedge-Estimation when $d = 2$.

[1] In References [6, 7], the oracle is named as Generalized Partite Independent Set oracle. Here, we follow the same suit as Dell et al. [10, 11] with respect to the name of the oracle.

d-Hyperedge-Sample
Input: Vertex set $U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ with n vertices, a CID oracle access to $\mathcal{H}$, and $\varepsilon \in (0,1)$.
Output: With probability $1 - 1/n^{\Omega(d)}$, report a sample from a distribution of hyperedges in $\mathcal{H}$ such that the probability that any particular hyperedge is sampled lies in the interval $[(1-\varepsilon)\frac{1}{m}, (1+\varepsilon)\frac{1}{m}]$, where m denotes the number of hyperedges in $\mathcal{H}$.

This area started with the investigation of Edge Estimation problem by Dell and Lapinskas [8, 9] and Beame et al. [2, 3]. Then, Bhattacharya et al. [4, 5] studied d-Hyperedge-Estimation for $d = 3$, and more recently Dell et al. [10, 11] gave algorithms for d-Hyperedge-Estimation and d-Hyperedge-Sample for general d. Beame et al. [2, 3] showed that Edge Estimation problem can be solved using $O(\frac{\log^{14} n}{\varepsilon^4})$ BIS queries. Having estimated the number of edges in a graph using BIS queries, a very natural question was to estimate the number of hyperedges in a hypergraph using an appropriate query oracle. This extension is nontrivial, as two edges in a graph can intersect in at most one vertex but the intersection pattern between two hyperedges in a hypergraph is more complicated. As a first step towards resolving this question, Bhattacharya et al. [4, 5] considered d-Hyperedge-Estimation in 3-uniform hypergraphs using CID queries. They showed that when *co-degree* of any pair of vertices in a 3-uniform hypergraph is bounded above by Δ, then one can solve d-Hyperedge-Estimation using $O(\frac{\Delta^2 \log^{18} n}{\varepsilon^4})$ CID queries. Recall that the co-degree of two vertices in a hypergraph is the number of hyperedges that contain both the vertices. Dell et al. [10, 11] generalized the results of Beame et al. [2, 3] and Bhattacharya et al. [4, 5] and obtained a similar result (with an improved dependency in terms of ε) for the d-Hyperedge-Estimation problem for general d. Apart from d-Hyperedge-Estimation problem, they also considered the problem of d-Hyperedge-Sample.

Proposition 1.2 (Dell et al. [10, 11]). *d-Hyperedge-Estimation and d-Hyperedge-Sample can be solved by using $O_d(\frac{\log^{4d+8} n}{\varepsilon^2})$ and $O_d(\frac{\log^{4d+12} n}{\varepsilon^2})$ CID queries, respectively.*[2]

Currently, the best-known bound (prior to this work) for solving d-Hyperedge-Estimation problem, for general d, is due to Dell et al. [10, 11], but note that for constant $\varepsilon \in (0,1)$, Beame et al. [2, 3] still have the best bound for the Edge Estimation problem.

Our main result is an improved *coarse estimation* technique, named Rough Estimation, and is stated in the following theorem. The significance of the coarse estimation technique will be discussed in Section 1.2.

Theorem 1.3 (Main Result). *There exists an algorithm* Rough Estimation *that has* CID *query access to a d-uniform hypergraph $\mathcal{H}(U, \mathcal{F})$ and returns $\widehat{m}$ as an estimate for $m = |\mathcal{F}(\mathcal{H})|$ with probability at least $1 - 1/n^{\Omega(d)}$ using at most $C_d \log^{d+2} n$ CID queries such that*

$$\frac{1}{C_d \log^{d-1} n} \leq \frac{\widehat{m}}{m} \leq C_d \log^{d-1} n,$$

where C_d is a constant that depends only on d and n denotes the number of vertices in $\mathcal{H}$.

[2]Dell et al. [10, 11] studied d-Hyperedge-Estimation and d-Hyperedge-Sample where the probability of success is $1 - \delta$ for some given $\delta \in (0,1)$ and have shown that d-Hyperedge-Estimation and d-Hyperedge-Sample can be solved by using $O_d(\frac{\log^{4d+7} n}{\varepsilon^2} \log \frac{1}{\delta})$ and $O_d(\frac{\log^{4d+11} n}{\varepsilon^2} \log \frac{1}{\delta})$ CID queries, respectively. In Proposition 1.2, we have taken $\delta = n^{O(d)}$. Note that both the results of Beame et al. [2, 3] and Bhattacharya et al. [4, 5] are in the high probability regime.

In this article, we work with success probability to be $1 - 1/n^{\Omega(d)}$ for simplicity of the presentation and compare our results with all previous results in the high probability regime.

Coarse estimation gives a crude polylogarithmic approximation for m, the number of hyperedges in $\mathcal{H}$. This improvement in the coarse estimation algorithm coupled with *importance sampling* and the algorithmic framework of Dell et al. [10, 11] gives an improved algorithm for both d-Hyperedge-Estimation and d-Hyperedge-Sample problems, stated in the following theorem:

Theorem 1.4 (Improved Bounds for Estimating and Sampling). *d-Hyperedge-Estimation and d-Hyperedge-Sample problems can be solved by using $O_d(\frac{\log^{3d+5} n}{\varepsilon^2})$ and $O_d(\frac{\log^{3d+9} n}{\varepsilon^2})$ CID queries, respectively.*

The proof of Theorem 1.3 is discussed in Section 3. The details regarding how Theorem 1.3 can be used together with the framework of Dell et al. [10, 11] to prove Theorem 1.4 is discussed in Section 5. Using Theorem 1.4, we directly get the following improved bounds for Edge Estimation and d-Hyperedge-Estimation in 3-uniform hypergraph by substituting $d = 2$ and $d = 3$, respectively:

Corollary 1.5. (a) Edge Estimation *can be solved using $O(\frac{\log^{11} n}{\varepsilon^2})$ BIS queries.*

(b) d-Hyperedge-Estimation *in a 3-uniform hypergraph can be solved using $O(\frac{\log^{14} n}{\varepsilon^2})$ CID queries.*

Addanki et al. [1] gave an algorithm for Edge Estimation that makes, ignoring lower order terms, $O(\frac{\log^6 n}{\varepsilon^5})$ BIS queries.[3] For constant $\varepsilon \in (0, 1)$, note that the algorithm of Addanki et al. [1] provides the best known bound for Edge Estimation. However, observe that if $\varepsilon = o(\frac{1}{\log^{5/3} n})$, then Corollary 1.5 (a) provides the best known bound for Edge Estimation. Recall that Bhattacharya et al. [4, 5] proved that when the co-degree of a 3-uniform graph is bounded by Δ, then d-Hyperedge-Estimation in that hypergraph can be solved using $O(\frac{\Delta^2 \log^{18} n}{\varepsilon^4})$ CID queries. For fixed $\varepsilon \in (0, 1)$ and $\Delta = o(\log n)$ the bound obtained by Bhattacharya et al. [4, 5] is asymptotically better than the bound we get from Dell et al. [10, 11], see Proposition 1.2. Corollary 1.5 (b) improves the bounds obtained by Bhattacharya et al. [4, 5] and Dell et al. [10, 11] for all values of Δ and $\varepsilon \in (0, 1)$. Table 1 gives a full comparison of our results with others.

Though there has been quite a progress on the upper bound side of d-Hyperedge-Estimation and d-Hyperedge-Sample (as summarized in Table 1), there is no work on the lower bound side until very recently by Dell et al. [12]. In particular, Dell et al. [12] showed that d-Hyperedge-Estimation requires $\Omega((\frac{\log n}{\log \log n})^{d-3})$ CID queries.

1.2 Fundamental Role of Coarse Estimation

The framework of Dell et al. [10, 11] is inspired by the following observation: Let us consider $t = O(\frac{\log n}{\varepsilon^2})$ independent subhypergraphs each induced by $n/2$ uniform random vertices. The probability that a particular hyperedge is present in a subhypergraph induced by $n/2$ uniform random vertices is $\frac{1}{2^d}$. Denoting X as the sum of the numbers of the hyperedges present in the t subhypergraphs, observe that $\frac{2^d}{t}X$ is a $(1 \pm \varepsilon)$-approximation of m. If we repeat the procedure

[3] Addanki et al. [1] solve d-Hyperedge-Estimation and d-Hyperedge-Sample for $d = 2$ with $O(\frac{\log^5 n}{\varepsilon^5} \cdot \log^5(\log n) \cdot \log(\frac{\log n}{\varepsilon}))$ and $O(\frac{\log^6 n}{\varepsilon^4} \cdot \log(\frac{\log n}{\varepsilon}) + \frac{\log^5 n}{\varepsilon^6} \cdot \log^6(\log n) \cdot \log(\frac{\log n}{\varepsilon}))$ number of BIS queries, respectively. Note that their algorithms succeed with constant probability and, unlike previous works in this area, Addanki et al.'s [1] algorithms are non-adaptive. For simplicity, ignoring the lower-order terms and assuming high probability of success, we can say that their algorithms for d-Hyperedge-Estimation and d-Hyperedge-Sample for $d = 2$ make $O(\frac{\log^6 n}{\varepsilon^5})$ and $O(\frac{\log^7 n}{\varepsilon^6})$ BIS queries, respectively.

