Topology Dependent Bounds For FAQs

Michael Langberg
University at Buffalo

Shi Li
University at Buffalo

Sai Vikneshwar Mani Jayaraman
University at Buffalo

Atri Rudra
University at Buffalo

ABSTRACT

In this paper, we prove topology dependent bounds on the number of rounds needed to compute Functional Aggregate Queries (FAQs) studied by Abo Khamis et al. [PODS 2016] in a synchronous distributed network under the model considered by Chattopadhyay et al. [FOCS 2014, SODA 2017]. Unlike the recent work on computing database queries in the Massively Parallel Computation model, in the model of Chattopadhyay et al., nodes can communicate only via private point-to-point channels and we are interested in bounds that work over an *arbitrary* communication topology. This model, which is closer to the well-studied CONGEST model in distributed computing and generalizes Yao's two party communication complexity model, has so far only been studied for problems that are common in the two-party communication complexity literature.

This is the first work to consider more practically motivated problems in this distributed model. For the sake of exposition, we focus on two specific problems in this paper: Boolean Conjunctive Query (BCQ) and computing variable/factor marginals in Probabilistic Graphical Models (PGMs). We obtain tight bounds on the number of rounds needed to compute such queries as long as the underlying hypergraph of the query is $O(1)$-degenerate and has $O(1)$-arity. In particular, the $O(1)$-degeneracy condition covers most well-studied queries that are efficiently computable in the centralized computation model like queries with constant treewidth. These tight bounds depend on a new notion of 'width' (namely *internal-node-width*) for Generalized Hypertree Decompositions (GHDs) of acyclic hypergraphs, which minimizes the number of internal nodes in a sub-class of GHDs. To the best of our knowledge, this width has not been studied explicitly in the theoretical database literature. Finally, we consider the problem of computing the product of a vector with a chain of matrices and prove tight bounds on its round complexity (over a finite field of two elements) using a novel min-entropy based argument.

KEYWORDS

Topology Dependent Bounds, CONGEST model, Probabalistic Graphical Models, Boolean Conjunctive Query, Communication Complexity Lower Bounds.

ACM Reference Format:
Michael Langberg, Shi Li, Sai Vikneshwar Mani Jayaraman, and Atri Rudra. 2019. Topology Dependent Bounds For FAQs. In *38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3294052.3319686

1 INTRODUCTION

In this paper, we prove topology dependent bounds on the number of rounds needed to compute Functional Aggregate Queries (FAQs) of [32] in a synchronous distributed network under the model considered by Chattopadhyay et al. [18, 19]. For ease of exposition, we consider the FAQ-SS problem [7, 32, 41] i.e., FAQ with a single semiring (also called *Marginalize a Product Function* in [3]), which is a special case of the general FAQ problem (defined in Section 5). In FAQ-SS, we are given a multi-hypergraph $\mathcal{H} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ where for each hyperedge $e \in \overline{\mathcal{E}}$ we are given an input function $f_e : \prod_{v \in e} \mathrm{Dom}(v) \to \mathbb{D}$. In addition we are given a set of *free variables*[1] $\mathcal{F} \subseteq \overline{\mathcal{V}}$ and our goal is to compute the function:

$$\phi_{\mathcal{F}}(\mathbf{x}) = \sum_{\mathbf{y} \in \prod_{v \in \overline{\mathcal{V}}} \mathrm{Dom}(v) : \mathbf{y}_{\mathcal{F}} = \mathbf{x}} \prod_{e \in \overline{\mathcal{E}}} f_e(\mathbf{y}_e) \qquad (1.0)$$

for every $\mathbf{x} \in \prod_{v \in \mathcal{F}} \mathrm{Dom}(v)$, where $\mathbf{y}_e$ and $\mathbf{y}_{\mathcal{F}}$ are $\mathbf{y}$ projected down to co-ordinates in $e \subseteq \overline{\mathcal{V}}$ for every $e \in \overline{\mathcal{E}}$ and $\mathcal{F} \subseteq \overline{\mathcal{V}}$ respectively. Further, all the operations are over the *commutative semiring*[2] $(\mathbb{D}, +, \cdot)$ with additive identity $\mathbf{0}$.

PODS'19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6227-6/19/06…$15.00
https://doi.org/10.1145/3294052.3319686

[1] We would like to mention here that our results hold only for specific choices of free variables.

[2] A triple $(\mathbb{D}, \oplus, \otimes)$ is a *commutative semiring* if $\oplus$ and $\otimes$ are commutative binary operators over $\mathbb{D}$ satisfying the following: (1) $(\mathbb{D}, \oplus)$ is a commutative monoid with an additive identity, denoted by $\mathbf{0}$. (2) $(\mathbb{D}, \otimes)$ is a commutative

As with database systems, we assume that the functions are given in *listing* representation i.e., the function f_e is represented as a list of its non-zero values: $R_e = \{(\mathbf{y}, f_e(\mathbf{y}))|\mathbf{y} \in \prod_{v\in e} Dom(v) : f_e(\mathbf{y}) \neq \mathbf{0}\}$.[3] We define $D = \max_{v\in\overline{\mathcal{V}}} |\mathrm{Dom}(v)|$, $N = \max_{e\in\overline{\mathcal{E}}} |R_e|$, $k = |\overline{\mathcal{E}}|$ and r as the maximum arity among all functions.

Though our results are semiring agnostic, we mention two special problems that we consider in this paper. The first problem is when $\mathcal{F} = \emptyset$ and the semiring is the *Boolean semiring* ($\mathbb{D} = \{0, 1\}, \vee, \wedge$). This corresponds to the *Boolean Conjunctive Query* (which we will call BCQ).[4] The other problem is when $\mathcal{F} = e$ for some $e \in \overline{\mathcal{E}}$ and the semiring is $(\mathbb{R}_{\geq 0}, +, \cdot)$, which corresponds to computing a *factor marginal* in *Probabilistic Graphical Models* (or PGMs) – here we think of f_e as a probability distribution. The FAQ setup (and even FAQ-SS) encompasses a large class of problems in varied domains. We refer the reader to the surveys [3, 33] for an overview of these applications.

Given a query $q = \left(\mathcal{H}, \{f_e\}_{e\in\overline{\mathcal{E}}}, \mathcal{F}\right)$, we will consider the number of rounds needed to compute q in a distributed environment. In particular, the underlying *communication topology*[5] $G = (V, E)$ is assumed to be a synchronous network and we would like to compute q on G with the following constraints [18, 19]. Initially, all functions $\{f_e\}_{e\in\overline{\mathcal{E}}}$ are assigned to specific nodes $K \subseteq V : 1 \leq |K| \leq k$ (called *players*). In each *round* of communication, $O(r \cdot \log_2(D))$ bits[6] can be simultaneously communicated on each edge in E (each such edge or *channel* is private to the nodes at its endpoints). At the end of the protocol, a pre-determined player in K knows the answer to q. Naturally, we would like to design protocols that minimize the total number of rounds of communication (rounds hereon) needed to compute q on G. More generally, we would like to obtain tight bounds depending on $\mathcal{H}$ and G for this problem for *every* query topology $\mathcal{H}$ and *every* network topology G. Note that we do not take into account the internal computation done by nodes in G and we assume that all nodes in V co-operatively compute the answer to q. Due to space constraints, many of the technical details are deferred to the version of this paper [38], which we refer to as full paper.

monoid with a multiplicative identity, denoted by $\mathbf{1}$. (In the usual semiring definition, we do not need the multiplicative monoid to be commutative.) (3) $\otimes$ distributes over $\oplus$. (4) For any element $d \in \mathbb{D}$, we have $d \otimes \mathbf{0} = \mathbf{0} \otimes d = \mathbf{0}$.

[3] We use function/relation interchangeably for f_e/R_e but both mean the same.

[4] $\mathcal{F} = \overline{\mathcal{V}}$ over the Boolean semiring is the natural join problem.

[5] Note that this is distinct from $\mathcal{H}$ and is just a simple graph: see Figure 1 for an example illustrating this difference.

[6] This is a natural choice since any tuple in any function can be communicated with at most $O(r \cdot \log_2(D))$ bits. Our bounds seamlessly generalize to the cases when each edge – (1) can transmit $B \neq r \cdot \log_2(D)$ bits and (2) has a different capacity, but for ease of exposition, we will not consider these generalizations in this paper.

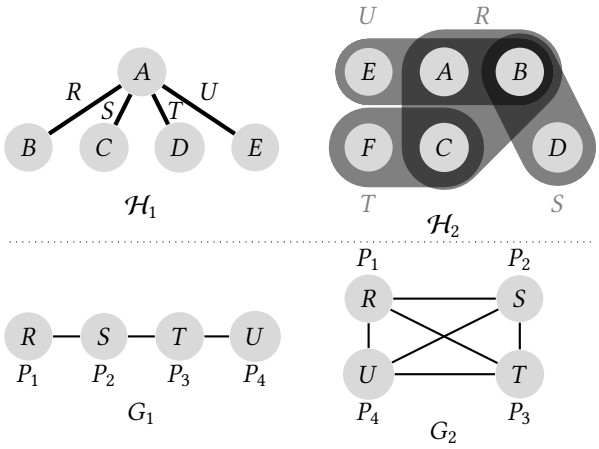

Figure 1: Two example queries $\mathcal{H}_1$ and $\mathcal{H}_2$ and two topologies – "line" G_1 and "clique" G_2. $\mathcal{H}_2$ has hyperedges $R(A, B, C)$, $S(B, D)$, $T(C, F)$ and $U(A, B, E)$.

1.1 Why this distributed model?

We believe that the strength of our model is its generality. Specifically, it captures query computation in three different paradigms, namely: (1) Computing the natural join query in the Massively Parallel Computation (MPC) model [2, 9, 10, 31, 36, 37], (2) Computing join and aggregation queries for sensor networks [13, 23, 40] and (3) Computation of FAQs on arbitrary topologies using *software defined networks* and optical reconfigurable networks like ProjecToR [24]. Before we discuss these in detail, we would like to mention that the CONGEST model in distributed computing has the same setup as ours [42] with one crucial difference. Unlike our case, where we can compute FAQs on *any* topology in the CONGEST model, the topologies for computing a fixed FAQ typically depend on the query itself.

The sequence of works in the MPC setting focus on computing the natural join q (which is a special case of FAQ-SS as mentioned earlier) on a topology G with p nodes, which is typically well-connected. Each round of communication has two phases – (1) internal computation among the nodes and (2) communication between the nodes bounded by a node capacity L. The goal in MPC is to minimize the number of rounds h needed for computing q. There are two different lines of work in this regime – one where p is fixed and the goal is to determine h, L [9, 10, 36, 37] and the other is when h, L are fixed and the goal is to determine p [2]. We compare both these classes of models with ours in Appendix A and present an executive summary here.

Roughly speaking, the MPC model defined in [9] is a special case of our model. We consider two different MPC models – one with no replication (which we call MPC(0) [9]) and one with replication (which we dub MPC(ϵ) [2, 36]). Both

these models have some differences from ours and among themselves. For instance, both these models assume a specific network topology G' (as opposed to any topology G in our case), work on node capacities L (as opposed to edge capacities in our setting) and prove bounds for the natural join problem (in contrast, our bounds apply for the more general FAQ). The input functions are systematically assigned to players in MPC(0) and are uniformly distributed among players in MPC(ϵ). The instantation of these models for the setting where p is fixed and h, L is to be determined is the closest to our model. In particular, when $\mathcal{H}$ is a star, our protocols obtain the same guarantees as MPC(0) and are slightly worse in MPC(ϵ). Our model does not (yet) handle the scenario when L is fixed and the goal is to determine p. Sensor networks are typically tree-like topologies, where the goal is to efficiently and accurately report aggregate queries on data generated by the sensors. Since the sensors can store only little data, these queries are typically restrictive. We show in the full paper [38] that our results imply bounds for some of these queries. Recently, Internet of Things (IoT) devices [1] show the promise of expanding the data storage/class of queries that can be computed on sensor networks. We believe that our model/results will find more relevance in the IoT setting since the sensors used posess more computation power than those considered in [40]. Finally, our work initiates the study of computation on general topologies to be used in emerging technologies like ProjecToR [24], which has been proposed for use in data centers where topologies can be changed based on the workload.

1.2 Summary of Our Contributions

Table 1 lists our results and Section 2 contains a detailed overview of techniques used to obtain the results. We summarize our contributions here. For the sake of brevity, we focus on the BCQ problem. Our main result is the following. For (hyper)graphs $\mathcal{H}$ with constant degeneracy[7] (d) and constant arity (r), we prove tight bounds (up to constant factors) for computing *any* BCQ on *any* network topology G. Constant treewidth implies constant d and, as a result, queries having constant d encompass most well-studied queries that are efficiently computable in the centralized computation model.