Table 1. Our Result Vis-a-vis All Previous and Concurrent Works: We Improve All Previous Results

	Beame et al. [2, 3]	Bishnu et al. [4, 5]	Dell et al. [10, 11]	This work	Addanki et al. [1]
HEST ($d = 2$)	$O\left(\frac{\log^{14} n}{\varepsilon^4}\right)$	–	$O\left(\frac{\log^{16} n}{\varepsilon^2}\right)$	$O\left(\frac{\log^{11} n}{\varepsilon^2}\right)$	$O\left(\frac{\log^{6} n}{\varepsilon^5}\right)$
HEST ($d = 3$)	–	$O\left(\frac{\Delta^2 \log^{18} n}{\varepsilon^4}\right)$	$O\left(\frac{\log^{20} n}{\varepsilon^2}\right)$	$O\left(\frac{\log^{14} n}{\varepsilon^2}\right)$	–
HEST ($d \geq 2$)	–	–	$O_d\left(\frac{\log^{4d+8} n}{\varepsilon^2}\right)$	$O_d\left(\frac{\log^{3d+5} n}{\varepsilon^2}\right)$	–
HSAMPLE ($d \geq 2$)	–	–	$O_d\left(\frac{\log^{4d+12} n}{\varepsilon^2}\right)$	$O_d\left(\frac{\log^{3d+9} n}{\varepsilon^2}\right)$	$O\left(\frac{\log^{7} n}{\varepsilon^6}\right)$

HEST stands for d-HYPEREDGE-ESTIMATION and HSAMPLE stands for d-HYPEREDGE-SAMPLE. The bounds in the table are on the number of CID queries for different values of d. All the algorithms mentioned in this table succeed with high probability.

recursively $O(\log n)$ times, then all the subhypergraphs will have a bounded number of vertices in terms of d, at which point the number of hyperedges can be determined exactly by using $O_d(1)$ CID queries. However, the number of induced subhypergraphs in the worst case can become as *large* as $\Omega((\log n)^{\log n})$.

To have the number of subhypergraphs bounded during the algorithm, they use *importance sampling*. It is about maintaining the weighted sum of some variables whose approximate value is known to us. The output will be a bounded number of variables and some weight parameters such that the weighted sum of the variables estimates the required sum. The objective of the importance sampling procedure in Beame et al. [2, 3] and Bhattacharya et al. [4, 5] are also the same.[4] However, Dell et al. [10, 11] improved the importance sampling result by the use of a particular form of Bernstein inequality and by a very careful analysis.

To apply importance sampling, it is required to have a rough estimate (possibly with a poly-logarithmic approximation factor) of the number of hyperedges in each subhypergraph that are currently present for processing—this is exactly what coarse estimation does. The objective of coarse estimation in Beame et al. [2, 3] and Bhattacharya et al. [4, 5] are also the same.[5] But all these frameworks have a commonality. The approximation guarantee and the query complexity of the coarse estimation have a direct bearing on the query complexity of the final algorithm.

Therefore, any improvement in the coarse estimation algorithm will directly improve the query complexities of d-HYPEREDGE-ESTIMATION and d-HYPEREDGE-SAMPLE. In this article, we focus on improving the coarse estimation algorithm.

[4]In fact, Bhattacharya et al. [4, 5] directly use the importance sampling developed by Beame et al. [2, 3].
[5]Note that the main merit of the framework of Dell et al. [10, 11], over Beame et al. [2, 3] and Bhattacharya et al. [4, 5], is not only that it generalized to hypergraphs, but also the dependence on ε is $1/\varepsilon^2$ in Dell et al.'s [10, 11] work as opposed to $1/\varepsilon^4$ in Beame et al. [2, 3] and Bhattacharya et al. [4, 5].

1.3 Setup and Notations

We denote the sets $\{1, \ldots, n\}$ and $\{0, \ldots, n\}$ by $[n]$ and $[n^*]$, respectively. A hypergraph $\mathcal{H}$ is a *set system* $(U(\mathcal{H}), \mathcal{F}(\mathcal{H}))$, where $U(\mathcal{H})$ denotes the set of vertices and $\mathcal{F}(\mathcal{H})$ denotes the set of hyperedges. The set of vertices present in a hyperedge $F \in \mathcal{F}(\mathcal{H})$ is denoted by $U(F)$ or simply F. A hypergraph $\mathcal{H}$ is said to be d-uniform if all the hyperedges in $\mathcal{H}$ consist of exactly d vertices. The cardinality of the hyperedge set is $m(\mathcal{H}) = |\mathcal{F}(\mathcal{H})|$. For $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ (not necessarily pairwise disjoint), $\mathcal{F}(A_1, \ldots, A_d) \subseteq \mathcal{F}(\mathcal{H})$ denotes the set of hyperedges having a vertex in each A_i, and $m(A_1, \ldots, A_d)$ is the number of hyperedges in $|\mathcal{F}(A_1, \ldots, A_d)|$.

Let $\mathbb{E}[X]$ and $\mathbb{V}[X]$ denote the expectation and variance of the random variable X. For an event $\mathcal{E}$, the complement of $\mathcal{E}$ is denoted by $\overline{\mathcal{E}}$. The statement "a is a $(1 \pm \varepsilon)$-approximation of b" means $|b - a| \leq \varepsilon \cdot b$. For $x \in \mathbb{R}$, $\exp(x)$ denotes the standard exponential function e^x. In this article, d is a constant, and $O_d(\cdot)$ and $\Omega_d(\cdot)$ denote the standard $O(\cdot)$ and $\Omega(\cdot)$, where the constant depends only on d. We use $\log^k n$ to denote $(\log n)^k$ and by polylogarithmic, we mean $O_d((\frac{\log^d n}{\varepsilon})^{O(1)})$ in this article.

1.4 Article Organization

In Section 2, we will introduce *ordered hyperedges* and also define three other query oracles that can be simulated by using $O_d(\log n)$ CID queries. The role of ordered hyperedges and these oracles are mostly for expository purposes, that is, they help us to describe our algorithms and the calculations more neatly. The equivalence proofs of the CID oracle and its variants are discussed in Section 2. Sections 3 and 4 give the proof of our main technical result, that is, Theorem 1.3. We give the implications of our main result and how it can be used to prove Theorem 1.4 in Section 5. Some useful probability results are given in Appendix A. Since we use different types of oracles in the calculations, we have recalled all their definitions in Appendix B for the ease of reference.

2 PRELIMINARIES: CID ORACLE, ITS VARIANTS AND ORDERED HYPEREDGES

CID *oracle and its variants.* Note that the CID query takes as input d pairwise disjoint subsets of vertices. We now define two related query oracles CID_1 and CID_2 that remove the disjointness requirements for the input. Then, we extend CID_2 to the ordered setting. We show that both query oracles can be simulated, with high probability, by making $O_d(\log n)$ queries to the CID oracle.

- CID_1**:** Given s pairwise disjoint subsets of vertices $A_1, \ldots, A_s \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ and $a_1, \ldots, a_s \in [d]$ such that $\sum_{i=1}^{s} a_i = d$, CID_1 query on input $A_1^{[a_1]}, A_2^{[a_2]}, \cdots, A_s^{[a_s]}$ answers Yes if and only if $m(A_1^{[a_1]}, \ldots, A_s^{[a_s]}) \neq 0$. Here, $A^{[a]}$ denotes the set A repeated a times.
- CID_2**:** Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$, CID_2 query on input $A_1, \ldots, A_d$ answers Yes if and only if $m(A_1, \ldots, A_d) \neq 0$.

Observe that the CID query requires the input sets to be disjoint, whereas there is no such restriction for the CID_2 query. For the CID_1 query, multiple repetitions of the same set is allowed in the input. It is obvious that a CID query can be simulated by a CID_1 or a CID_2 query.

Observation 2.1 (Connection between Query Oracles for Unordered Hypergraphs). *Let $\mathcal{H}(U, \mathcal{F})$ denote a hypergraph.*

(i) *A* CID_1 *query for $\mathcal{H}(U, \mathcal{F})$ can be simulated using $O_d(\log n)$* CID *queries with probability $1 - 1/n^{\Omega(d)}$.*
(ii) *A* CID_2 *query for $\mathcal{H}(U, \mathcal{F})$ can be simulated using $O_d(1)$* CID_1 *queries.*
(iii) *A* CID_2 *query for $\mathcal{H}(U, \mathcal{F})$ can be simulated using $O_d(\log n)$* CID *queries with probability $1 - 1/n^{\Omega(d)}$.*

Proof. (i) Let the input of CID_1 query be $A_1^{[a_1]}, \ldots, A_s^{[a_s]}$ such that $a_i \in [d]\ \forall i \in [s]$ and $\sum_{i=1}^{s} a_i = d$. We partition each A_i randomly into a_i parts B_i^j for $j \in [a_i]$. We make a CID query with input $B_1^1, \ldots, B_1^{a_1}, \ldots, B_s^1, \ldots, B_s^{a_s}$. Note that

$$\mathcal{F}(B_1^1, \ldots, B_1^{a_1}, \ldots, B_s^1, \ldots, B_s^{a_s}) \subseteq \mathcal{F}\left(A_1^{[a_1]}, \ldots, A_s^{[a_s]}\right).$$

So, if CID_1 outputs "No" to query $A_1^{[a_1]}, \ldots, A_s^{[a_s]}$, then the above CID query will also report "No" as its answer. If CID_1 answers "Yes," then consider a particular hyperedge $F \in \mathcal{F}(A_1^{[a_1]}, \ldots, A_s^{[a_s]})$. Observe that

$$\begin{aligned}
&\mathbb{P}(\text{CID oracle answers "Yes"}) \\
&\geq \mathbb{P}\left(F \text{ is present in } \mathcal{F}\left(B_1^1, \ldots, B_1^{a_1}, \ldots\ldots, B_s^1, \ldots, B_s^{a_s}\right)\right) \\
&\geq \prod_{i=1}^{s} \frac{1}{a_i^{a_i}} \\
&\geq \prod_{i=1}^{s} \frac{1}{d^{a_i}} && \because a_i \leq d \text{ for all } i \in [d] \\
&= \frac{1}{d^d} && \because \sum_{i=1}^{s} a_i = d.
\end{aligned}$$

We can boost up the success probability arbitrarily by repeating the above procedure poly-logarithmic times.

(ii) Let the input to CID_2 query oracle be $A_1, \ldots, A_d$. Let us partition each set A_i into at most $2^{d-1} - 1$ subsets depending on A_i's intersection with A_j's for $j \neq i$. For $i \in [d]$, let $\mathcal{P}_i$ be the partition of A_i satisfying the following property: $\mathcal{P}_i$ contains all the maximal subsets of A_i, which are also subsets of ℓ number of A_j's for some $\ell \in [d-1]$ and $j \neq i$.
Observe that for any $i \neq j$, if we take any $B \in \mathcal{P}_i$ and $C \in \mathcal{P}_j$, then either $B = C$ or $B \cap C = \emptyset$. For each $(B_1, \ldots, B_d) \in \mathcal{P}_1 \times \ldots \times \mathcal{P}_d$, we make a CID_1 query with input $(B_1, \ldots, B_d)$. The total number of such CID_1 queries is at most $2^{O(d^2)}$, and we report "Yes" to the CID_2 query if and only if at least one CID_1 query, out of the $2^{O(d^2)}$ queries, reports "Yes".

(iii) It follows from (i) and (ii). □

Next, we discuss the meaning of an ordered hypergraph corresponding to an unordered hypergraph. Then, we consider query oracles for ordered hypergraph. Note that, in this article, the main result is on unordered hypergraphs. The introduction of ordered hypergraphs are just for simplicity of exposition of our ideas.

Ordered hyperedges. We will use the subscript "o" to denote the set of ordered hyperedges. For example, $\mathcal{H}_o(U, \mathcal{F}_o)$ denotes the ordered hypergraph corresponding to $\mathcal{H}(U, \mathcal{F})$. Here, $\mathcal{F}_o(\mathcal{H})$ denotes the set of ordered hyperedges that contains $d!$ ordered d-tuples for each hyperedge in $\mathcal{H}(U, \mathcal{F})$. Let $m_o(\mathcal{H}_o)$ denote $|\mathcal{F}_o(\mathcal{H}_o)|$ and note that $m_o(\mathcal{H}_o) = d!m(\mathcal{H})$. Also, let $\mathcal{F}_o(A_1, \ldots, A_d)$ denote the set $\{F_o \in \mathcal{F}_o(\mathcal{H}) : \text{the } i\text{th vertex of } F_o \text{ is in } A_i, \forall i \in [d]\}$. The corresponding number for ordered hyperedges is $m_o(A_1, \ldots, A_d)$ and observe that $\mathcal{F}_o(U(\mathcal{H}), \ldots, U(\mathcal{H})) = \mathcal{F}_o(\mathcal{H}_o)$.

Consider query oracles CID^o, CID_1^o, and CID_2^o for $\mathcal{H}_o$ analogous to query oracles CID, CID_1 , and CID_2 for $\mathcal{H}$, respectively.

CID^o: Given d pairwise disjoint subsets $A_1, \ldots, A_d$ of vertices of $U(\mathcal{H})$ of an ordered hypergraph $\mathcal{H}_o$ as input, CID^o query answers Yes if $m_o(A_1, \ldots, A_d) \neq 0$. Otherwise, the oracle answers No.

CID_1^o: Given s pairwise disjoint subsets of vertices $A_1, \ldots, A_s \subseteq U(\mathcal{H}_o)$ of an ordered hypergraph $\mathcal{H}_o$ and $a_1, \ldots, a_s \in [d]$ such that $\sum_{i=1}^{s} a_i = d$, CID_1^o query on input $A_1^{[a_1]}, A_2^{[a_2]}, \cdots, A_s^{[a_s]}$ answers Yes if and only if $m_o(A_1^{[a_1]}, \ldots, A_s^{[a_s]}) \neq 0$. Recall that $A^{[a]}$ denotes the set A repeated a times.

CID_2^o: Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H}_o)$ of an ordered hypergraph $\mathcal{H}_o$, CID_2^o query on input $A_1, \ldots, A_d$ answers Yes if and only if $m_o(A_1, \ldots, A_d) \neq 0$.

The following observation establishes the connection between CID^o, CID_1^o, and CID_2^o:

Observation 2.2 (Connection between Query Oracles for Ordered Hypergraph). *Let $\mathcal{H}(U, \mathcal{F})$ and $\mathcal{H}_o(U, \mathcal{F}_o)$ denote the corresponding hypergraph and its ordered variant, respectively.*

(i) *A CID_1^o query for $\mathcal{H}_o(U, \mathcal{F}_o)$ can be simulated using $O_d(\log n)$ CID^o queries with probability $1 - 1/n^{\Omega(d)}$*

(ii) *A CID_2^o query for $\mathcal{H}_o(U, \mathcal{F}_o)$ can be simulated using $O_d(1)$ CID_1^o queries.*

(iii) *A CID_2^o query for $\mathcal{H}_o(U, \mathcal{F}_o)$ can be simulated using $O_d(\log n)$ CID^o queries with probability $1 - 1/n^{\Omega(d)}$.*

Proof. The proof of the above observation is similar to the proof of Observation 2.1. □

Now consider the following observation about connecting CID^o query to $\mathcal{H}_o(U, \mathcal{F}_o)$ and CID query to $\mathcal{H}(U, \mathcal{F})$:

Observation 2.3 (CID^o Query to $\mathcal{H}_o(U, \mathcal{F}_o)$ and CID Query to $\mathcal{H}(U, \mathcal{F})$). *Let $\mathcal{H}(U, \mathcal{F})$ and $\mathcal{H}_o(U, \mathcal{F}_o)$ denote a hypergraph and its corresponding ordered hypergraph, respectively. A CID^o query to $\mathcal{H}_o(U, \mathcal{F}_o)$ can be simulated using a CID query to $\mathcal{H}(U, \mathcal{F})$.*

Proof. The proof follows from the definition of $\mathcal{H}_o(U, \mathcal{F}_o)$ as the corresponding ordered hypergraph of unordered hypergraph $\mathcal{H}(U, \mathcal{F})$, CID query oracle for $\mathcal{H}(U, \mathcal{F})$, and the definition of CID^o query oracle for $\mathcal{H}_o(U, \mathcal{F}_o)$. □

The next observation establishes the connection between CID_2^o query for $\mathcal{H}_o(U, \mathcal{F}_o)$ and CID query to $\mathcal{H}(U, \mathcal{F})$.

Observation 2.4 (CID_2^o Query to $\mathcal{H}_o(U, \mathcal{F}_o)$ and CID Query to $\mathcal{H}(U, \mathcal{F})$). *Let $\mathcal{H}(U, \mathcal{F})$ and $\mathcal{H}_o(U, \mathcal{F}_o)$ denote a hypergraph and its corresponding ordered hypergraph, respectively. A CID^o query to $\mathcal{H}_o(U, \mathcal{F}_o)$ can be simulated using $O_d(\log n)$ CID queries to $\mathcal{H}(U, \mathcal{F})$, with probability $1 - \frac{1}{n^{\Omega(d)}}$.*

Proof. The proof follows from Observation 2.2(iii) and Observation 2.3. □

The crucial takeaway from this section is that all the oracles introduced are logarithmically equivalent, i.e., an oracle can be simulated by using $O_d(\log n)$ number of queries to another oracle.

3 OVERVIEW OF THE MAIN STRUCTURAL RESULT

The main technical result of this article is Lemma 3.1 is an ordered hypergraph analogue of Theorem 1.3.

Lemma 3.1 (Main Lemma). *There exists an algorithm* Rough Estimation *that has CID_2^o query access to a d-uniform ordered hypergraph $\mathcal{H}_o(U, \mathcal{F}_o)$ corresponding to hypergraph $\mathcal{H}(U, \mathcal{F})$ and returns $\widehat{m}_o$ as an estimate for $m_o = |\mathcal{F}_o(\mathcal{H}_o)|$ such that*

$$\frac{1}{C_d \log^{d-1} n} \leq \frac{\widehat{m}_o}{m_o} \leq C_d \log^{d-1} n$$

with probability at least $1 - 1/n^{\Omega(d)}$ using at most $C_d \log^{d+1} n$ CID_2^o queries, where C_d is a constant that depends only on d.