Upper Bounds. Our upper bounds need protocols for solving the following two basic algorithmic tasks: (1) set intersection and (2) sending all inputs to a single node. For (1), our protocol is new in the FAQ literature and for (2), we use a standard protocol from flow networks. Interestingly, our results highlight a notion of width of *acyclic* queries– the number of internal nodes for a subclass of GHDs[8] (defined in Section 2.2.2), which to the best of our knowledge, has not been explicitly studied in the database literature.

Lower Bounds. Our lower bounds follow from known lower bounds on the well-studied TRIBES function in two-party communication complexity literature (defined in Section 2.2.2). At a high level, we start with an arbitrary TRIBES instance and show that it can be reduced to a suitable BCQ instance in our model. We then prove lower bounds on the BCQ instance using known lower bounds on TRIBES.

We note here that the simplicity of our techniques allows us to extend our results to the general FAQ problem. Further, we would like to mention that extending our bounds to d-degenerate graphs with non-constant d has a known bottleneck of solving BCQ of $\mathcal{H}$ on G when $\mathcal{H}$ is a clique and G is an edge. In particular, the gaps dependent on d in Table 1 cannot be resolved without addressing this bottleneck.

Finally, we consider the following FAQ-SS problem of Chain Matrix-Vector Multiplication (MCM): computing $\mathbf{A}_k \cdot \mathbf{A}_{k-1} \cdot \ldots \cdot \mathbf{A}_1 \cdot \mathbf{x}$, where each player gets $\mathbf{x}, \mathbf{A}_1, \ldots, \mathbf{A}_k$ in order and they would like to compute the product over the finite field $\mathbb{F}_2$.[9] Note that this problem is different from the well-known Online Matrix Vector Multiplication problem[10] and is related to k layer neural networks.[11] We prove a tight bound for this problem. The upper bound is simple but the lower bound argument (though conceptually simple) is technically the most involved part of the paper. We use an entropy-based argument using min-entropy instead of the standard Shannon's entropy. This requires more care since we can no longer use the chain rule.

2 OUR MODEL AND DETAILED OVERVIEW OF OUR RESULTS

In this section, our goal is to provide a walkthrough of our results and techniques used to prove them. We start with a formal definition of our model. Then, we illustrate with examples our results for the case when $\mathcal{H}$ has arity at most two and subsequently, our new notion of width for GHDs. We conclude this section with our results on Chain Matrix-Vector Multiplication (MCM).

2.1 Our Model

We first define our model.

[7] Degeneracy is defined as the smallest d such that every sub(hyper)graph in $\mathcal{H}$ has a vertex of degree at most d.

[8] An internal node is a non-leaf node in a GHD.

[9] $\mathbb{F}_2$ has two elements: the additive identity $\mathbf{0}$ and multiplicative identity $\mathbf{1}$. Addition and Multiplication are all modulo 2.

[10] We illustrate this difference in the full paper [38].

[11] In neural networks, a non-linear function is applied after each matrix-vector multiplication and the multiplication is over reals instead of $\mathbb{F}_2$. Our lower bounds hold for this setting as well.

Query	G	d, r	Gap	Ref
FAQ	L	$O(1), O(1)$	$\widetilde{O}(1)$	Thm 5.1
FAQ	A	$O(1), O(1)$	$\widetilde{O}(1)$	Thm 5.1
BCQ	A	$d, 2$	$\widetilde{O}(d)$	Thm 4.1
FAQ	A	d, r	$\widetilde{O}(d^2 r^2)$	Thm 5.2
MCM*	L	$1, 2$	$O(1)$	Sec 6

Table 1: The first and second columns denote the query that we compute and topology on which the query is computed. In the second column, L denotes a line and A denotes an arbitrary G. The third column denotes the degeneracy (Definition 3.3) and arity conditions (d, r). The fourth column denotes the gap between our upper and lower bounds ignoring polylogarithmic factors in N and G (denoted by $\widetilde{O}$). The final column denotes the relevant result in this paper. Note that all our results except MCM (denoted by a '*') assume worst-case assignment of input functions in the query to nodes in G.

Model 2.1. *We are given a query q, its underlying hypergraph $\mathcal{H} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ with input functions f_e (having at most N non-zero values) for every $e \in \overline{\mathcal{E}}$ and a topology $G = (V, E)$. Further, each function is completely assigned to a unique node in V. It follows that there exists a subset $K : K \subseteq V$ that contains the players with functions and $|K| \leq k = |\overline{\mathcal{E}}|$. We assume $N \geq |V(G)|^2$ and consider worst-case inputs for the functions. We would like to compute* BCQ *(and more generally an* FAQ*) of $\mathcal{H}$ on G. To design a protocol for this computation, we assume that every node in G has the knowledge of $\mathcal{H}$ and G. In each round of the protocol, at most $O(r \cdot \log_2(D))$ bits can be communicated over every edge in E. In particular, this implies any subset of edges in G can communicate in the same round. Further, at the end of the protocol, a pre-determined player in K has the answer to q.*
Finally, given the above setup, our goal is to design protocols that minimize the total number of rounds needed to compute q assuming worst-case assignment of the functions to players in G. Note that we do not take into account the internal computation done by nodes in G and we assume that all nodes are always available in V (i.e., node failures do not happen) and they co-operatively compute the answer to q.

We prove both upper and lower bounds on the total number of rounds needed to compute q on G for every query hypergraph $\mathcal{H}$ and every topology G. While our upper bounds hold for any assignment of input functions to players in G, our lower bounds hold for a specific class of worst-case assignments of input functions to players in G. In Section 8, we further discuss the assumptions on $\mathcal{H}$ and G in the above model.
Before we move to our results for the case when $\mathcal{H}$ has arity at most two, we would like to point out that our bounds do not assume that the size of q is negligible compared to N, which is a standard assumption for computing database queries. Thus, our results are more general and in particular, for applications in PGMs, this is necessary since the size of q cannot be assumed as negligible w.r.t. N.

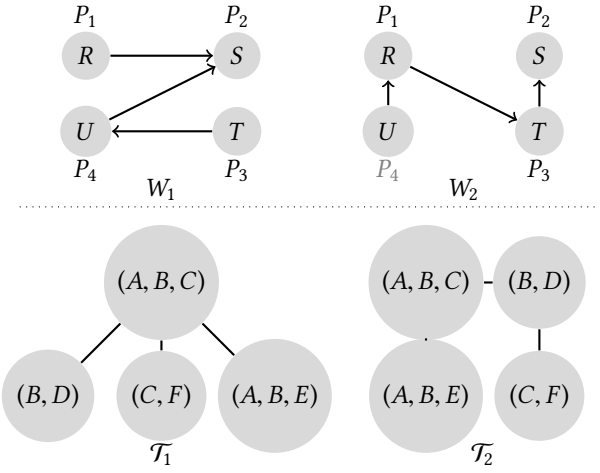

Figure 2: Two directed paths W_1 and W_2 for G_2 and two GHDs $\mathcal{T}_1$ (with 1 internal node) and $\mathcal{T}_2$ (with 2 internal nodes) both rooted at (A, B, C) for $\mathcal{H}_2$. G_2 and $\mathcal{H}_2$ are in Figure 1.

2.2 Arity Two $\mathcal{H}$

We consider the case when $\mathcal{H}$ has arity at most two and illustrate our upper and lower bound techniques through examples.

2.2.1 Upper Bounds. We start with a trivial protocol to compute any query $\mathcal{H}$ on any G. We then show how to improve upon it when $\mathcal{H}$ has a special structure. We use two extremal instances of G for an easy exposition of our results – a line (least connectivity) and a clique (full connectivity). We refer the reader to Figure 1 for all examples (except $\mathcal{H}_0$) considered in this section.

Trivial Protocol. There is always a *trivial protocol* to solve any query $\mathcal{H}$ on any G in which all players send their functions to one designated player who then computes the answer.
We consider the topologies G_1 and G_2 from Figure 1. We first start by computing a toy query $\mathcal{H}_0$ on G_1.

Example 2.1. Consider the query hypergraph $\mathcal{H}_0 = (\overline{\mathcal{V}} = \{A\}, \overline{\mathcal{E}} = \{R(A), S(A), T(A), U(A)\})$ i.e., all edges are self-loops on A and the line G_1. We would like to solve BCQ of $\mathcal{H}_0$ on G_1, which in Datalog format is $q_0() : -R(A), S(A), T(A), U(A)$. In G_1, player P_1 gets R, P_2 gets S, P_3 gets T and P_4 gets U. Then, solving BCQ of $\mathcal{H}_0$ on G_1 is equivalent to checking if

the set-intersection $R(A) \cap S(A) \cap T(A) \cap U(A)$ is empty. Let us assume that player P_4 needs to know the answer for this query.

We can solve this query in $N+2$ rounds as follows. In the first round, player P_1 sends a value $a \in \text{Dom}(A)$ such that there exists $R(a) = 1$ to player P_2 who then checks if $S(a) = 1$. More generally, in the i-th round, player P_j for $2 \leq j \leq 4$ receives an a from its left neighbor $(j-1)$ and checks if a is present in its table. If so, it passes a to its right neighbor $(j+1)$ (if $j \leq 3$) in the next $(i+1)$-th round. Otherwise, it does not pass anything. Notice that this protocol will terminate once all matching values of a are passed from P_1 to P_4 which takes $N+2$ rounds in the worst case. In other words, we are computing the semijoin (see Definition 3.5) query $((R(A) \ltimes S(A)) \ltimes T(A)) \ltimes U(A)$, which is equivalent to computing $R(A) \cap S(A) \cap T(A) \cap U(A)$. Note that this is much better than the *trivial protocol* for this case, which takes $3 \cdot N + 2$ rounds.

At the end of this protocol, P_4 knows the answer to the query. It is not too hard to see that we can extend the above protocol to the case when any other player say P_i for some $i \in [3]$ is designated to know the answer. In particular, we can orient G_1 in such a way that all paths are directed towards P_i and then run the protocol above simultaneously on all paths (there are at most two) towards P_i (recall that we assume knowledge of G for all nodes). Note that P_i would have the answer to the query and the new protocol takes $N+x$ rounds, where $x \leq 2$ depends on the choice of P_i.

It is not too hard to see that our protocol in the above example can be extended to the case when $\mathcal{H}$ is a star. We illustrate this in the following example.

Example 2.2. Consider the star $\mathcal{H}_1$ and the line G_1 in Figure 1. We would like to solve BCQ of $\mathcal{H}_1$ on G_1, which in Datalog format is $q_1() :- R(A,B), S(A,C), T(A,D), U(A,E)$. In G_1, player P_1 gets R, P_2 gets S, P_3 gets T and P_4 gets U. Then, BCQ of $\mathcal{H}_1$ is 1 iff $\pi_A(R) \cap \pi_A(S) \cap \pi_A(T) \cap \pi_A(U)$ is non-empty and 0 otherwise. Here, $\pi_A(\cdot)$ denotes the projection onto attribute A. We assume P_2 needs to know the answer for this query.

We can solve this query in $N+2$ rounds using the same protocol as in Example 2.1. In other words, we are computing the semjoin query[12] $((\pi_A(R) \ltimes \pi_A(S)) \ltimes \pi_A(T)) \ltimes \pi_A(U)$. Note that each node needs to compute $\pi_A(\cdot)$ internally but this doesn't need any communication between the nodes. At the end of this protocol, P_2 knows the answer to the query.

We now show how to do the same computation (i.e., BCQ of $\mathcal{H}_1$) on G_2.

[12]We would like to mention that casting the computation of BCQ on a star query as a semijoin is well-known [34].

Example 2.3. Consider the star $\mathcal{H}_1$ and the clique G_2 in Figure 1. We would like to compute BCQ of $\mathcal{H}_1$ on G_2, which in Datalog format is same as q_1 from Example 2.2. In G_2, player P_1 gets R, P_2 gets S, P_3 gets T and P_4 gets U. We assume that $\text{Dom}(A)$ is split into two halves and P_2 needs to know the answer for this query.

We can solve this query in $\frac{N}{2} + 2$ rounds as follows. We consider the two edge-disjoint directed paths W_1 and W_2 (see Figure 2) on G_2 that end with P_2. Our protocol from Example 2.2 runs on both these paths simultaneously with one caveat – the values of a in the first half of $\text{Dom}(A)$ are sent through W_1 and the ones in the second half of $\text{Dom}(A)$ are sent through W_2. Since both these directed paths involve the same set of nodes, our protocol is valid and takes only $\frac{N}{2} + 2$ rounds as claimed above. Note that this is better than our bound in Example 2.2.