Why Lemma 3.1 implies Theorem 1.3? The algorithm, say, $\mathcal{A}$, of Theorem 1.3 is on an unordered hypergraph $\mathcal{H}(U, \mathcal{F})$ with CID query access. $\mathcal{A}$ just simulates the algorithm $\mathcal{A}'$ of Lemma 3.1 on the corresponding ordered hypergraph $\mathcal{H}_o(U, \mathcal{F}_o)$ using CID_2^o queries. The output of $\mathcal{A}'$ is also a desired approximation stated in Theorem 1.3 as $m_o(\mathcal{H}_o) = d!m(\mathcal{H})$. The query complexity of $\mathcal{A}$ is $O_d(\log^{d+2} n)$ as $\mathcal{A}'$ uses $O_d(\log^{d+1} n)$ CID_2^o queries (see Lemma 3.1), and each CID_2^o query can be simulated by using $O_d(\log n)$ CID queries with high probability (see Observation 2.4).

In the rest of the section, we discuss the overview of the proof of Lemma 3.1. At a high level, the idea for an improved coarse estimation (Lemma 3.1) involves a recursive bucketing technique of vertices and a careful analysis of the intersection pattern of hyperedges.

Notice that $U(\mathcal{H}) = U(\mathcal{H}_o)$, i.e., the set of vertices for the hypergraph and the ordered hypergraph are the same. So, w.l.o.g. we can use both notations. To build up towards the proof of Lemma 3.1, we first need to define some quantities and prove Claim 3.2. For that, let us think of partitioning the vertex set in $U_1 = U(\mathcal{H})$ into buckets such that for any two vertices u and v in the same bucket B, $|\mathcal{F}_o(u)| \approx |\mathcal{F}_o(v)|$, where $\mathcal{F}_o(x)$ denotes the set of ordered hyperedge with x as the first vertex. It can be shown that there is a bucket of vertices $Z_1 \subseteq U_1$ such that its (subset of) vertices serve as the first vertex of at least $\frac{m_o}{d \log n+1}$ hyperedges. For each vertex $z_1 \in Z_1$, let the number of hyperedges in $\mathcal{H}_o$ having z_1 as the first vertex, lie between 2^{q_1} and $2^{q_1+1} - 1$ for some suitable q_1. Then, we can argue that the number of vertices in bucket Z_1 is given by

$$|Z_1| \geq \frac{m_o}{2^{q_1+1}(d \log n + 1)}.$$

Similarly, we extend the bucketing idea to tuples as follows: Consider a vertex a_1 in a particular bucket of U_1 and $\mathcal{F}_o(a_1)$, the set of all ordered hyperedges containing a_1 as the first vertex. We can bucket the vertices in $U_2 = U(\mathcal{H})$ such that the vertices in each bucket of U_2 are present as the second vertex in approximately the same number of hyperedges in $\mathcal{F}_o(a_1)$. We generalize the above bucketing strategy with the vertices in U_i's, which is formally described below. Notice that this way of bucketing will allow us to use conditionals on sampling vertices from the desired buckets of U_i's.

For $q_1 \in [(d \log n)^*] = \{0, 1, \ldots, d \log n\}$, let $U_1(q_1) \subseteq U_1$ be the set of vertices such that for each $a_1 \in U_1(q_1)$, the number of hyperedges in $\mathcal{F}_o(\mathcal{H}_o)$, containing a_1 as the first vertex, lies between 2^{q_1} and $2^{q_1+1} - 1$. For $2 \leq i \leq d-1$, and $q_j \in [(d \log n)^*]$ for each $j \in [i-1]$, consider $a_1 \in U_1(q_1), a_2 \in U_2((q_1, a_1), q_2), \ldots, a_{i-1} \in U_{i-1}((q_1, a_1), \ldots, (q_{i-2}, a_{i-2}), q_{i-1})$. Let $U_i((q_1, a_1), \ldots, (q_{i-1}, a_{i-1}), q_i)$ be the set of vertices in U_i such that for each $u_i \in U_i((q_1, u_1), \ldots, (q_{i-1}, a_{i-1}), q_i)$, the number of ordered hyperedges in $\mathcal{F}_o(\mathcal{H}_o)$, containing u_j as the jth vertex for all $j \in [i]$, lies between 2^{q_i} and $2^{q_i+1} - 1$. We need the following result to proceed further: For ease of presentation, we use (Q_i, A_i) to denote $(q_1, a_1), \ldots, (q_{i-1}, a_{i-1})$ for $2 \leq i \leq d-1$. Informally, Claim 3.2 says that for each $i \in [d-1]$, there exists a bucket in U_i having a *large* number of vertices contributing approximately the same number of hyperedges.

Claim 3.2. (i) *There exists* $q_1 \in [(d \log n)^*] = \{0, 1, \ldots, d \log n\}$ *such that*

$$|U_1(q_1)| > \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d \log n + 1)}.$$

(ii) *Let* $2 \leq i \leq d-1$ *and* $q_j \in [(d \log n)^*]$ $\forall j \in [i-1]$. *Let* $a_1 \in U_1(q_1)$, $a_j \in U_j((Q_{j-1}, A_{j-1}), q_j)$ $\forall j \neq 1$ *and* $j < i$. *There exists* $q_i \in [(d \log n)^*]$ *such that*

$$|U_i((Q_i, A_i), q_i)| > \frac{2^{q_{i-1}}}{2^{q_i+1}(d \log n + 1)}.$$

Proof. (i) Observe that

$$m_o(\mathcal{H}_o) = \sum_{q_1=0}^{d\log n} m_o(U_1(q_1), U_2, \ldots, U_d).$$

So, there exists $q_1 \in [(d\log n)^*]$ such that $m_o(U_1(q_1), U_2, \ldots, U_d) \geq \frac{m_o(\mathcal{H}_o)}{d\log n+1}$. From the definition of $U_1(q_1)$, we have

$$m_o(U_1(q_1), U_2, \ldots, U_d) < |U_1(q_1)| \cdot 2^{q_1+1}.$$

Hence, there exists $q_1 \in [(d\log n)^*]$ such that

$$|U_1(q_1)| > \frac{m_o(U_1(q_1), U_2, \ldots, U_d)}{2^{q_1+1}} \geq \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d\log n+1)}.$$

(ii) Note that

$$m_o(\{a_1\}, \ldots, \{a_{i-1}\}, U_i, \ldots, U_d) = \sum_{q_i=0}^{d\log n} m_o(\{a_1\}, \ldots, \{a_{i-1}\}, U_i((Q_{i-1}, A_{i-1}), q_i), \ldots, U_d).$$

So, there exists $q_i \in [(d\log n)^*]$ such that

$$m_o(\{a_1\}, \ldots, \{a_{i-1}\}, U_i((Q_{i-1}, A_{i-1}), q_i), \ldots, U_d) \geq \frac{m_o(\{a_1\}, \ldots, \{a_{i-1}\}, U_i, \ldots, U_d)}{d\log n+1}.$$

From the definition of $U_i((Q_{i-1}, A_{i-1}), q_i)$, we have

$$m_o(\{a_1\}, \ldots, \{a_{i-1}\}, U_i((Q_{i-1}, A_{i-1}), q_i), \ldots, U_d) < |U_i((Q_{i-1}, A_i), q_i)| \cdot 2^{q_i+1}.$$

Hence, there exists $q_i \in [(d\log n)^*]$ such that

$$\begin{aligned} |U_i((Q_{i-1}, A_i), q_i)| &> \frac{m_o(\{a_1\}, \ldots, \{a_{i-1}\}, U_i((Q_{i-1}, A_{i-1}), q_i), \ldots, U_d\})}{2^{q_i+1}} \\ &\geq \frac{m_o(\{a_1\}, \ldots, \{a_{i-1}\}, U_i, \ldots, U_d\})}{2^{q_i+1}(d\log n+1)} \\ &\geq \frac{2^{q_{i-1}}}{2^{q_i+1}(d\log n+1)}. \end{aligned}$$

□

From Claim 3.2, it follows that there exists $(q_1, \ldots, q_{d-1}) \in [(d\log n)^*]^{d-1}$ such that

$$|U_1(q_1)| > \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d\log n+1)}$$

and

$$|U_i((Q_i, A_i), q_i)| > \frac{2^{q_{i-1}}}{2^{q_i+1}(d\log n+1)}.$$

So, if we sample each vertex in U_1 with probability $p_1 = \min\{\frac{2^{q_1}}{m_o}, 1\}$ independently to generate B_1, each vertex of U_i $(2 \leq i \leq d-1)$ with probability $p_i = \min\{2^{q_i - j_{i-1}} \cdot d\log n, 1\}$ independently to generate B_i, and each vertex in U_d with probability $\min\{2^{-q_{d-1}}, 1\}$ to generate B_d, then we can show that $\mathcal{F}_o(B_1, \ldots, B_d)$ is nonempty with probability at least $\prod_{i=1}^{d} p_i \geq \frac{1}{2^d}$. The success probability $\frac{1}{2^d}$ can be amplified by repeating the procedure suitable number of times. So, if we consider all possible $O_d(\log^{d-1} n)$ guesses for $(q_1, \ldots, q_{d-1})$, then we can show that there exists a guess for which $\mathcal{F}_o(B_1, \ldots, B_d)$ is nonempty, that is, $m_o(B_1, \ldots, B_d) \neq 0$, and that can be determined by a CID_2^o query. In total, there will be $O_d(\log^{d-1} n)$ CID_2^o queries.

However, the sampling probability p_1 to sample the vertices from U_1 depends on m_o. But, we do not know m_o. Observe that the above procedure works even if we know any lower bound on m_o. So, the idea is to consider geometrically decreasing guesses for m_o starting from $m_o = n^d$, and call the above procedure for the guesses until the CID_2^o query corresponding to the guess $\widehat{\mathcal{R}}$ for m_o reports that $m_o(B_1, \ldots, B_d) \neq 0$. Then, we report the final output by scaling the current $\widehat{\mathcal{R}}$ suitably.