The protocols in Examples 2.2 and 2.3 can be generalized to solve any star $\mathcal{H}$ on any G. Given the protocol for a star, there is a natural extension to $\mathcal{H}$ being a tree (or more generally a forest): we handle all the stars of the tree in a bottom-up fashion (starting with the stars at the "end" of the tree) and recurse. In particular, we can apply our protocol for the star case as a black-box on each of these stars. To extend this result to general d-degenerate graphs $\mathcal{H}$, we first decompose $\mathcal{H}$ into a forest and a *core* that contains the roots of all trees in the forest and all remaining vertices not in the forest. We run the above protocol on the forest and use the *trivial protocol* on the core. For general G, note that we need to find optimal ways of applying these protocols – for the forest part, we extend the idea of packing edge-disjoint paths from Example 2.3 to a Steiner tree (Definition 3.8) packing and for the *trivial protocol*, we use standard ideas from network flows (Definition 3.12). We would like to mention here that our upper bounds hold even when more than one function is assigned to a player (i.e., $|K| < k$). We will crucially exploit this fact in our lower bounds. We present more details in Section 4.1.

We are now ready to talk about our lower bounds.

2.2.2 Lower Bounds. All our lower bounds follow from known lower bounds on the well-studied TRIBES function (see [18] and references therein) in two-party communication complexity literature. To this end, we first consider an arbitrary TRIBES instance of a specific size and show that it can be reduced to a suitable two-party BCQ instance. In particular, solving the two-party BCQ instance we constructed indeed solves the TRIBES instance we started with. Thus, known lower bounds on TRIBES imply lower bounds for BCQ. Finally, we generalize our results from the two-party setting to general G using ideas from [18, 19] and exploit the fact that our (upper and) lower bounds are for worst-case input

functions and worst-case assignments of input functions to players in G.
We start by defining the two-party communication complexity model as a special case of Model 2.1.

Model 2.2. *Consider two players Alice (a) and Bob (b) on a graph $\mathcal{G} = (V = \{a, b\}, E = \{(a, b)\})$ with strings $\bar{X} = (X_1, \ldots, X_m)$ and $\bar{Y} = (Y_1, \ldots, Y_m)$, where $X_i, Y_i \in \{0,1\}^N$. Further, Alice gets $\bar{X}$, Bob gets $\bar{Y}$ and both have knowledge of only their inputs. The goal for these two players is to compute the boolean function $f(\bar{X}, \bar{Y}) : \{0,1\}^{m \cdot N} \times \{0,1\}^{m \cdot N} \to \{0,1\}$. The randomized two-party communication complexity of computing f, denoted by $\mathcal{R}(f(\bar{X}, \bar{Y}), \mathcal{G}, \{a, b\})$, is defined as the minimum worst-case number of rounds[13] needed by a randomized protocol that deterministically computes $f(\bar{X}, \bar{Y})$ with error at most $\frac{1}{3}$.*

We would like to mention that considering the randomized two-party communication complexity over its deterministic counterpart makes our lower bounds only stronger. We define TRIBES and state the lower bound result that we will use in our arguments.

Theorem 2.3 (Jayram et. al [30]). *Let $\text{TRIBES}_{m,N}(\bar{X}, \bar{Y}) \equiv \bigwedge_{i=1}^{m} \text{DISJ}_N(X_i, Y_i)$, where $\text{DISJ}_N(X_i, Y_i)$ is 1 if $X_i \cap Y_i \neq \emptyset$ and 0 otherwise, $X_i, Y_i \in \{0,1\}^N$ for every $i \in [m]$ and $\bar{X} = (X_1, \ldots, X_m)$, $\bar{Y} = (Y_1, \ldots, Y_m)$. Note that in the two-party model, Alice gets $\bar{X}$ and Bob gets $\bar{Y}$. Given this setup, we have*

$$\mathcal{R}\left(\text{TRIBES}_{m,N}(\bar{X}, \bar{Y}), \mathcal{G}, \{a, b\}\right) \geq \Omega(m \cdot N).$$

We start with an arbitrary TRIBES instance $\text{TRIBES}_{m,N}(\bar{X}, \bar{Y})$ of a suitable size and show that it can be reduced to a suitable two-party BCQ instance $\text{BCQ}_{\mathcal{H},\bar{X},\bar{Y}}$, where m is a function of $\mathcal{H}$. In particular, such a reduction would imply

$$\begin{aligned}\mathcal{R}\left(\text{BCQ}_{\mathcal{H},\bar{X},\bar{Y}}, \mathcal{G}, \{a, b\}\right) &\geq \mathcal{R}\left(\text{TRIBES}_{m,N}(\bar{X}, \bar{Y}), \mathcal{G}, \{a, b\}\right) \\ &\geq \Omega(m \cdot N),\end{aligned}$$

where the final inequality follows from Theorem 2.3. The above inequality implies the following since we consider worst-case input functions for a fixed $\mathcal{H}$.

$$\mathcal{R}\left(\text{BCQ}_{\mathcal{H},N}, \mathcal{G}, \{a, b\}\right) \geq \mathcal{R}\left(\text{BCQ}_{\mathcal{H},\bar{X},\bar{Y}}, \mathcal{G}, \{a, b\}\right), \quad (1)$$

where $\text{BCQ}_{\mathcal{H},N}$ denote the class of problems where all functions in $\mathcal{H}$ have size at most N. We generalize the above result to any G using ideas from [18, 19]. We consider an appropriate cut $C = (A, B)$ of G that partitions V into two vertex-disjoint subsets A and B and a correponding assignment, where each function $e \in \overline{\mathcal{E}}(\mathcal{H})$ is assigned to a node in either A or B. Since this is a valid assignment of functions in $\mathcal{H}$ to players in G, the minimum number of rounds needed to compute an instance of $\text{BCQ}_{\mathcal{H},N}$ on G assuming worst-assignments of functions to players in K, denoted by $\mathcal{R}\left(\text{BCQ}_{\mathcal{H}}, G, K\right)$, is at least $\frac{\mathcal{R}\left(\text{BCQ}_{\mathcal{H},\bar{X},\bar{Y}}, \mathcal{G}, \{a,b\}\right)}{\text{MinCut}(G,K)\lceil\log(\text{MinCut}(G,K))\rceil}$. We reconsider $\mathcal{H}_1$ and G_1 from Example 2.2 here.

Example 2.4. Recall we proved an upper bound of $N + 2$ for computing BCQ of $\mathcal{H}_1$ on G_1. We start with an arbitrary $\text{TRIBES}_{m=1,N}(\bar{X} = (X_1), \bar{Y} = (Y_1))$ instance. With a slight abuse of notation, we treat X_1, Y_1 as subsets of $[N]$ (instead of elements in $\{0,1\}^N$). We now construct a corresponding $\text{BCQ}_{\mathcal{H}_1,\bar{X},\bar{Y}}$ instance from the TRIBES one as follows – we assign $R(A, B) = X_1 \times \{1\}$, $S(A, C) = T(A, D) = [N] \times \{1\}$ and $U(A, E) = Y_1 \times \{1\}$. It is not too hard to see that $\text{BCQ}_{\mathcal{H}_1,\bar{X},\bar{Y}}$ is 1 iff $\text{TRIBES}_{1,N}(\bar{X}, \bar{Y})$ is 1, implying that solving the BCQ instance would solve the TRIBES instance. Finally, to obtain a lower bound for computing $\text{BCQ}_{\mathcal{H}_1,\bar{X},\bar{Y}}$ on the line G_1, we only need a cut where R and U are on different sides. We consider the cut $C = (\{P_1, P_2\}, \{P_3, P_4\})$ of G_1 and the assignment where P_1 gets R, P_2 gets S, P_3 gets T and P_4 gets U. Then, we can us (1) and Theorem 2.3 to obtain the required lower bound of $\Omega(N)$ since $\text{MinCut}(G, K) = 1$. Note that the above lower bound holds for any star $\mathcal{H}$. The same TRIBES instance can be used for Examples 2.1 and 2.3 as well. While a similar assignment holds for Example 2.1, Example 2.3 requires a different assignment where $C = \{P_1\}, \{P_2, P_3, P_4\})$ and P_1 gets R and S, P_2 gets T and P_3 gets U. Note that more than one input function can be assigned to the same player in G.

For general d-degenerate graphs $\mathcal{H}$, we start by recalling that m (i.e., size of the TRIBES instance) is a function of $\mathcal{H}$. As mentioned in Section 2.2.1, we can decompose $\mathcal{H}$ into a forest and a *core*. We prove three different lower bounds on $\mathcal{H}$, where the size of the TRIBES instance m used in our reduction is the maximum of three different bounds, each one on a different part of $\mathcal{H}$. The first one is on $\mathcal{H}$'s forest part, the second and third ones are on $\mathcal{H}$'s *core* part – lower bounded by applying Moore's bound [5] and Turan's theorem [6] respectively. For each case, we show that we can reduce the TRIBES instance to a suitable two-party BCQ instance. Thus, known lower bounds on the TRIBES instance from Theorem 2.3 apply for the BCQ instance. Finally, to generalize our results from two-party BCQ to general G, we use ideas from [18, 19] to obtain an appropriate cut for G and use lower bounds from the induced two-party communication complexity problem across the cut. Note that the assignment of functions depends on the cut. We present the details in Section 4.2.
For constant d, our upper and lower bounds match. However, for non-constant d, we have a gap of $\widetilde{O}(d)$. We would like to note that there is a fundamental bottleneck in getting rid of this factor as the case of $\mathcal{H}$ being a clique is an outstanding

[13] In each round, we assume at most one bit is sent from a to b instead of $O(\log_2(r \cdot D))$ bits to be consistent with the two-party communication complexity literature.

open question[14] (even in Model 2.2) and seems beyond the reach of current communication complexity techniques [16].

2.3 Notion of Width

We start by defining the notion of *GHD*s and acyclic (hyper)graphs.

Definition 2.4 (GHD). *A GHD of $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is defined by a triple $\langle \mathcal{T}, \chi, \lambda \rangle$, where $\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}))$ is a tree, $\chi : V(\mathcal{T}) \to 2^{\mathcal{V}}$ is a function associating a set of vertices $\chi(v) \subseteq \mathcal{V}$ to each node v of $\mathcal{T}$, and $\lambda : V(\mathcal{T}) \to 2^{\mathcal{E}}$ is a function associating a set of hyperedges to each node v of $\mathcal{T}$ such that the following two properties hold. First, for each $e \in \mathcal{E}$, there is at least one node $v \in V(\mathcal{T})$ such that $e \subseteq \chi(v)$ and $e \in \lambda(v)$. Second, for every $V' \subseteq \mathcal{V}$, the set $\{v \in V(\mathcal{T}) | V' \subseteq \chi(v)\}$ is connected in $\mathcal{T}$, called the running intersection property (RIP hereon). We only consider rooted GHDs.*
A reduced-GHD *has the additional property that every hyperedge $e \in \overline{\mathcal{E}}$ has a unique node $v \in V(\mathcal{T})$ such that $\chi(v) = e$ (note that this is an equality).*

Definition 2.5 (Acyclicity). *A hypergraph $\mathcal{H} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ is acyclic iff there exists a GHD $(\mathcal{T}, \chi, \lambda)$ in which for every node $v \in V(\mathcal{T})$, $\chi(v)$ is a hyperedge in $\overline{\mathcal{E}}$.*

We now define the sub-classes of reduced-GHDs that we consider in this paper. In particular, we construct reduced-GHDs using the GYO algorithm [28, 46, 51] (GYOA, also called GYO-Elimination order) and call them GYO-GHDs. We start by defining the *GYO-reduction* $\mathcal{H}'$ of a hypergraph $\mathcal{H}$.

Definition 2.6 (GYO-reduction and GYOA). *For any hypergraph $\mathcal{H}$, the GYO-reduction $\mathcal{H}'$ is defined as the leftover hypergraph after running GYOA on $\mathcal{H}$. We describe GYOA here. The input to GYOA is $\mathcal{H}$ and its output is a hypergraph $\mathcal{H}'$. It performs the following two steps iteratively on $\mathcal{H}'$ (starting with $\mathcal{H}' = \mathcal{H}$): (a) Eliminate a vertex that is present in only one hyperedge and (b) Delete a hyperedge that is contained in another hyperedge. GYOA terminates when it cannot perform any of the above two steps on $\mathcal{H}'$.*

We note here that running GYOA on an acyclic hypergraph $\mathcal{H}$ returns an empty $\mathcal{H}'$. For general hypergraphs $\mathcal{H}$, we start by running GYOA on $\mathcal{H}$, which returns a hypergraph $\mathcal{H}'$. Note that the hyperedges removed in this process form a forest of acyclic hypergraphs. For each hypertree in the forest of acyclic hypergraphs, we can construct a reduced-GHD and root it arbitrarily. We now define $\mathrm{C}(\mathcal{H})$ and $\mathrm{F}(\mathcal{H})$ based on the GYO-reduction $\mathcal{H}'$ and the reduced-GHDs constructed for the original hyperedges of $\mathcal{H}$ removed during GYOA.