Observe that, if we stop for a guess $\widehat{\mathcal{R}}$ that is within a suitable polylogarithmic factor, then we report a desired output. To ensure that the algorithm reports a desired output with a good probability, the only case that needs to be taken care of is the case when the guess is at least a suitable polylogarithmic factor more than the correct m_o. We will show that, in any of these cases, the corresponding CID_2^o query reports $m_o(B_1, \ldots, B_d) \neq 0$ with probability at most $O(\frac{1}{2^d})$. The success probability of $1 - \Omega(\frac{1}{2^d})$ can be amplified by repeating the procedure suitable number of times for each guess. In the next section, we will give the full proof of Lemma 3.1.

4 PROOF OF LEMMA 3.1

We now prove Lemma 3.1 formally. The algorithm corresponding to Lemma 3.1 is Algorithm 2 (named Rough Estimation). Algorithm 1 (named Verify-Estimate) is a subroutine of Algorithm 2. Algorithm 2 calls Verify-Estimate in a loop for different guesses of $\widehat{R}$. Algorithm 1 determines whether a given estimate $\widehat{R}$ of the number of ordered hyperedges is correct up to $O_d(\log^{2d-3} n)$ factor. Note that Lemmas 4.1 and 4.2 are intermediate results needed to prove Lemma 3.1 and they bound the probability, from above and below, of Verify-Estimate accepting the estimate $\widehat{\mathcal{R}}$.

4.1 Upper Bound on the Probability of Accepting the Estimate $\widehat{R}$

We prove the following lemma:

Lemma 4.1. *If* $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d\, m_o(\mathcal{H}_o) \log^{2d-3} n$, *then*

$$\mathbb{P}(\text{Verify-Estimate } (\mathcal{H}_o, \widehat{\mathcal{R}}) \text{ accepts the estimate } \widehat{R}) \leq \frac{1}{20 \cdot 2^d}.$$

Proof. Consider the set of ordered hyperedges $\mathcal{F}_o(\mathcal{H}_o)$ in $\mathcal{H}_o$. Algorithm Verify-Estimate taking parameters $\mathcal{H}_o$ and $\widehat{\mathcal{R}}$ and described in Algorithm 1, loops over all possible $\mathbf{j} = (j_1, \ldots, j_{d-1}) \in [(d \log n)^*]^{d-1}$.[6] For each $\mathbf{j} = (j_1, \ldots, j_{d-1}) \in [(d \log n)^*]^{d-1}$, Verify-Estimate $(\mathcal{H}_o, \widehat{\mathcal{R}})$ samples vertices in each U_i with *suitable* probability values $p_{i,\mathbf{j}}$, depending on $\mathbf{j}$, $\widehat{R}$, d, and $\log n$, to generate the sets $B_{i,\mathbf{j}}$ for $1 \leq i \leq d$. See Algorithm 1 for the exact values of $p_{i,\mathbf{j}}$'s.[7] Verify-Estimate $(\mathcal{H}_o, \widehat{\mathcal{R}})$ reports Accept if there exists one $\mathbf{j} \in [(d \log n)^*]^{d-1}$ such that $m_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}}) \neq 0$. Otherwise, Verify-Estimate $(\mathcal{H}_o, \widehat{\mathcal{R}})$ reports Reject.

For an ordered hyperedge $F_o \in \mathcal{F}_o(\mathcal{H}_o) = \mathcal{F}_o(U_1, \ldots, U_d)$ and $\mathbf{j} \in [(d \log n)^*]^{d-1}$. Note that

$$U_1 = \cdots = U_d = U(\mathcal{H}).$$

Let $X^{\mathbf{j}}_{F_o}$ denote the indicator random variable such that $X^{\mathbf{j}}_{F_o} = 1$ if and only if $F_o \in \mathcal{F}_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}})$. Let

$$X_{\mathbf{j}} = \sum_{F_o \in \mathcal{F}_o(\mathcal{H}_o)} X^{\mathbf{j}}_{F_o}.$$

[6] Recall that $[n^*]$ denotes the set $\{0, \ldots, n\}$.

[7] The buckets were indexed as $B_1, B_2, \ldots, B_d$ in Section 3. Here, the same buckets are indexed as $B_{i,\mathbf{j}}$'s to indicate the loop indices where the buckets have been generated. Same with the $p_{i,\mathbf{j}}$'s.

ALGORITHM 1: Verify-Estimate $(\mathcal{H}_o, \widehat{\mathcal{R}})$.

Input: CID query access to a d-uniform hypergraph $\mathcal{H}_o(U, \mathcal{F})$ and a guess $\widehat{R}$ for the number of hyperedges in $\mathcal{H}_o$.
Output: Accept $\widehat{\mathcal{R}}$ or Reject $\widehat{\mathcal{R}}$.

Let $U_1 = \ldots = U_d = U(\mathcal{H})$
for $(j_1 = d \log n \text{ to } 0)$ **do**
  Find $B_1 \subseteq U_1$ by sampling every element of U_1 with probability $p_1 = \min\{\frac{2^{j_1}}{\widehat{\mathcal{R}}}, 1\}$ independently of other elements.
  for $(j_2 = d \log n \text{ to } 0)$ **do**
    Find $B_2 \subseteq U_2$ by sampling every element of U_2 with probability $p_2 = \min\{2^{j_2 - j_1} \cdot d \log n, 1\}$ independently of other elements.
    ⋮⋮
    for $(j_{d-1} = d \log n \text{ to } 0)$ **do**
      Find $B_{d-1} \subseteq U_{d-1}$ by sampling every element of U_{d-1} with probability $p_{d-1} = \min\{2^{j_{d-1} - j_{d-2}} \cdot d \log n, 1\}$ independently of other elements.
      Let $\mathbf{j} = (j_1, \ldots, j_{d-1}) \in [(d \log n)^*]^{d-1}$
      /* The vector $\mathbf{j}$ denotes the values of the loop variables $j_1, \ldots, j_{d-1}$ */
      Let $p_{i,\mathbf{j}} = p_i$, where $1 \le i \le d-1$
      Let $B_{i,\mathbf{j}} = B_i$, where $1 \le i \le d-1$
      /* $B_{i,\mathbf{j}}$'s and $p_{i,\mathbf{j}}$'s indicate the sequence of buckets and probabilities formed during the execution of the nested loops; the vector $\mathbf{j}$ indicates the current status of the loop. */
      Find $B_{d,\mathbf{j}} = B_d \subseteq U_d$ by sampling every element of U_d with probability $p_d = \min\{2^{-j_{d-1}}, 1\}$ independently of other elements.
      if $(m_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}}) \neq 0)$ **then**
        Accept /*[Note that CID_2^o query is called in the above line.]*/
      end
    end
  end
end
Reject

Note that $m_o(B_{1,\mathbf{j}}, \ldots, B_{d,\mathbf{j}}) = X_{\mathbf{j}}$. We have,

$$\begin{aligned}\mathbb{P}\left(X^{\mathbf{j}}_{F_o} = 1\right) &= \prod_{i=1}^{d} p_{i,\mathbf{j}} \\ &\le \frac{2^{j_1}}{\widehat{\mathcal{R}}} \times \frac{2^{j_2}}{2^{j_1}} d \log n \times \cdots \times \frac{2^{j_{d-1}}}{2^{j_{d-2}}} d \log n \times \frac{1}{2^{j_{d-1}}} \\ &= \frac{d^{d-2} \log^{d-2} n}{\widehat{\mathcal{R}}}.\end{aligned}$$

Then,

$$\mathbb{E}\left[X_{\mathbf{j}}\right] \le \frac{m_o(\mathcal{H}_o)}{\widehat{\mathcal{R}}} d^{d-2} \log^{d-2} n,$$

and, since $X_{\mathbf{j}} \ge 0$, we have

$$\mathbb{P}\left(X_{\mathbf{j}} \neq 0\right) = \mathbb{P}(X_{\mathbf{j}} \ge 1) \le \mathbb{E}\left[X_{\mathbf{j}}\right] \le \frac{m_o(\mathcal{H}_o)}{\widehat{\mathcal{R}}} d^{d-2} \log^{d-2} n.$$

Now, using the fact that $\widehat{\mathcal{R}} \geq 20d^{2d-3} \cdot 4^d \cdot m_o(\mathcal{H}_o) \log^{2d-3} n$, we have

$$\mathbb{P}\left(X_{\mathbf{j}} \neq 0\right) \leq \frac{1}{20d^{d-1} \cdot 4^d \cdot \log^{d-1} n}.$$

Recall that Verify-Estimate accepts if and only if there exists $\mathbf{j}$ such that $X_{\mathbf{j}} \neq 0$.[8] Using the union bound, we get

$$\begin{aligned}\mathbb{P}\left(\text{Verify-Estimate } (\mathcal{H}_o, \widehat{\mathcal{R}}) \text{ accepts the estimate } \widehat{R}\right) &\leq \sum_{\mathbf{j} \in [(d \log n)^*]^{d-1}} \mathbb{P}(X_{\mathbf{j}} \neq 0) \\ &\leq \frac{(d \log n + 1)^{d-1}}{20 \cdot 4^d \cdot (d \log n)^{d-1}} \\ &\leq \frac{1}{20 \cdot 2^d}.\end{aligned}$$

□

4.2 Lower Bound on the Probability of Accepting the Estimate $\widehat{R}$

We prove the following:

Lemma 4.2. *If* $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$, *then* $\mathbb{P}$(Verify-Estimate $(\mathcal{H}_o, \widehat{\mathcal{R}})$ *accepts the estimate* $\widehat{R}$) $\geq \frac{1}{2^d}$.