Definition 2.7 ($\mathrm{C}(\mathcal{H}), \mathrm{F}(\mathcal{H})$). *$\mathrm{C}(\mathcal{H})$ is the union of $\mathcal{H}'$ and the root in each reduced-GHD we constructed. $\mathrm{F}(\mathcal{H}) = \mathcal{H} \setminus \mathcal{H}'$.*

[14] We state this problem formally in the full paper [38].

We are now ready to construct GYO-GHDs.

Construction 2.8. *Let $\mathcal{T}$ be the GYO-GHD be obtained from the following procedure. We define the root r' of $\mathcal{T}$ with $\chi(r') = V(\mathrm{C}(\mathcal{H}))$. For each edge $e \in \overline{\mathcal{E}}$ with $e \subset V(\mathrm{C}(\mathcal{H}))$, we create a new node v'_e in $\mathcal{T}$ with $\chi(v'_e) = e$ and add the edge (r', v'_e) to $\mathcal{T}$ in order to make it a reduced-GHD. For every reduced-GHD $\mathcal{T}'$ in $\mathrm{F}(\mathcal{H})$, we add the edge (r', r'') to $\mathcal{T}$, where r'' is the root of $\mathcal{T}'$. We add all the remaining nodes and edges in each reduced-GHD in $\mathrm{F}(\mathcal{H})$ to $\mathcal{T}$.*

We argue that the above procedure produces a reduced-GHD in the full paper [38]. Our new notion of width based on GYO-GHDs, which we call y (Internal Node Width), is defined as follows.

Definition 2.9.

$$y(\mathcal{H}) = \min_{\forall \mathcal{T} : \mathcal{T} \text{ is a GYO-GHD of } \mathcal{H}} y(\mathcal{T})$$

where $y(\mathcal{T})$ is the number of internal/non-leaf nodes in $\mathcal{T}$.

Unless specified otherwise, in the rest of the paper when we refer to GHDs, we are referring to GYO-GHDs. As an example in Figure 2, we consider two different GHDs $\mathcal{T}_1$ and $\mathcal{T}_2$ for the acyclic hypergraph $\mathcal{H}_2$ from Figure 1. Both are outcomes of Construction 2.8 and while $\mathcal{T}_2$ has two internal nodes, $\mathcal{T}_1$ has only one, implying $y(\mathcal{H}) = 1$. For $\mathcal{H}_1$ in Figure 1, it is easy to construct a GHD with one internal node (i.e., $y(\mathcal{H}) = 1$) by keeping (A, B) as the root and $(A, C), (A, D), (A, E)$ as leaves. We show how this can be achieved for simple graphs $\mathcal{H}$ in Section 4.

2.4 Chain Matrix-Vector Multiplication

Finally, in this work, we consider the problem of computing $\mathbf{A}_k \cdots \mathbf{A}_1 \mathbf{x}$ where the computation is over $\mathbb{F}_2$. The player P_i gets $\mathbf{A}_i$ for $i \in [k]$ and P_0 gets $\mathbf{x}$. Player P_{k+1} wants to know the answer (and does not have any input). The topology G is a line with P_i connected to P_{i+1} for $0 \leq i \leq k$. We show that when $k \leq N$ the natural algorithm that computes the partial product $\mathbf{A}_i \cdots \mathbf{A}_1 \mathbf{x}$ at P_i taking $\Theta(kN)$ rounds is indeed optimal. By contrast, if the matrices are assigned randomly to the players then the optimal number of rounds is $\Theta(k^2 N)$ (this follows from a trivial protocol). On the other extreme, if all matrices are assigned to one player, then the problem is trivial. So we are proving a tight lower-bound for arguably the simplest assignment of matrices to players that is not trivial.

We note that the existing technique of [18] cannot prove a lower bound better than $\Omega(N)$ for this problem (see the full paper [38] for a more detailed description). To get a better lower bound of $\Omega(kN)$, we use an entropy based inductive argument to show that at end of the $\Omega(iN)$ rounds, in player P_i's view, $\mathbf{A}_{i-1} \cdots \mathbf{A}_1 \mathbf{x}$ has very high entropy. However, Shannon's entropy is too weak for this argument to go through

and we use the stronger notion of min-entropy, which is omnipresent in pseudorandomness and cryptography [49]. Unfortunately, this means that we can no longer appeal to the chain rule and the arguments become a bit more delicate. Finally, in the process we prove the following natural result: if $\mathbf{A}$ and $\mathbf{x}$ have high enough min-entropy, then $\mathbf{Ax}$ has higher min-entropy than $\mathbf{x}$. To the best of our knowledge this result is new, though it follows by combining known results in pseudorandomness.[15]

3 PRELIMINARIES AND NOTATION

In this section, we define some notions related to the query hypergraph $\mathcal{H}$ and network topology G. We conclude the section with some asymptotic notation.

Query (Hyper)graph $\mathcal{H}$.

Definition 3.1 ($n_2(\mathcal{H})$). *Using Definition 2.7, we can decompose any $\mathcal{H}$ into a core $\mathrm{C}(\mathcal{H})$ and a forest $\mathrm{F}(\mathcal{H})$. We define $n_2(\mathcal{H}) = |V(\mathrm{C}(\mathcal{H}))|$.*

Definition 3.2 (Degree). *The degree of a vertex $v \in \mathcal{H}$ is given by $\left|\{e \ni v : e \in \overline{\mathcal{E}}\}\right|$.*

Definition 3.3 (d-degenerate (hyper)graph [35]). *In a d-degenerate (hyper)graph, every sub(hyper)graph has a vertex of degree at most d.*

We now define *natural join* and *semijoin*.

Definition 3.4 (Natural Join). *The* natural join $J = \bowtie_{e \in \overline{\mathcal{E}}} R_e$ *is a relation J with attribute set $V(\mathcal{H})$ satisfying the following condition (where $\bowtie$ denotes the* join *operator). A tuple $\mathbf{t} \in J$ iff for every $e \in E(\mathcal{H})$, the* projection *of $\mathbf{t}$ onto attributes in $v(e)$ - denoted by $\pi_{v(e)}(\mathbf{t})$ - belongs to R_e. Note that $J \subseteq \prod_{v \in V(\mathcal{H})} \mathrm{Dom}(v)$.*

Definition 3.5 (Semijoin). *A semijoin $J' = R_1 \ltimes R_2$ of relations R_1 and R_2 is defined as $J' = R_1 \bowtie \pi_{\mathrm{attr}(R_1) \cap \mathrm{attr}(R_2)}(R_2)$, where $\mathrm{attr}(\cdot)$ denotes the attribute set of the relations and $\ltimes$ is the semijoin operator.*

We show in the full paper [38] that *natural join* and *semijoin* are special cases of FAQ.

Network Topology G. We define some standard graph notions that will be used throughout the paper.

Definition 3.6 ($\mathrm{MinCut}(G, K)$). *We denote the size of the minimum cut of G separating vertices in K by* $\mathrm{MinCut}(G, K)$.

Definition 3.7 (Star Graph). *A star is a tree on n vertices with one internal node and $n-1$ leaves (e.g. $\mathcal{H}_1$ in Figure 1).*

[15] We thank David Zuckerman for showing us the high level proof idea of this result.

Definition 3.8 (Steiner tree). *Given a graph $G = (V, E)$ and a set of nodes $K \subseteq V$, we call a tree T a Steiner tree if it connects all vertices in K only using edges in E.*

In particular, we are interested in Steiner trees with diameter at most Δ (i.e., distance between any two nodes in K). Let $\mathcal{T}_{\Delta, K}$ denote the set of all such Steiner trees.

Definition 3.9 ($\mathrm{ST}(G, K, \Delta)$). $\mathrm{ST}(G, K, \Delta)$ *denotes the maximum number of edge disjoint Steiner trees from $\mathcal{T}_{\Delta, K}$ that can be packed in G.*

We will need this result:

Theorem 3.10 ([39]).

$$\mathrm{ST}(G, K, |V(G)|) = \Omega(\mathrm{MinCut}(G, K)).$$

Finally, we state a recent result under Model 2.1 on set-intersection queries over any topology G and any subset of players $K \subseteq V : |K| \leq k$, which we will use frequently in our arguments.

Theorem 3.11 ([18]). *Let $\mathbf{x}_u \in \{0,1\}^N$ for every player $u \in K$. The number of rounds taken by a protocol that deterministically computes $\bigwedge_{u \in K} \mathbf{x}_u$ (where the $\wedge$ is bit-wise AND) is given by $\Theta\left(\min_{\Delta \in [|V|]} \left(\frac{N}{\mathrm{ST}(G,K,\Delta)} + \Delta\right)\right)$.*

We will use the following notation for a special case of a multi-commodity flow problem:

Definition 3.12. *For every graph G, subset of players K and integer $N' \geq 0$, let $\tau_{\mathrm{MCF}}(G, K, N')$ be the minimum number of rounds needed to route at most $N' \log_2(N')$ bits from all players in K to any one player in K, assuming $\log_2(N')$ bits are sent in each round.* [16]

Let the minimum number of rounds taken by a protocol to determinstically compute BCQ of $\mathcal{H}$ on G be denoted by $\mathcal{D}(\mathrm{BCQ}_{\mathcal{H},N}, G, K)$, where each function in $\mathcal{H}$ has size at most N and is assigned to some player in $K \subseteq V, |K| \leq k$. Recall that $\mathcal{R}(\mathrm{BCQ}_{\mathcal{H},N}, G, K)$ is themost minimum worst-case number of rounds needed to deterministically compute any instance in $\mathrm{BCQ}_{\mathcal{H},N}$ with error at most $\frac{1}{3}$. The *trivial protocol*, along with Definition 3.12, implies the following.

Lemma 3.13.

$$\mathcal{D}(\mathrm{BCQ}_{\mathcal{H},N}, G, K) = O\left(\tau_{\mathrm{MCF}}(G, K, k \cdot r \cdot N)\right).$$

3.1 Asymptotic Notation

For notational clarity, we will ignore a factor of $\log_2(N) \cdot \log_2(\mathrm{MinCut}(G, K)) \cdot \log_2(n_2(\mathcal{H}))$ in our lower bounds. Further, we ignore these factors while arguing for the tightness of our bounds, which we denote by $\widetilde{\Omega}(\cdot)$, $\widetilde{O}(\cdot)$ and $\widetilde{\Theta}(\cdot)$.

[16] Here, we will consider the worst-case over all possible ways the $N' \log_2(N')$ bits are distributed over K. While our upper bounds can be smaller than this, we use this worst-case measure to simplify our bounds.

4 $\mathcal{H}$ IS A SIMPLE GRAPH

In this section, we consider the class of queries $\text{BCQ}_{\mathcal{H},N}$ for a given d-degenerate graph $\mathcal{H}$ with arity r at most two and all functions have size at most N. We prove upper and lower bounds that are tight within a factor of $\widetilde{O}(d)$ for computing any query in $\text{BCQ}_{\mathcal{H},N}$. The following is our main result.

Theorem 4.1. *For arbitrary topology G, subset of players K and d-degenerate simple graph $\mathcal{H}$, we have*

$$\begin{aligned}&\mathcal{D}(\text{BCQ}_{\mathcal{H},N}, G, K) = \\ &O\left(y(\mathcal{H}) \cdot \min_{\Delta\in[|V|]}\left(\frac{N}{\text{ST}(G,K,\Delta)} + \Delta\right)\right) \\ &+ O\left(\tau_{\text{MCF}}(G, K, n_2(\mathcal{H}) \cdot d \cdot N)\right). \end{aligned} \tag{1.1}$$

Further, for all simple graphs $\mathcal{H}$, we have

$$\begin{aligned}&\mathcal{R}(\text{BCQ}_{\mathcal{H},N}, G, K) \geq \\ &\widetilde{\Omega}\left(\frac{y(\mathcal{H}) \cdot N}{\text{MinCut}(G,K)}\right) + \widetilde{\Omega}\left(\frac{n_2(\mathcal{H}) \cdot N}{\text{MinCut}(G,K)}\right). \end{aligned} \tag{1.2}$$

We would like to point out that our upper bound holds for every assignment of the functions f_e to players in K while our lower bound holds for some assignment of functions to players in K. We first prove the upper bound (1.1), followed by the lower bound (1.2). Finally, we argue how our bounds are tight up to a factor of $\widetilde{O}(d)$.