Proof. We will be done by showing the following: Verify-Estimate accepts with probability at least $1/2^d$ when the loop variables $j_1, \ldots, j_{d-1}$, respectively, attain values $q_1, \ldots, q_{d-1}$ such that

$$|U_1(q_1)| > \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d \log n + 1)}$$

and

$$|U_i((Q_i, A_i), q_i)| > \frac{2^{q_{i-1}}}{2^{q_i+1}(d \log n + 1)}$$

for all $i \in [d-1] \setminus \{1\}$. The existence of such j_is is evident from Claim 3.2. Let $\mathbf{q} = (q_1, \ldots, q_{d-1})$. Recall that $B_{i,\mathbf{q}} \subseteq U_i$ is the sample obtained when the loop variables $j_1, \ldots, j_{d-1}$ attain values $q_1, \ldots, q_{d-1}$, respectively. Let $\mathcal{E}_i, i \in [d-1]$, be the events defined as follows:

- $\mathcal{E}_1$: $U_1(q_1) \cap B_{1,\mathbf{q}} \neq \emptyset$.
- $\mathcal{E}_i$: $U_j((Q_{j-1}, A_{j-1}), q_j) \cap B_{j,\mathbf{q}} \neq \emptyset$, where $2 \leq i \leq d-1$.

As noted earlier, Claim 3.2 says that for each $i \in [d-1]$, there exists a bucket in U_i having a *large* number of vertices contributing approximately the same number of hyperedges. The above events correspond to the nonempty intersection of vertices in heavy buckets corresponding to U_i and the sampled vertices $B_{i,\mathbf{j}}$, where $i \in [d-1]$. Observe that

$$\begin{aligned}\mathbb{P}(\overline{\mathcal{E}_1}) \leq \left(1 - \frac{2^{q_1}}{\widehat{\mathcal{R}}}\right)^{|U_1(q_1)|} &\leq \exp\left(-\frac{2^{q_1}}{\widehat{\mathcal{R}}} |U_1(q_1)|\right) \\ &\leq \exp\left(-\frac{2^{q_1}}{\widehat{\mathcal{R}}} \cdot \frac{m_o(\mathcal{H}_o)}{2^{q_1+1}(d \log n + 1)}\right) \\ &\leq \exp(-1).\end{aligned}$$

The last inequality uses the fact that $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$, which is the condition of the lemma. Assume that $\mathcal{E}_1$ occurs and $a_1 \in U_1(q_1) \cap B_{1,\mathbf{q}}$. The particular way of constructing the buckets $B_{i,\mathbf{q}}$ by drawing

[8] Note that $\mathbf{j}$ is a vector but $X_{\mathbf{j}}$ is a scalar.

samples from U_i's will allow us now to get into conditionals. We will bound the probability that $U_2(Q_1, A_1), q_2) \cap B_{2,\mathbf{q}} = \emptyset$, that is, $\overline{\mathcal{E}_2}$. Note that, by Claim 3.2 (ii),

$$|U_2(Q_1, A_1), q_2)| \geq \frac{2^{q_1}}{2^{q_2+1}(d \log n + 1)}.$$

So,

$$\mathbb{P}\left(\overline{\mathcal{E}_2} \mid \mathcal{E}_1\right) \leq \left(1 - \frac{2^{q_2}}{2^{q_1}} \log n\right)^{|U_2(Q_1, A_1), q_2)|} \leq \exp(-1).$$

Assume that $\mathcal{E}_1, \ldots, \mathcal{E}_{i-1}$ hold, where $3 \leq i \in [d-1]$. Let $a_1 \in U_1(q_1)$ and $a_{i-1} \in A_{i-1}((Q_{i-2}, U_{i-2}), q_{i-1})$. We will bound the probability that $U_i((Q_{i-1}, A_{i-1}), q_i) \cap B_{i,\mathbf{q}} = \emptyset$, that is, $\overline{\mathcal{E}_i}$. Note that

$$|U_i((Q_{i-1}, A_{i-1}), q_i)| \geq \frac{2^{q_{i-1}}}{2^{q_i+1}(d \log n + 1)}.$$

So, for $3 \leq i \in [d-1]$, we have

$$\mathbb{P}\left(\overline{\mathcal{E}_i} \mid \mathcal{E}_1, \ldots, \mathcal{E}_{i-1}\right) \leq \left(1 - \frac{2^{q_i}}{2^{q_{i-1}}} \log n\right)^{|U_i(Q_{i-1}, A_{i-1}), q_i)|} \leq \exp(-1).$$

Assume that $\mathcal{E}_1, \ldots, \mathcal{E}_{d-1}$ hold. Let $a_1 \in U_1(q_1)$ and $a_{i-1} \in U_{i-1}((Q_{i-2}, U_{i-2}), q_{i-1})$ for all $i \in [d-1] \setminus \{1\}$. Let $S \subseteq U_d$ be the set of dth vertex of the ordered hyperedges in $\mathcal{F}_o(\mathcal{H}_o)$ having u_j as the jth vertex for all $j \in [d-1]$. Note that $|S| \geq 2^{q_{d-1}}$. Let $\mathcal{E}_d$ represent the event $S \cap B_{d,\mathbf{q}} \neq \emptyset$. Therefore,

$$\mathbb{P}(\overline{\mathcal{E}_d} \mid \mathcal{E}_1, \ldots, \mathcal{E}_{d-1}) \leq \left(1 - \frac{1}{2^{q_{d-1}}}\right)^{q_{d-1}} \leq \exp(-1).$$

Observe that Verify-Estimate accepts if $m(B_{1,\mathbf{q}}, \ldots, B_{d,\mathbf{q}}) \neq 0$. Also,

$$m_o(B_{1,\mathbf{q}}, \ldots, B_{d,\mathbf{q}}) \neq 0 \text{ if } \bigcap_{i=1}^{d} \mathcal{E}_i \text{ occurs.}$$

Hence,

$$\begin{aligned}
\mathbb{P}(\textsc{Verify-Estimate}\ (\mathcal{H}_o, \widehat{\mathcal{R}}) \text{ accepts}) &\geq \mathbb{P}\left(\bigcap_{i=1}^{d} \mathcal{E}_i\right) \\
&= \mathbb{P}(\mathcal{E}_1) \prod_{i=2}^{d} \mathbb{P}\left(\mathcal{E}_i \,\middle|\, \bigcap_{j=1}^{i-1} \mathcal{E}_j\right) \\
&> \left(1 - \frac{1}{e}\right)^d > \frac{1}{2^d}.
\end{aligned}$$

□

4.3 The Proof of Lemma 3.1 Using the Probability Bounds

Now, we will prove Lemma 3.1, which will be based on Algorithm 2.

Proof of Lemma 3.1. Note that an execution of Rough Estimation for a particular $\widehat{\mathcal{R}}$ repeats Verify-Estimate for $\Gamma = d \cdot 4^d \cdot 2,000 \log n$ times and gives output $\widehat{\mathcal{R}}$ if more than $\frac{\Gamma}{10 \cdot 2^d}$ Verify-Estimate accepts. For a particular $\widehat{\mathcal{R}}$, let X_i be the indicator random variable such that $X_i = 1$ if and only if the ith execution of Verify-Estimate accepts, where $i = 1, \ldots, \Gamma$. Also, let $X = \sum_{i=1}^{\Gamma} X_i$. Rough Estimation gives output $\widehat{\mathcal{R}}$ if $X > \frac{\Gamma}{10 \cdot 2^d}$.

Consider the execution of Rough Estimation for a particular $\widehat{\mathcal{R}}$. If $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d \cdot m_o(\mathcal{H}_o) \cdot \log^{2d-3} n$, then we first show that Rough Estimation does not accept with high probability. Recall

ALGORITHM 2: ROUGH ESTIMATION($\mathcal{H}_o(U, \mathcal{F}_o)$).