4.1 Upper Bound

We first consider the case when $\mathcal{H}$ is a star, which will be a basic building block for our algorithms for general $\mathcal{H}$.

4.1.1 $\mathcal{H}$ is a star. Let $P = (v_0, v_1, \ldots, v_k)$ be the vertices of the star with v_0 as it's center. In this case, $\mathcal{H}$ includes k relations of the form R_{v_0,v_i} for every $i \in [k]$. Note that computing the corresponding BCQ query q can be solved via a *set-intersection* problem where we compute $R'_P = \bigcap_{i=1}^{k} R'_{v_i}$, where $R'_{v_i} = \{a_0 | (a_0, a_i) \in R_{v_0,v_i} \text{ for some } a_i \in \text{Dom}(v_i)\}$. It is easy to see that the final output of q is 1 if $R'_P \neq \emptyset$ and 0 otherwise. We can solve the resulting set intersection problem using Theorem 3.11 to compute R'_P. The procedure to compute R'_P is described in Algorithm 1, which when combined with the fact that at most $O(\log_2(D))$ bits can be communicated in each round, implies the following result.

Corollary 4.2. *When $\mathcal{H}$ is a star, for arbitrary graphs G and subset of players K, we have*

$$\mathcal{D}(\text{BCQ}_{\mathcal{H},N}, G, K) = O\left(\min_{\Delta\in[|V|]}\left(\frac{N}{\text{ST}(G,K,\Delta)} + \Delta\right)\right).$$

For the case when G is a line with k vertices, note that $\text{ST}(G, K, \Delta) = 0$ for every $\Delta > k-1$ and $\text{ST}(G, K, k-1) = 1$, which in turn implies the following.

Corollary 4.3. *Let $\mathcal{H}$ be a star and G be a line with k vertices. Then*

$$\mathcal{D}(\text{BCQ}_{\mathcal{H},N}, G, K) \leq N + k.$$

Note that the above result is a generalization of the upper bound in Example 2.2.

Algorithm 1 Algorithm for Star

1: **Input:** A star query with attributes $P = (v_0, \ldots, v_k)$ and relations $\{R_{(v_0,v_i)} : i \in [k]\}$. Note that v_0 is the center.
2: **Output:** R'_P
3: Each player containing a relation R_{v_0,v_i} computes $R'_{v_i} = \{a_0 | (a_0, a_i) \in R_{v_0,v_i} \exists a_i \in \text{Dom}(v_i)\}, \forall i \in [k]$ internally.
4: $R'_P = \bigcap_{i=1}^{k} R'_{v_i}$ is computed using Theorem 3.11.
5: **return** R'_P

4.1.2 $\mathcal{H}$ is a forest. We now use the above algorithm to obtain upper bounds for the case when $\mathcal{H}$ is a forest.

Lemma 4.4. *For arbitrary G, subset of players K and $\mathcal{H}$ being a forest, we have*

$$\mathcal{D}(\text{BCQ}_{\mathcal{H},N}, G, K) = O\left(y(\mathcal{H}) \cdot \min_{\Delta\in[|V|]}\left(\frac{N}{\text{ST}(G,K,\Delta)} + \Delta\right)\right). \tag{2}$$

Proof Sketch. We keep removing stars from trees in $\mathcal{H}$ in a bottom-up fashion and solve the induced query on each removed star using Algorithm 1. Since the number of stars we remove in this process is $y(\mathcal{H})$, the stated bound follows. The details are in the full paper [38]. □

4.1.3 The general case: d-degenerate graphs. We now state our upper bound when $\mathcal{H}$ is a d-degenerate simple graph:

Lemma 4.5. *For arbitrary G, subset of players K, and any d-degenerate simple graph $\mathcal{H}$, we have*

$$\begin{aligned}&\mathcal{D}(\text{BCQ}_{\mathcal{H},N}, G, K) = \\ &O\left(y(\mathcal{H}) \cdot \min_{\Delta\in[|V|]}\left(\frac{N}{\text{ST}(G,K,\Delta)} + \Delta\right)\right) \\ &+ O\left(\tau_{\text{MCF}}(G, K, n_2(\mathcal{H}) \cdot d \cdot N)\right). \end{aligned} \tag{2.1}$$

Proof Sketch. We decompose $\mathcal{H}$ into two components using Definition 2.7 – forest ($\text{F}(\mathcal{H})$) and core ($\text{C}(\mathcal{H})$). We then use Lemma 4.4 to solve the induced query on $\text{F}(\mathcal{H})$. For the core, we use the *trivial protocol* of sending all the remaining relations to one player. The details are in the full paper [38]. □

4.2 Lower Bound

We start with an overview, followed by lower bounds for the case when $\mathcal{H}$ is a forest and conclude with lower bounds for all simple graphs $\mathcal{H}$.

4.2.1 Overview. As we showed in Section 2.2.2, we start by considering an arbitrary TRIBES instance of size m where m is a function of $\mathcal{H}$. We then show that it can be reduced to a suitable two-party BCQ instance, which is functionally equivalent to the TRIBES instance we started with. In particular, solving the BCQ instance we constructed indeed solves the TRIBES instance we started with. We denote this reduction succinctly by $\text{TRIBES}_{m,N} \leq \text{BCQ}_{\mathcal{H},N}$. Finally, we generalize our results from the two-party setting to general G using ideas from [18, 19]. We crucially exploit the fact that our lower bounds are for worst-case assignment of input functions to players in G and show a very specific class of assignments that achieves the required lower bound.

4.2.2 $\mathcal{H}$ is a forest. We prove the following lemma.

Lemma 4.6. *When $\mathcal{H}$ is a forest, we have*

$$\text{TRIBES}_{\frac{y(\mathcal{H})}{2},N} \leq \text{BCQ}_{\mathcal{H},N}.$$

Proof. For notational simplicity, define $y = y(\mathcal{H})$. Given $\mathcal{H}$ and a $\text{TRIBES}_{\frac{y}{2},N}$ instance we design a corresponding $\text{BCQ}_{\mathcal{H},N}$ instance. As $\mathcal{H}$ is bipartite, let (L, R) be the node partition of $\mathcal{H}$ and consider the set O_L (O_R resp.) consisting of all nodes of degree at least two included in L (R resp.). Let O equal the largest of O_L and O_R (i.e., O consists of nodes of odd or even distance from the roots of the forest). Note that $|O| \geq \frac{y}{2}$,[17] and assume w.l.o.g. that the size of O is exactly $\frac{y}{2}$ (otherwise we take a subset of O). We associate a pair of sets (S_o, T_o) from $\text{TRIBES}_{\frac{y}{2},N}$ with each node $o \in O$, such that

$$\text{TRIBES}_{\frac{y}{2},N}(\hat{S}, \hat{T}) = \bigwedge_{o \in O} \text{DISJ}_N(S_o, T_o), \tag{3}$$

where $\text{DISJ}_N(S_o, T_o) = 1$ if $S_o \cap T_o \neq \emptyset$ and 0 otherwise.

We now construct a corresponding $\text{BCQ}_{\mathcal{H},N}$ instance in detail. We start by defining a pair of relations corresponding to each pair (S_o, T_o). Let $o \in O$. If o has a parent in $\mathcal{H}$, let o_p be its parent. Let o_c be a child of o. We consider the relations $R_{S_o} = S_o \times \{1\}$ and $R_{T_o} = T_o \times \{1\}$, where the attribute set of R_{S_o} is (o, o_c) and that of R_{T_o} is (o, o_p). Here we treat S_o and T_o as subsets of $[N]$ (instead of elements in $\{0,1\}^N$). In the case that o does not have a parent node, it is a root in $\mathcal{H}$ with at least two children, and thus we can set o_p to be a child of o that differs from o_c. Thus, $\text{TRIBES}_{\frac{y}{2},N}(\hat{S}, \hat{T}) = 1$ iff for each $o \in O$, the join $R_{S_o} \bowtie R_{T_o}$ is not empty. To complete the description of the BCQ instance, for each $o \in O$, we associate all additional edges (o, v) adjacent to o in $\mathcal{H}$ with the relation $[N] \times \{1\}$ on attributes (o, v); and remaining edges (u, v) that are not adjacent to any $o \in O$ with the relation $\{1\} \times \{1\}$. Note that no two vertices $o_1, o_2 \in O$ are adjacent in $\mathcal{H}$. Let us denote the BCQ instance constructed above by $q_{\mathcal{H},\hat{S},\hat{T}}$.

[17] Note that in the arity two case, it is easy to construct a GYO-GHD with y internal nodes using the structure of $\mathcal{H}$. Details in the full paper [38].

To complete the proof, we show that $q_{\mathcal{H},\hat{S},\hat{T}} = 1$ iff $\text{TRIBES}_{\frac{y}{2},N}(\hat{S}, \hat{T}) = 1$. If $q_{\mathcal{H},\hat{S},\hat{T}} = 1$ then there exists a tuple $\mathbf{t} \in \prod_{v \in V(\mathcal{H})} \text{Dom}(v)$ that satisfies all relations in $q_{\mathcal{H},\hat{S},\hat{T}}$, i.e. $\mathbf{t}_e \in R_e$ for every $e \in \overline{\mathcal{E}}$. Specifically, for each $o \in O$, $R_{S_o} \bowtie R_{T_o}$ is not empty which implies that $\text{TRIBES}_{\frac{y}{2},N}(\hat{S}, \hat{T}) = 1$. Alternatively, if $\text{TRIBES}_{\frac{y}{2},N}(\hat{S}, \hat{T}) = 1$, we can find a tuple $\mathbf{t} \in \prod_{v \in V(\mathcal{H})} \text{Dom}(v)$ that satisfies all relations in $q_{\mathcal{H},\hat{S},\hat{T}}$. For each $o \in O$ we set $\pi_o(\mathbf{t})$ to be any element in the intersection of S_o and T_o, and for all remaining nodes v we set $\pi_v(\mathbf{t}) = 1$. It holds that the relations corresponding to edges of the form (o, o_p), (o, o_c), (o, v), and (u, v) described above are all satisfied. This concludes our proof. □

Note that the above argument was independent of G. We now use the structure of G to obtain a lower bound on $\mathcal{R}(\text{BCQ}_{\mathcal{H},N}, G, K)$ using known results for $\text{TRIBES}_{\frac{y}{2},N}$.

Lower bounds dependent on G. We show the following lower bound for arbitrary G, assuming worst-case assignment of relations to players in K.

Lemma 4.7. *For any topology G and $\mathcal{H}$ being a forest,*

$$\mathcal{R}(\text{BCQ}_{\mathcal{H},N}, G, K) \geq \widetilde{\Omega}\left(\frac{y(\mathcal{H}) \cdot N}{\text{MinCut}(G, K)}\right).$$

Proof. We first consider a min-cut (A, B) of G that separates K, where A and B denote the set of vertices in each partition ($A \cup B = V$). Using the notation given in the proof of Lemma 4.6, let $q_{\mathcal{H},\hat{S},\hat{T}}$ be the query corresponding to a given instance $\text{TRIBES}_{\frac{y}{2},N}(\hat{S}, \hat{T})$. We assign relations $\{R_{S_o}\}_{o \in O}$ to vertices in A and relations $\{R_{T_o}\}_{o \in O}$ to vertices in B. The other relations in $q_{\mathcal{H},\hat{S},\hat{T}}$ can be assigned arbitrarily. Note that any protocol to compute $q_{\mathcal{H},\hat{S},\hat{T}}$ on G gives a two-party protocol (Alice, Bob) for $\text{TRIBES}_{\frac{y}{2},N}$. In particular, Alice gets the sets $\{S_o\}_{o \in O}$ (corresponding to R_{S_o}) assigned to vertices in A and Bob gets the sets $\{T_o\}_{o \in O}$ (corresponding to R_{T_o}) assigned to vertices in B (ignoring the additional relations). It follows that if there exists a $\mathcal{R}(\text{BCQ}_{\mathcal{H},N}, G, K)$ round protocol on G, then we have a two-party protocol (i.e., on a graph $\mathcal{G} = (\{a, b\}, (a, b))$) with at most $\mathcal{R}(\text{BCQ}_{\mathcal{H},N}, G, K) \cdot \text{MinCut}(G, K) \cdot \lceil \log_2(\text{MinCut}(G, K)) \rceil$ rounds. Indeed, we can simulate the two-party protocol on G across the cut (A, B), where Alice is responsible for A and Bob for B. In particular, if Alice needs to send a message to Bob (or vice-versa), it will be sent across edges crossing the cut. Note that in each round, at most $\text{MinCut}(G, K) \lceil \log_2(\text{MinCut}(G, K)) \rceil$ bits will be exchanged between Alice and Bob. We need $\lceil \log_2(\text{MinCut}(G, K)) \rceil$ bits in order to know the edge on which the message was sent. We can now invoke (1) to obtain a lower bound of

$$\mathcal{R}(\text{BCQ}_{\mathcal{H},N}, G, K) \geq \widetilde{\Omega}\left(\frac{y(\mathcal{H}) \cdot N}{\text{MinCut}(G, K)}\right).$$

□

4.2.3 General $\mathcal{H}$. We are now ready to prove our general lower bound for all simple graphs $\mathcal{H}$.

Theorem 4.8. *For arbitrary $G, K \subseteq V$, and graph $\mathcal{H}$, we have*

$$\mathcal{R}(\text{BCQ}_{\mathcal{H},N}, G, K) \geq \widetilde{\Omega}\left(\frac{(y(\mathcal{H}) + n_2(\mathcal{H})) \cdot N}{\text{MinCut}(G,K)}\right).$$

Proof Sketch. We present a proof sketch here. For notational convenience, define $y = y(\mathcal{H})$ and $n_2 = n_2(\mathcal{H})$. Let $m = \max\left(\frac{y}{2}, \frac{n_2}{2\log(n_2)}\right)$. In general, as in the proof of Lemma 4.6, given $\mathcal{H}$ and a TRIBES instance $\text{TRIBES}_{m,N}(\hat{S}, \hat{T})$ we construct a BCQ instance $q_{\mathcal{H},\hat{S},\hat{T}}$ such that $q_{\mathcal{H},\hat{S},\hat{T}} = 1$ iff $\text{TRIBES}_{m,N}(\hat{S}, \hat{T}) = 1$. To this end we need to "embed" the m pairs of sets (S_i, T_i) from $\text{TRIBES}_{m,N}(\hat{S}, \hat{T})$ as relations in $q_{\mathcal{H},\hat{S},\hat{T}}$. For $m = \frac{y}{2}$, we embed the pairs (S_i, T_i) in the forest $\text{F}(\mathcal{H})$ as done in Lemma 4.6. For $m = \frac{n_2}{2\cdot\log(n_2)}$, we consider $\text{C}(\mathcal{H})$. We then show that it must be the case that $\text{C}(\mathcal{H})$ either includes $\left(\frac{n_2}{2\log(n_2)}\right)$ vertex-disjoint cycles (or) it has an independent set of size $\Omega(n_2)$. In both cases, we show how one can embed $\frac{n_2}{2\log(n_2)}$ pairs (S_i, T_i) of $\text{TRIBES}_{m,N}(\hat{S}, \hat{T})$ in $C(\mathcal{H})$. We defer the proof to the full paper [38].

Assuming the above embeddings, we conclude that $q_{\mathcal{H},\hat{S},\hat{T}} = 1$ iff $\text{TRIBES}_{m,N}(\hat{S}, \hat{T}) = 1$, where $m = \max\left(\frac{y}{2}, \frac{n_2}{2\log(n_2)}\right)$. Since sum and max are within a factor 2 of each other, we can write $m \geq \frac{y}{4} + \frac{n_2}{4\log(n_2)}$. We can now apply ideas from the proof of Lemma 4.7 to obtain the required lower bound $\widetilde{\Omega}\left(\frac{(y+n_2)\cdot N}{\text{MinCut}(G,K)}\right)$. □

Note that in Theorem 4.1, the upper bound follows from Lemma 4.5 and the lower bound from Theorem 4.8. We conclude this section by noting that when $N \geq |V|^2$, our upper and lower bounds differ by $\widetilde{O}(d)$ factor (for worst-case assignments of relations to players). In particular, Theorem 3.10 implies that the first two terms in the upper and lower bounds match up to an $\widetilde{O}(1)$ factor. In the full paper [38], we show that for worst-case assignment of relations, the second terms in the upper and lower bounds also differ by a $\widetilde{O}(d)$ factor, as desired.

5 HYPERGRAPHS $\mathcal{H}$ AND GENERAL FAQ

Our results generalize fairly seamlessly to hypergraphs $\mathcal{H}$. For constant d, r, our upper and lower bounds match. However, for non-constant d, we have a gap of $\widetilde{O}(d^2 \cdot r^2)$, which is worse than our gap of $\widetilde{O}(d)$ for the arity two case. Details are deferred to the full paper [38].

We extend our results from BCQ to the general FAQ problem. We define the general FAQ problem here, which is a generalization of FAQ-SS. We are given a multi-hypergraph $\mathcal{H} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ where for each hyperedge $e \in \overline{\mathcal{E}}$, we also have an input function $f_e : \prod_{v\in e} \text{Dom}(v) \to \mathbb{D}$. In addition, we are given a set of *free variables* $\mathcal{F} \subseteq \overline{\mathcal{V}} : |\mathcal{F}| = \ell$ and[18] we would like to compute the function:

$$\phi\left(\mathbf{x}_{[\ell]}\right) = \bigoplus_{x_{\ell+1}\in\text{Dom}(x_{\ell+1})}^{(\ell+1)} \cdots \bigoplus_{x_n\in\text{Dom}(x_n)}^{(n)} \bigotimes_{S\in\overline{\mathcal{E}}} f_S(\mathbf{x}_S), \quad (4)$$

where $\mathbf{x} = (x_u)_{u\in\overline{\mathcal{V}}}$ and $\mathbf{x}_S$ is $\mathbf{x}$ projected down to co-ordinates in $S \subseteq \overline{\mathcal{V}}$. The variables in $\overline{\mathcal{V}} \setminus \mathcal{F}$ are called *bound variables*. For every bound variable $i > \ell$, $\oplus^{(i)}$ is a binary (aggregate) operator on the domain $\mathbb{D}$. Different bound variables may have different aggregates. Finally, for each bound variable $i > \ell$ either $\oplus^{(i)} = \otimes$ (*product aggregate*) or $(\mathbb{D}, \oplus^{(i)}, \otimes)$ forms a commutative semiring (*semiring aggregate*) with the same additive identity $\mathbf{0}$ and multiplicative identity $\mathbf{1}$. As with FAQ-SS, we assume that the functions are input in the *listing* representation, i.e. the function f_e is represented as a list of its non-zero values: $R_e = \{(\mathbf{y}, f_e(\mathbf{y}))|\mathbf{y} \in \prod_{v\in e} Dom(v) : f_e(\mathbf{y}) \neq \mathbf{0}\}$. Note that when $\oplus^{(i)} = \oplus$ is the same semiring aggregate for every $\ell < i \leq n$, we have the FAQ-SS problem.

For any $\mathbb{D}$, let $\text{FAQ}_{\mathbb{D},\mathcal{H},N,\mathcal{F}}$ denote the class of FAQ problems, where each function in $\mathcal{H}$ has at most N non-zero entries. (Note that we are not explicitly stating the operators for the bound variables $(\oplus^{(\ell+1)}, \ldots, \oplus^{(n)})$ since our upper and lower bounds hold for all such operators.) Let $\mathcal{R}(\text{FAQ}_{\mathbb{D},\mathcal{H},N,\mathcal{F}}, G, K)$ denote the minimum worst-case number of rounds needed by a randomized protocol with error at most $\frac{1}{3}$ that computes any query in $\text{FAQ}_{\mathbb{D},\mathcal{H},N,\mathcal{F}}$ on G with functions assigned to nodes in K. We prove the following results.

Theorem 5.1. *For $O(1)$-degenerate hypergraphs $\mathcal{H}$ with $O(1)$-arity, we have*

$$\mathcal{R}(\text{FAQ}_{\mathbb{D},\mathcal{H},N,\mathcal{F}}, G, K) = \widetilde{\Theta}\left(\frac{(y(\mathcal{H}) + n_2(\mathcal{H})) \cdot N}{\text{MinCut}(G,K)}\right)$$

for any $\mathbb{D}$, specific choices of $\mathcal{F}$, arbitrary G and K. When G is a line (as in the first row in Table 1), $\text{MinCut}(G,K) = 1$.

Theorem 5.2. *For general degenerate hypergraphs $\mathcal{H}$ with arity at most r, we have*

$$\begin{aligned}&\mathcal{D}\left(\text{FAQ}_{\mathbb{D},\mathcal{H},N,\mathcal{F}}, G, K\right)\\ &= O\left(y(\mathcal{H}) \cdot \min_{\Delta\in[|V|]}\left(\frac{N\cdot r}{\text{ST}(G,K,\Delta)} + \Delta\right)\right)\\ &+ O\left(\tau_{\text{MCF}}(G, K, n_2(\mathcal{H}) \cdot d \cdot r \cdot N)\right)\end{aligned}$$

and

$$\mathcal{R}\left(\text{FAQ}_{\mathbb{D},\mathcal{H},N,\mathcal{F}}, G, K\right) \geq \widetilde{\Omega}\left(\frac{(d\cdot y(\mathcal{H}) + n_2(\mathcal{H})) \cdot N}{d \cdot r \cdot \text{MinCut}(G,K)}\right).$$

[18] For a fixed $\mathcal{F}$, the vertices in $\overline{\mathcal{V}}$ can be renumbered so that $\mathcal{F} = [\ell]$ w.l.o.g.

The lower bound differs from the upper bound by a factor of $O(r^2d^2)$ in the worst case.

We would like to mention here once again that our upper bound is a deterministic protocol and the lower bound is for randomized protocols. Details are in the full paper [38].

6 MATRIX CHAIN MULTIPLICATION

We consider the following FAQ-SS problem. The network topology has $k+2$ players $P_0, \ldots, P_{k+1}$ such that (P_i, P_{i+1}) is an edge (i.e. G is a line) where P_0 receives $\mathbf{x} \in \mathbb{F}_2^N$ and P_i for $i \in [k]$ receives $\mathbf{A}_i \in \mathbb{F}_2^{N\times N}$. Player P_{k+1} wants to compute $\mathbf{A}_k \cdot \mathbf{A}_{k-1} \cdots \mathbf{A}_1 \cdot \mathbf{x}$. Alternatively, for every $i \in [k]$, define $\mathbf{y}_i = \mathbf{A}_i \cdot \mathbf{y}_{i-1}$, with $\mathbf{y}_0 = \mathbf{x}$. Note that we want to compute $\mathbf{y}_k$. Note that this is an FAQ-SS problem since we can re-write the above as

$$\phi(z_k) = \sum_{(z_i)_{i=0}^{k-1} \in [N]^k} \left(\prod_{j=1}^{k} A_j(z_j, z_{j-1}) \right) X(z_0), \tag{5}$$

where the functions satisfy $A_j(x,y) = \mathbf{A}_j[x,y]$ and $X(z) = \mathbf{x}[z]$ for every triple of indices $x, y, z \in [N]$. [19]

We note that this problem can be solved in $O(kN)$ rounds.[20]

Proposition 6.1. *The* FAQ-SS *problem from* (5) *can be computed in $O(kN)$ rounds.*

We prove this proposition in the full paper [38]. We remark that when k is large, a bottom-to-top fashion merge algorithm can achieve $O(N^2 \log k + k)$ rounds. (See the full paper [38] for details.) In the next section, we prove a tight lower bound of $\Omega(kN)$ for the case $k \le N$.

6.1 The Lower Bound

We will argue that the upper bound of $O(kN)$ rounds in Proposition 6.1 is tight if $k \le N$. Before we do that we collect some definitions and results related to the min-entropy of a random variable.

6.1.1 Background. The *min-entropy* of a random variable X is defined as $H_\infty(X) := -\log \max_{x \in \text{supp}(X)} \Pr[X = x]$. For a random variable X and an event $\mathcal{E}$ that is possibly correlated with X, define $H_\infty(X\mathcal{E}) = -\log \max_{x \in \text{supp}(X)} \Pr[X = x, \mathcal{E}]$. Notice that in the above definition, we do not "normalize" $\Pr[X = x, \mathcal{E}]$ by a factor of $\Pr[\mathcal{E}]$.

[19] Note that N here is the dimension of the matrices as opposed to the number of non-zero entries used in the previous sections.

[20] Note that the trivial algorithm takes $\Omega(kN^2)$ rounds.