Input: CID_2^o query access to a d-uniform hypergraph $\mathcal{H}_o(U, \mathcal{F}_o)$.
Output: An estimate $\widehat{m}_o$ for $m_o = m_o(\mathcal{H}_o)$.
1 **for** ($\widehat{\mathcal{R}} = n^d, n^d/2, \ldots, 1$) **do**
2 Repeat VERIFY-ESTIMATE ($\mathcal{H}_o, \widehat{\mathcal{R}}$) for $\Gamma = d \cdot 4^d \cdot 2,000 \log n$ times. If more than $\frac{\Gamma}{10 \cdot 2^d}$ VERIFY-ESTIMATE accepts, then output $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d \cdot (\log n)^{d-2}}$.
3 **end**

Lemma 4.1. If $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d \cdot m_o(\mathcal{H}_o) \log^{2d-3} n$, then $\mathbb{P}(X_i = 1) \leq \frac{1}{20 \cdot 2^d}$, and hence $\mathbb{E}[X] \leq \frac{\Gamma}{20 \cdot 2^d}$. By using Chernoff-Hoeffding's inequality (see Lemma A.2 (i) in Section A), we get

$$\mathbb{P}\left(X > \frac{\Gamma}{10 \cdot 2^d}\right) = \mathbb{P}\left(X > \frac{\Gamma}{20 \cdot 2^d} + \frac{\Gamma}{20 \cdot 2^d}\right) \leq \frac{1}{n^{10d}}.$$

Using the union bound for all $\widehat{\mathcal{R}}$, the probability that ROUGH ESTIMATION outputs some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d}$ such that $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d \cdot m_o(\mathcal{H}_o) \log^{2d-3} n$, is at most $\frac{d \log n}{n^{10}}$. Now consider the instance when the for loop in the algorithm ROUGH ESTIMATION executes for a $\widehat{\mathcal{R}}$ such that $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$. In this situation, $\mathbb{P}(X_i = 1) \geq \frac{1}{2^d}$. So, $\mathbb{E}[X] \geq \frac{\Gamma}{2^d}$. By using Chernoff-Hoeffding's inequality (see Lemma A.2 (ii) in Section A),

$$\mathbb{P}\left(X \leq \frac{\Gamma}{10 \cdot 2^d}\right) \leq \mathbb{P}\left(X < \frac{\Gamma}{2^d} - \frac{4}{5} \cdot \frac{\Gamma}{2^d}\right) \leq \frac{1}{n^{100d}}.$$

By using the union bound for all $\widehat{\mathcal{R}}$, the probability that ROUGH ESTIMATION outputs some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d}$ such that $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$, is at most $\frac{d \log n}{n^{100d}}$. Observe that the probability that ROUGH ESTIMATION outputs some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d}$ such that $\widehat{\mathcal{R}} \geq 20d^{2d-3}4^d m_o(\mathcal{H}_o) \log^{2d-3} n$ or $\widehat{\mathcal{R}} \leq \frac{m_o(\mathcal{H}_o)}{4d \log n}$, is at most

$$\frac{d \log n}{n^{10d}} + \frac{d \log n}{n^{100d}} \leq \frac{1}{n^{8d}}.$$

Putting everything together, ROUGH ESTIMATION gives some $\widehat{m}_o = \frac{\widehat{\mathcal{R}}}{d^{d-2} \cdot 2^d \cdot (\log n)^{d-2}}$ as the output with probability at least $1 - \frac{1}{n^{8d}}$ satisfying

$$\frac{m_o(\mathcal{H}_o)}{8d^{d-1}2^d \log^{d-1} n} \leq \widehat{m}_o \leq 20d^{d-1}2^d \cdot m_o(\mathcal{H}_o) \log^{d-1} n.$$

From the pseudocode of VERIFY-ESTIMATE (Algorithm 1), we call for CID_2 queries only at line number 12. In the worst case, VERIFY-ESTIMATE executes line number 12 for each $\mathbf{j} \in [(d \log n)^*]$. That is, the query complexity of VERIFY-ESTIMATE is $O(\log^{d-1} n)$. From the description of ROUGH ESTIMATION, we can see that ROUGH ESTIMATION calls VERIFY-ESTIMATE $O_d(\log n)$ times for each choice of $\widehat{R}$. Hence, ROUGH ESTIMATION makes $O_d(\log^{d+1} n)$ CID_2^o queries. □

5 THE ROLE OF ROUGH ESTIMATION

In this section, we briefly overview the work [10, 11] that we improve upon; show how Theorem 1.4 follows from Theorem 1.3 and how our ROUGH ESTIMATION algorithm improves upon the COARSE algorithm of Dell et al. [10, 11].

Before getting into the reasons *why Theorem 1.4 follows from Theorem 1.3*, let us first review the algorithms for d-HYPEREDGE-ESTIMATION and d-HYPEREDGE-SAMPLE by Dell et al. [10, 11].

Overview of Dell et al. [10, 11]. Dell et al.'s algorithm for d-Hyperedge-Sample make repeated calls to d-Hyperedge-Estimation. Their algorithm for d-Hyperedge-Estimation calls mainly three subroutines over $O_d(\log n)$ iterations: Coarse, Halving, and Trim. Halving and Trim call Coarse repeatedly. So, Coarse is the main building block for their algorithms for d-Hyperedge-Estimation and d-Hyperedge-Sample.

Coarse *algorithm.* It estimates the number of hyperedges in the hypergraph up to polylogarithmic factors by using polylogarithmic queries. The result is formally stated as follows (see References [10, 11, Section 4]):

Lemma 5.1 (Coarse Algorithm by Dell et al. [10, 11]). *There exists an algorithm* Coarse, *which has* CID *query access to a hypergraph* $\mathcal{H}(U, \mathcal{F})$, *makes* $O_d(\log^{2d+3} n)$ CID *queries, and finds* $\widehat{m}$ *satisfying*

$$\Omega_d\left(\frac{1}{\log^d n}\right) \leq \frac{\widehat{m}}{m} \leq O_d\left(\log^d n\right)$$

with probability at least $1 - 1/n^{\Omega(d)}$.

Remark 1. The objective of the Coarse algorithm by Dell et al. is essentially the same as that of our Rough Estimation algorithm. Both of them can estimate the number of hyperedges in any induced subhypergraph. However, note that our Rough Estimation (as stated in Theorem 1.3) has better approximation guarantee and better query complexity than that of Coarse algorithm of Dell et al. (as stated in Lemma 5.1).

The framework of Dell et al. implies that the query complexity of d-Hyperedge-Estimation and d-Hyperedge-Sample can be expressed by the approximation guarantee and the query complexity of the Coarse algorithm. This is formally stated as follows:

Lemma 5.2 (d-Hyperedge-Estimation and d-Hyperedge-Sample in Terms of Quality of Coarse Algorithm [10, 11]). *Let there be an algorithm* Coarse, *which has* CID *query access to a hypergraph* $\mathcal{H}(U, \mathcal{F})$, *makes* q CID *queries, and finds* $\widehat{m}$ *satisfying* $\frac{1}{b} \leq \frac{\widehat{m}}{m} \leq b$ *with probability at least* $1 - 1/n^{\Omega(d)}$. *Then*

(i) d-Hyperedge-Estimation *can be solved by using*

$$O_d\left(\log^2 n\left(\log nb + \frac{b^2 \log^2 n}{\varepsilon^2}\right) q\right) \text{ CID } \textit{queries.}$$

(ii) d-Hyperedge-Sample *can be solved by using*

$$O_d\left(\log^6 n\left(\log nb + \frac{b^2 \log^2 n}{\varepsilon^2}\right) q\right) \text{ CID } \textit{queries.}$$

Why Theorem 1.4 follows from Theorem 1.3? Observe that we get Proposition 1.2 (the result of Dell et al.) from Lemma 5.1 by substituting $b = O_d(\log^d n)$ and $q = O_d(\log^{2d+3} n)$ in Lemma 5.2. In Theorem 1.4, we improve on Proposition 1.2 by using our main result (Theorem 1.3) and substituting $b = O_d(\log^{d-1} n)$ and $q = O_d(\log^{d+2} n)$ in Lemma 5.2.

The main reason we get an improved query complexity for hyperedge estimation in Theorem 1.4 as compared to Dell et al. (Proposition 5.2) is our Rough Estimation algorithm is an improvement over the Coarse algorithm of Dell et al. [10, 11] in terms of approximation guarantee as well as query complexity.

How our Rough Estimation *improves over* Coarse *of Dell et al. [10, 11]?* At a very high level, the frameworks of our Rough Estimation algorithm and that of Dell et al.'s Coarse algorithm might look similar, but the main ideas involved are quite different. Our Rough Estimation (as stated in Lemma 3.1) directly deals with the hypergraph (though the ordered one) and makes use of CID_2^o queries. Note that each CID_2^o query can be simulated by using $O_d(\log n)$ CID queries. However, Coarse algorithm of Dell et al. considers $O_d(\log n)$ independent random d-partite hypergraphs by partitioning the vertex set into d parts uniformly at random, works on the d-partite hypergraphs, and reports the median, of the $O_d(\log n)$ outputs corresponding to random d-partite subhypergraphs, as the final output. So, there is an $O_d(\log n)$ blow-up in both our Rough Estimation algorithm and Dell et al.'s Coarse algorithm, though the reasons behind the blow-ups are different.

Our Rough Estimation calls repeatedly ($O_d(\log n)$ times) Verify Estimate for each guess, where the total number of guesses is $O_d(\log n)$. In the Coarse algorithm, Dell et al. use repeated calls ($O_d(\log^{d+1} n)$) times to an analogous routine of our Verify Estimate, which they name Verify Guess, $O_d(\log n)$ times. Their Verify Guess has the following criteria for any guess M:

— If $M \geq \frac{d^d \log^{2d} n}{2^{3d-1}} m$, Verify Guess accepts M with probability at most p;
— If $M \leq m$, Verify Guess accepts M with probability at least $2p$;
— It makes $O_d(\log^d n)$ CID queries.

Recall that the number of CID_2 queries made by each call to Verify Estimate is $O_d(\log^{d-1} n)$, that is, $O_d(\log^d n)$ CID queries. So, in terms of the number of CID queries, both our Rough Estimation and Coarse of Dell et al. have the same complexity.