For random variables X and Y, the conditional smooth min-entropy $H_\infty^\epsilon(X|Y)$ is defined as

$$H_\infty^\epsilon(X|Y) = \sup_{\mathcal{E}} \min_{y \in \text{supp}(Y)} H_\infty(X\mathcal{E}|Y = y)$$
$$= \sup_{\mathcal{E}} \left(-\log \max_{(x,y) \in \text{supp}(X,Y)} \Pr[\mathcal{E}, X = x | Y = y] \right)$$

where the quantification over $\mathcal{E}$ is over all events $\mathcal{E}$ (which can be correlated with X and Y) with $\Pr(\mathcal{E}) \ge 1 - \epsilon$. When Y is a deterministic variable (in other words, we are not conditioning on any randomized variable), then we simply use $H_\infty^\epsilon(X)$:

$$H_\infty^\epsilon(X) = \sup_{\mathcal{E}} H_\infty(X\mathcal{E}), \tag{6}$$

where again the quantification over $\mathcal{E}$ is over all events $\mathcal{E}$ with $\Pr(\mathcal{E}) \ge 1 - \epsilon$.

The following lemma will be useful in our analysis:

Lemma 6.2 (Lemma 4 and Lemma 7 of [44]). *Let Y be a random variable with support size at most 2^ℓ. Then we have for any $\epsilon \ge 0, \epsilon' > 0$ and random variable X, that $H_\infty^{\epsilon+\epsilon'}(X|Y) \ge H_\infty^\epsilon(X) - \ell - \log(1/\epsilon')$.*

Finally, we will use the following result where $h(p) = -p\log_2 p - (1-p)\log_2(1-p)$:

Theorem 6.3. *Let the constant $\gamma > 0$ be small enough. Let $\mathbf{x} \in \mathbb{F}_2^N$, $\mathbf{A} \in \mathbb{F}_2^{N\times N}$ and $\mathbf{Y}$ be random variables such that for every $y \in \text{supp}(\mathbf{Y})$, $\mathbf{x}$ and $\mathbf{A}$ are independent conditioned on $\mathbf{Y} = y$. Moreover for some reals $\epsilon_1, \epsilon_2 \ge 0$, we have*

$$H_\infty^{\epsilon_1+\epsilon_2+2^{-\Omega(\gamma N)}}(\mathbf{Ax}|\mathbf{Y}) \ge \left(1 - \sqrt{2\gamma}\right) \cdot N,$$

where $\alpha \stackrel{def}{=} 3\gamma + \sqrt{2\gamma} + h(\sqrt{2\gamma})$. Then, we have

$$H_\infty^{\epsilon_1+\epsilon_2+2^{-\Omega(\gamma N)}}(\mathbf{Ax}|\mathbf{Y}) \ge \left(1 - \sqrt{2\gamma}\right) \cdot N.$$

The proof of Theorem 6.3 follows from known results in pseudorandomness [21, 52] and appears in the full paper [38].

6.1.2 Showing Proposition 6.1 is tight for $k \le N$. At a high level, we will prove by induction that for player P_i at time about γiN, the min-entropy of $\mathbf{y}_{i-1}$ is at least $\alpha \cdot N$ (and the situation at P_{i+1} should be similar). Since by this time P_{i+1} would have received at most $O(\gamma iN) \le O(\gamma N^2)$ bits, this means $\mathbf{A}_i$ has min-entropy at least $(1-\gamma)N^2$. Thus, we can apply Theorem 6.3 to argue that at P_{i+1} the min-entropy of $\mathbf{y}_i = \mathbf{A}_i \cdot \mathbf{y}_{i-1}$ is large. To finish the inductive argument we have to wait for γN more steps but by Lemma 6.2, even then $\mathbf{y}_i$ will still have high enough min-entropy. It is natural to wonder if we can make the same argument using Shannon entropy instead of min-entropy. In the full paper [38], we show that this is not possible.

We define some useful notations before we prove the lower bound. At any given time t, let $\mathbf{m}^i(t)$ denote the transcript of messages exchanged on the link between P_{i-1} and P_i till time

t. For $i \in [k+1]$, define $t_i = \frac{\gamma}{4} \cdot iN$, and $\widetilde{\mathbf{m}}^i = \mathbf{m}^i(t_i)$. For a random variable $\mathbf{m}$, we will use m to denote a specific value of the random variable $\mathbf{m}$. In addition, we use $\widetilde{\mathbf{m}}^{[i]}$ and $\widetilde{m}^{[i]}$ to denote the tuples $(\widetilde{\mathbf{m}}^1, \widetilde{\mathbf{m}}^2, \cdots, \widetilde{\mathbf{m}}^i)$ and $(\widetilde{m}^1, \widetilde{m}^2, \cdots, \widetilde{m}^i)$ respectively.

Let $\epsilon^* = 2^{-\Omega(\gamma N)}$ be at least thrice the maximum of $2^{-\gamma N/4}$ and the $2^{-\Omega(\gamma N)}$ term in Theorem 6.3. We will argue:

Lemma 6.4. *Let $\mathbf{A}_i$ for every $i \in [k]$ and $\mathbf{x}$ be all uniformly and independently distributed. Let $\gamma > 0$ be such that*[21]

$$4\gamma + \sqrt{2\gamma} + h(\sqrt{2\gamma}) \le 1, \tag{7}$$

and $\gamma N/4$ is an integer. Then we have the following for every $i \in [k+1]$:

$$H_\infty^{i\epsilon^*}\left(\mathbf{y}_{i-1} | \widetilde{\mathbf{m}}^{[i]}\right) \ge N(1-\gamma-\sqrt{2\gamma}). \tag{8}$$

Proof. We will prove the claim by induction. For the base case of $i = 1$, Lemma 6.2 implies that (recall that $\widetilde{\mathbf{m}}^1 = \mathbf{m}^1(t_1) = \mathbf{m}^1(\gamma N/4)$ and $\mathbf{y}_0 = \mathbf{x}$): $H_\infty^{\epsilon^*}\left(\mathbf{y}_0 | \widetilde{\mathbf{m}}^1\right) \ge H_\infty(\mathbf{x}) - \gamma N/4 - \log(1/\epsilon^*) \ge N(1-\gamma/4-\gamma/2) \ge N(1-\gamma-\sqrt{2\gamma})$. Thus, (8) holds for $i = 1$.

We assume (8) holds for some $i \ge 1$; we prove that it also holds with i replaced by $i+1$. For any interval $[\ell, r]$ we use $\mathbf{A}_{[\ell:r]}$ to denote the tuple $(\mathbf{A}_\ell, \ldots, \mathbf{A}_r)$. Conditioned on $\widetilde{\mathbf{m}}^i = \widetilde{m}^i$, since all communication between $P_1, \ldots, P_{i-1}$ and $P_i, \ldots, P_{k+1}$ are independent, we have

(B1) $(\mathbf{x}, \mathbf{A}_{[1:i-1]})$ is independent of $\mathbf{A}_{[i,k]}$.
(B2) $\mathbf{y}_{i-1}$ and $\widetilde{\mathbf{m}}^{[i-1]}$ are determined by $(\mathbf{x}, \mathbf{A}_{[1:i-1]})$.
(B3) $\mathbf{m}^{i+1}(t_i)$ is determined by $\mathbf{A}_{[i:k]}$.

The above properties imply the following, which will be used many times in our analysis:

(C) Conditioned on $\widetilde{\mathbf{m}}^i = \widetilde{m}^i$, $(\mathbf{x}, \mathbf{A}_{[1:i-1]}, \mathbf{y}_{i-1}, \widetilde{\mathbf{m}}^{[i-1]})$ and $(\mathbf{A}_{[i,k]}, \mathbf{m}^{i+1}(t_i))$ are independent.

By (C), $\mathbf{y}_{i-1}\left|\left(\widetilde{\mathbf{m}}^{[i]}, \mathbf{m}^{i+1}(t_i)\right)\right. = \left(\widetilde{m}^{[i]}, m^{i+1}(t_i)\right)$ has the same distribution as $\mathbf{y}_{i-1}\left|\widetilde{\mathbf{m}}^{[i]} = \widetilde{m}^{[i]}\right.$. By the inductive hypothesis, we have

$$H_\infty^{i\epsilon^*}\left(\mathbf{y}_{i-1}\left|\left(\widetilde{\mathbf{m}}^{[i]}, \mathbf{m}^{i+1}(t_i)\right)\right.\right) = H_\infty^{i\epsilon^*}\left(\mathbf{y}_{i-1}\left|\widetilde{\mathbf{m}}^{[i]}\right.\right) \ge N(1-\gamma-\sqrt{2\gamma}). \tag{9}$$

We show in the full paper [38] that equality can be achieved in the above lemma. Further, Lemma 6.2 (with $\epsilon = 0$ and $\epsilon' = \epsilon^*/3$) implies that

$$H_\infty^{\epsilon^*/3}\left(\mathbf{A}_i\left|\left(\widetilde{\mathbf{m}}^i, \mathbf{m}^{i+1}(t_i)\right)\right.\right) \ge N^2 - 2i \cdot \frac{\gamma}{4} \cdot N - \log(\epsilon^*/3) \ge N^2(1-\gamma). \tag{10}$$

Again by (C), $\mathbf{A}_i\left|\left(\widetilde{\mathbf{m}}^i, \mathbf{m}^{i+1}(t_i)\right)\right. = \left(\widetilde{m}^i, m^{i+1}(t_i)\right)$ has the same distribution as $\mathbf{A}_i\left|\left(\widetilde{\mathbf{m}}^{[i]}, \mathbf{m}^{i+1}(t_i)\right)\right. = \left(\widetilde{m}^{[i]}, m^{i+1}(t_i)\right)$. Equality in (9) and (10) imply that

$$H_\infty^{\epsilon^*/3}\left(\mathbf{A}_i\left|\left(\widetilde{\mathbf{m}}^{[i]}, \mathbf{m}^{i+1}(t_i)\right)\right.\right) \ge N^2(1-\gamma). \tag{11}$$

By (9), (11), and (by (C)) the fact that $\mathbf{A}_i$ and $\mathbf{y}_{i-1}$ are independent conditioned on $(\widetilde{\mathbf{m}}^{[i]}, \mathbf{m}^{i+1}(t_i)) = (\widetilde{m}^{[i]}, m^{i+1}(t_i))$, we have the following via Theorem 6.3:

$$H_\infty^{(i+2/3)\epsilon^*}\left(\mathbf{y}_i = \mathbf{A}_i\mathbf{y}_{i-1}\left|\left(\widetilde{\mathbf{m}}^{[i]}, \mathbf{m}^{i+1}(t_i)\right)\right.\right) \ge N(1-\sqrt{2\gamma}),$$

as long as $1-\gamma-\sqrt{2\gamma} \ge 3\gamma + \sqrt{2\gamma} + h(\sqrt{2\gamma})$, which follows from (7). By applying Lemma 6.2 again (with $\epsilon = (i+2/3)\epsilon^*$ and $\epsilon' = \epsilon^*/3$), we get that[22]:

$$\begin{aligned} &H_\infty^{(i+1)\epsilon^*}\left(\mathbf{y}_i\left|\widetilde{\mathbf{m}}^{[i+1]}\right.\right) \\ &\ge H_\infty^{(i+2/3)\epsilon^*}\left(\mathbf{y}_i\left|\left(\widetilde{\mathbf{m}}^{[i]}, \mathbf{m}^{i+1}(t_i)\right)\right.\right) - N\gamma/4 - \log(\epsilon^*/3) \\ &\ge N(1-\sqrt{2\gamma} - \frac{\gamma}{4} - \frac{\gamma}{2}) \ge N(1-\gamma-\sqrt{2\gamma}), \end{aligned}$$

as desired. □

The above immediately gives us our lower bound (the proof is deferred to the full paper [38]):

Theorem 6.5. *Any protocol that solves the* FAQ-SS *problem from* (5) *with $k \le N$ and large enough N, with success probability at least* 1/2, *takes $\Omega(kN)$ rounds.*

We will need the following result (the proof appears in the full paper [38]) in our proof.

Lemma 6.6. *Assume $H_\infty^\epsilon(X|Y) \ge L$. Then for every function $f : \mathrm{supp}(Y) \to \mathrm{supp}(X)$, we have $\Pr[f(Y) = X] \le \epsilon + 2^{-L}$.*

We are now ready to prove Theorem 6.5.