The probability p in Verify Guess of Dell et al. [10, 11] satisfies $p \approx_d \frac{1}{\log^d n}$, where $\approx_d$ is used to suppress the terms involving only d. So, for each guess M, their Coarse algorithm has to call Verify Guess $O_d(\frac{1}{p} \log n) = O_d(\log^{d+1} n)$ times to distinguish whether $M \leq m$ or $M \geq \frac{d^d \log^{2d} n}{2^{3d-1}} m$, with a probability at least $1 - 1/n^{\Omega(d)}$. So, the total number of queries made by the Coarse algorithm of Dell et al. [10, 11] is

$$O_d(\log n) \cdot O_d(\log n) \cdot O_d\left(\log^{d+1} n\right) \cdot O_d\left(\log^d n\right) = O_d\left(\log^{2d+3} n\right).$$

The first $O_d(\log n)$ term is due to the blow-up incurred to convert original hypergraph to d-partite hypergraph, the second $O_d(\log n)$ term is due to the number of guesses for m, the third $O_d(\log^{d+1} n)$ term is the number of times Coarse calls Verify Guess, and the last term $O_d(\log^d n)$ is the number of CID queries made by each call to Verify Guess.

As it can be observed from Lemmas 4.1 and 4.2, p in our case (Verify Estimate) is $\Omega_d(1)$. So, it is enough for Rough Estimation to call Verify Estimate only $O_d(\log n)$ times. Therefore, the number of CID queries made by our Rough Estimation is

$$O_d(\log n) \cdot O_d(\log n) \cdot O_d(\log^{d-1} n) \cdot O_d(\log n) = O_d(\log^{d+2} n).$$

In the above expression, the first $O_d(\log n)$ term is due to the number of guesses for m, the second $O_d(\log n)$ term is the number of times Rough Estimation calls Verify Estimate, the third $O(\log^{d-1} n)$ term is the number of CID_2 queries made by each call to Verify Estimate, and the last $O_d(\log n)$ term is the number of CID queries needed to simulate a CID_2 query with probability at least $1 - 1/n^{\Omega(d)}$.

We do the improvement in approximation guarantee as well as query complexity in Rough Estimation algorithm (as stated in Theorem 1.3), as compared to Coarse algorithm of Dell et al. [10, 11] (as stated in Lemma 5.1), by a careful analysis of the intersection pattern of the hypergraphs and setting the sampling probability parameters in Verify Estimate (Algorithm 1) algorithm in a nontrivial way, which is evident from the description of Algorithm 1 and its analysis.

APPENDICES

A SOME PROBABILITY RESULTS

Lemma A.1 (Chernoff-Hoeffding Bound, see Reference [13]). *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$, the following holds for any $0 \le \delta \le 1$:*

$$\mathbb{P}(|X - \mu| \ge \delta\mu) \le 2\exp\left(-\mu\delta^2/3\right).$$

Lemma A.2 (Chernoff-Hoeffding Bound, see Reference [13]). *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum_{i=1}^{n} X_i$ and $\mu_l \le \mathbb{E}[X] \le \mu_h$, the following holds for any $\delta > 0$:*

(i) $\mathbb{P}(X > \mu_h + \delta) \le \exp(-2\delta^2/n)$.
(ii) $\mathbb{P}(X < \mu_l - \delta) \le \exp(-2\delta^2/n)$.

B ORACLE DEFINITIONS

*Definition B.1 (Independent Set Query (*IS*) [2, 3]).* Given a subset A of the vertex set V of a graph $G(V, E)$, IS query answers whether A is an independent set.

*Definition B.2 (Bipartite Independent Set Oracle (*BIS*) [2, 3]).* Given two disjoint subsets A, B of the vertex set V of a graph $G(V, E)$, BIS query reports whether there exists an edge having endpoints in both A and B.

*Definition B.3 (Tripartite Independent Set Oracle (*TIS*) [4, 5]).* Given three disjoint subsets A, B, C of the vertex set V of a graph $G(V, E)$, the TIS oracle reports whether there exists a triangle having endpoints in A, B, and C.

*Definition B.4 (Generalized d-partite Independent Set Oracle (*CID*) [6]).* Given d pairwise disjoint subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ as input, CID query answers whether $m(A_1, \ldots, A_d) \neq 0$, where $m(A_1, \ldots, A_d)$ denotes the number of hyperedges in $\mathcal{H}$ having exactly one vertex in each A_i, $\forall i \in \{1, 2, \ldots, d\}$.

Definition B.5 (CID_1 *Oracle).* Given s pairwise disjoint subsets of vertices $A_1, \ldots, A_s \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ and $a_1, \ldots, a_s \in [d]$ such that $\sum_{i=1}^{s} a_i = d$, CID_1 query on input $A_1^{[a_1]}, A_2^{[a_2]}, \cdots, A_s^{[a_s]}$ answers whether $m(A_1^{[a_1]}, \ldots, A_s^{[a_s]}) \neq 0$.

Definition B.6 (CID_2 *Oracle).* Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$, CID_2 query on input $A_1, \ldots, A_d$ answers whether $m(A_1, \ldots, A_d) \neq 0$.

Definition B.7 (CID^o*:).* Given d pairwise disjoint subsets $A_1, \ldots, A_d$ of vertices of $U(\mathcal{H})$ of an ordered hypergraph $\mathcal{H}_o$ as input, CID^o query answers Yes if $m_o(A_1, \ldots, A_d) \neq 0$. Otherwise, the oracle answers No.

Definition B.8 (CID_1^o*:).* Given s pairwise disjoint subsets of vertices $A_1, \ldots, A_s \subseteq U(\mathcal{H}_o)$ of an ordered hypergraph $\mathcal{H}_o$ and $a_1, \ldots, a_s \in [d]$ such that $\sum_{i=1}^{s} a_i = d$, CID_1^o query on input $A_1^{[a_1]}, A_2^{[a_2]}, \cdots, A_s^{[a_s]}$ answers Yes if and only if $m_o(A_1^{[a_1]}, \ldots, A_s^{[a_s]}) \neq 0$. Recall that $A^{[a]}$ denotes the set A repeated a times.

Definition B.9 (CID_2^o*:).* Given any d subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H}_o)$ of an ordered hypergraph $\mathcal{H}_o$, CID_2^o query on input $A_1, \ldots, A_d$ answers Yes if and only if $m_o(A_1, \ldots, A_d) \neq 0$.

REFERENCES

[1] Raghavendra Addanki, Andrew McGregor, and Cameron Musco. 2022. Non-adaptive edge counting and sampling via bipartite independent set queries. In *30th Annual European Symposium on Algorithms (ESA'22) (LIPIcs,* Vol. 244*)*, Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2:1–2:16. DOI: https://doi.org/10.4230/LIPICS.ESA.2022.2

[2] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. 2018. Edge estimation with independent set oracles. In *9th Innovations in Theoretical Computer Science Conference (ITCS'18)*, Vol. 94. 38:1–38:21.

[3] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. 2020. Edge estimation with independent set oracles. *ACM Trans. Algor.* 16, 4 (2020), 52:1–52:27.

[4] Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. 2019. Triangle estimation using tripartite independent set queries. In *30th International Symposium on Algorithms and Computation (ISAAC'12)* Vol. 149. 19:1–19:17.

[5] Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. 2021. On triangle estimation using tripartite independent set queries. *Theor. Comput. Syst.* 65, 8 (2021), 1165–1192.

[6] Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. 2018. Parameterized query complexity of hitting set using stability of sunflowers. In *29th International Symposium on Algorithms and Computation (ISAAC'18)*, Vol. 123. 25:1–25:12.

[7] Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. 2023. Almost optimal query algorithm for hitting set using a subset query. *J. Comput. Syst. Sci.* 137 (2023), 50–65. DOI: https://doi.org/10.1016/J.JCSS.2023.02.002

[8] Holger Dell and John Lapinskas. 2018. Fine-grained reductions from approximate counting to decision. In *50th Annual ACM SIGACT Symposium on Theory of Computing (STOC'18)*. 281–288.

[9] Holger Dell and John Lapinskas. 2021. Fine-grained reductions from approximate counting to decision. *ACM Trans. Comput. Theory* 13, 2 (2021), 8:1–8:24.

[10] Holger Dell, John Lapinskas, and Kitty Meeks. 2020. Approximately counting and sampling small witnesses using a colourful decision oracle. In *ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*. 2201–2211.

[11] Holger Dell, John Lapinskas, and Kitty Meeks. 2022. Approximately counting and sampling small witnesses using a colorful decision oracle. *SIAM J. Comput.* 51, 4 (2022), 849–899. DOI: https://doi.org/10.1137/19M130604X arXiv:https://doi.org/10.1137/19M130604X

[12] Holger Dell, John Lapinskas, and Kitty Meeks. 2022. Nearly optimal independence oracle algorithms for edge estimation in hypergraphs. *CoRR* abs/2211.03874 (2022).

[13] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press.

[14] Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. 2017. Approximately counting triangles in sublinear time. *SIAM J. Comput.* 46, 5 (2017), 1603–1646. DOI: https://doi.org/10.1137/15M1054389

[15] Talya Eden, Dana Ron, and C. Seshadhri. 2020. On approximating the number of k-Cliques in sublinear time. *SIAM J. Comput.* 49, 4 (2020), 747–771.

[16] Uriel Feige. 2006. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM J. Comput.* 35, 4 (2006), 964–984.

[17] Oded Goldreich and Dana Ron. 2008. Approximating average parameters of graphs. *Rand. Struct. Algor.* 32, 4 (2008), 473–493.

[18] Jirí Matousek. 2002. *Lectures on Discrete Geometry (Graduate Texts in Mathematics,* Vol. 212). Springer.

Received 17 February 2023; revised 6 April 2024; accepted 8 April 2024