Proof of Theorem 6.5. Let Π be any protocol with at most $t_{k+1} = \gamma(k+1)N/4$ rounds. Lemma 6.4 implies that at the end of the protocol, we have

$$H_\infty^{(k+1)\epsilon^*}\left(\mathbf{y}_k\left|\widetilde{\mathbf{m}}^{[k+1]}\right.\right) \ge N(1-\gamma-\sqrt{2\gamma}).$$

This implies that even if the player $k+1$ is given $\widetilde{\mathbf{m}}^{[k+1]}$ (instead of only $\widetilde{\mathbf{m}}^{k+1} = \mathbf{m}^{k+1}(t_{k+1})$), it can only output the correct answer with probability at most

$$(k+1)\epsilon^* + 2^{-N(1-\gamma-\sqrt{2\gamma})},$$

by Lemma 6.6 (here $f(Y)$ is the output at P_{k+1} for $Y = \widetilde{\mathbf{m}}^{[k+1]}$ and $X = \mathbf{y}_k$). For large enough N, the above quantity is less than 1/2. □

7 RELATED WORK

We now survey the most closely related work. Due to lack of space, a detailed discussion is deferred to the full paper [38].

[21] There exists a value $\gamma \ge 0.01$ (for large enough N) that satisfies the required conditions.

[22] Note that we are not conditioning on $\mathbf{m}^{i+1}(t_{i+1}) = \mathbf{m}^{i+1}(t_i + \gamma N/4)$ instead of the earlier $\mathbf{m}^{i+1}(t_i)$.

Parallel Database Query Computation. The MPC model has seen a lot of research activity in the last few years [2, 9, 10, 31, 36, 37]. We compared these models with ours in Section 1.1 and Appendix A.

Widths of GHDs. The Internal Node Width $y(\mathcal{H})$ of a GHD focuses on minimizing the number of internal (non-leaf) nodes in GHDs of acyclic hypergraphs. There is a related notion for Tree Decompositions called *Lean* Tree Decompositions (LTDs) [11, 22, 47]. The LTDs minimize the number of internal nodes in the following way – they try to retain only pairs of connected internal nodes whose intersection forms a bridge in the original graph $\mathcal{H}$. The other nodes are forced to become leaves of one of the internal nodes. While our construction procedure of GYO-GHDs tries to convert existing internal nodes to become leaves of some internal nodes, we do not (yet) see an exact one-to-one mapping from GYO-GHDs to LTDs. We would like to mention here that both the goals of GYO-GHDs and LTDs are the same i.e., to minimize the number of internal nodes. We would like to note here that $y(\mathcal{H})$ can potentially reduce the depth of the GHD as well. Reducing depth of GHDs (sometimes by increasing the treewidth) has been considered before [2, 4, 12].

For GHDs, the problem of computing GHDs that minimize certain cost functions of the HDs are studied in the framework of Weighted GHDs [29, 45]. For a given hypergraph $\mathcal{H}$, one way to map our notion of width to their setting is to consider a vertex aggregating function on every candidate HD $\mathcal{T}$ for $\mathcal{H}$. In particular, we can write

$$\Lambda_H^{f'}(\mathcal{T}) = \sum_{v' \in V(\mathcal{T})} f'_{\mathcal{H}}(v'), \tag{12}$$

where $f'_{\mathcal{H}} = 1$ if v' is a internal node and 0 otherwise. It follows that $f'_{\mathcal{H}}$ can be computed in linear time in size of $\mathcal{T}$. Given this setup, Theorem 3.4 in [29] proves that computing Minimal GHDs over HDs for arbitrary vertex aggregation functions is NP-Hard.

However, this does not hold in our case since there is always a GHD with one internal node (containing all the variables in $\mathcal{H}$). As a result, considering the minimization over all GHDs for our case is trivial and doesn't give use tight results. In particular, our setup is a bit different and we minimize over GYO-GHDs (Construction 2.8). For the tightness of our bounds for $\mathcal{H}$ with constant degeneracy and constant arity, we only need an $O(1)$-factor approximation of Internal-Node-Width, which we achieve (details in the full paper [38]).

We refer the reader to [27] for a recent survey on widths for GHD.

Distributed Computing and Communication Complexity. As stated earlier, our model is similar to (and different from) the CONGEST model in distributed computing [42]. Recently, there has been work on the same model as ours but instead of minimizing the number of rounds, they focus on minimizing the total communication of the protocols [14, 19, 20, 43, 48, 50]. Finally, [18] obtained results on minimizing the number of rounds of protocols in our setup for some well-studied functions in two-party communication complexity literature.

Entropy in Communication Complexity. Information complexity by now is a well-established sub-field of communication complexity that uses Shannon entropy to measure the amount of information exchanges in a two-party communication protocol. Information complexity was introduced in the work of Chakrabarti et al. [15] and later used in a systematic way to tackle multiple problems in [8]. To the best of our knowledge, min-entropy has only been used very recently in communication complexity [25, 26] though it has found numerous applications in pseudorandomness and cryptography for at least two decades [49]. Our work adds to the recently growing body of work that uses min-entropy to prove communication complexity results [17].

8 FUTURE WORK

We leave the following questions as future work: handling node failures, finding optimal assignments of functions to players in G, identifying the optimal topology for a given query and finally, closing the gap between our upper and lower bounds for d-degenerate graphs for super-constant d. We cannot (yet) handle the condition that N has to be larger than the size of G. We address the assumptions on the knowledge of q and G and the assumption that the input functions are completely assigned to players in G in the full paper [38].

ACKNOWLEDGMENTS

We thank the anonymous reviewers of PODS'19 for their helpful comments that greatly improved the presentation of the paper. We are greatly indebted to Arkadev Chattopadhyay and David Zuckerman for their insights that led to the results in Section 6. We thank Martin Grohe, Oliver Kennedy, Hung Ngo and Dan Suciu for helpful discussions at various stages of this work. This work was supported by NSF grant CCF-1717134.

REFERENCES

[1] Abu-Elkheir, M., Hayajneh, M., and Ali, N. A. Data management for the internet of things: Design primitives and solution. *Sensors 13*, 11 (2013), 15582–15612.

[2] Afrati, F. N., Joglekar, M. R., Ré, C., Salihoglu, S., and Ullman, J. D. GYM: A multiround distributed join algorithm. In *ICDT* (2017), pp. 4:1–4:18.

[3] Aji, S. M., and McEliece, R. J. The generalized distributive law. *IEEE Transactions on Information Theory 46*, 2 (Mar 2000), 325–343.

[4] Akatov, D. *Exploiting parallelism in decomposition methods for constraint satisfaction.* PhD thesis, University of Oxford, UK, 2010.

[5] Alon, N., Hoory, S., and Linial, N. The moore bound for irregular graphs. *Graphs and Combinatorics 18*, 1 (Mar 2002), 53–57.

[6] Alon, N., and Spencer, J. *The Probabilistic Method.* John Wiley, 1992.

[7] Bakibayev, N., Kociský, T., Olteanu, D., and Zavodny, J. Aggregation and ordering in factorised databases. *PVLDB 6*, 14 (2013), 1990–2001.

[8] Bar-Yossef, Z., Jayram, T. S., Kumar, R., and Sivakumar, D. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci. 68*, 4 (2004), 702–732.

[9] Beame, P., Koutris, P., and Suciu, D. Communication steps for parallel query processing. In *PODS* (2013), pp. 273–284.

[10] Beame, P., Koutris, P., and Suciu, D. Skew in parallel query processing. In *PODS* (2014), pp. 212–223.

[11] Bellenbaum, P., and Diestel, R. Two short proofs concerning tree-decompositions. *Combinatorics, Probability & Computing 11*, 6 (2002), 541–547.

[12] Bodlaender, H. L. Nc-algorithms for graphs with small treewidth. In *Graph-Theoretic Concepts in Computer Science, 14th International Workshop, WG '88, Amsterdam, The Netherlands, June 15-17, 1988, Proceedings* (1988), pp. 1–10.

[13] Bonnet, P., Gehrke, J., and Seshadri, P. Towards sensor database systems. In *Mobile Data Management, Second International Conference, MDM 2001, Hong Kong, China, January 8-10, 2001, Proceedings* (2001), pp. 3–14.

[14] Braverman, M., Ellen, F., Oshman, R., Pitassi, T., and Vaikuntanathan, V. A tight bound for set disjointness in the message-passing model. In *FOCS* (2013), pp. 668–677.

[15] Chakrabarti, A., Shi, Y., Wirth, A., and Yao, A. C. Informational complexity and the direct sum problem for simultaneous message complexity. In *FOCS* (2001), pp. 270–278.

[16] Chattopadhyay, A. Personal communication, 2018.

[17] Chattopadhyay, A., Koucký, M., Loff, B., and Mukhopadhyay, S. Simulation beats richness: New data-structure lower bounds. In *STOC* (2018).

[18] Chattopadhyay, A., Langberg, M., Li, S., and Rudra, A. Tight network topology dependent bounds on rounds of communication. In *SODA* (2017), pp. 2524–2539.

[19] Chattopadhyay, A., Radhakrishnan, J., and Rudra, A. Topology matters in communication. In *FOCS* (2014), pp. 631–640.

[20] Chattopadhyay, A., and Rudra, A. The range of topological effects on communication. In *ICALP* (2015), pp. 540–551.

[21] Dodis, Y., and Oliveira, R. On extracting private randomness over a public channel. In *RANDOM* (2003), pp. 252–263.

[22] Erde, J. A unified treatment of linked and lean tree-decompositions. *J. Comb. Theory, Ser. B 130* (2018), 114–143.

[23] Gehrke, J., and Madden, S. Query processing in sensor networks. *IEEE Pervasive Computing 3*, 1 (2004), 46–55.

[24] Ghobadi, M., Mahajan, R., Phanishayee, A., Devanur, N. R., Kulkarni, J., Ranade, G., Blanche, P., Rastegarfar, H., Glick, M., and Kilper, D. C. ProjecToR: Agile reconfigurable data center interconnect. In *SIGCOMM* (2016), pp. 216–229.

[25] Göös, M., Lovett, S., Meka, R., Watson, T., and Zuckerman, D. Rectangles are nonnegative juntas. In *STOC* (2015), pp. 257–266.

[26] Göös, M., Pitassi, T., and Watson, T. Query-to-communication lifting for BPP. In *FOCS* (2017), pp. 132–143.

[27] Gottlob, G., Greco, G., Leone, N., and Scarcello, F. Hypertree decompositions: Questions and answers. In *PODS* (2016), pp. 57–74.

[28] Graham, M. H. On the universal relation. In *Tech Report* (1979).

[29] Greco, G., Leone, N., and Scarcello, F. On weighted hypertree decompositions. In *Proceedings of the Twelfth Italian Symposium on Advanced Database Systems, SEBD 2004, S. Margherita di Pula, Cagliari, Italy, June 21-23, 2004* (2004), pp. 54–61.

[30] Jayram, T. S., Kumar, R., and Sivakumar, D. Two applications of information complexity. In *STOC* (2003), pp. 673–682.

[31] Joglekar, M., and Ré, C. It's all a matter of degree: Using degree information to optimize multiway joins. In *ICDT* (2016), pp. 11:1–11:17.

[32] Khamis, M. A., Ngo, H. Q., and Rudra, A. FAQ: questions asked frequently. In *PODS* (2016), pp. 13–28.

[33] Khamis, M. A., Ngo, H. Q., and Rudra, A. Juggling functions inside a database. *SIGMOD Record 46*, 1 (2017), 6–13.

[34] Kossmann, D. The state of the art in distributed query processing. *ACM Comput. Surv. 32*, 4 (2000), 422–469.

[35] Kostochka, A. V. On almost (k-1)-degenerate (k+1)-chromatic graphs and hypergraphs. *Discrete Mathematics 313*, 4 (2013), 366–374.

[36] Koutris, P., Beame, P., and Suciu, D. Worst-case optimal algorithms for parallel query processing. In *ICDT* (2016), pp. 8:1–8:18.

[37] Koutris, P., and Suciu, D. A guide to formal analysis of join processing in massively parallel systems. *SIGMOD Record 45*, 4 (2016), 18–27.

[38] Langberg, M., Li, S., Jayaraman, S. V. M., and Rudra, A. Topology Dependent Bounds For FAQs. UB-CSE Tech Report 2019-01, 2019. https://cse.buffalo.edu/tech-reports/2019-01.pdf.

[39] Lau, L. C. An approximate max-steiner-tree-packing min-steiner-cut theorem*. *Combinatorica 27*, 1 (2007), 71–90.

[40] Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst. 30*, 1 (2005), 122–173.

[41] Olteanu, D., and Závodný, J. Size bounds for factorised representations of query results. *ACM Trans. Database Syst. 40*, 1 (2015), 2:1–2:44.

[42] Peleg, D. *Distributed Computing: A Locality-Sensitive Approach.*

[43] Phillips, J. M., Verbin, E., and Zhang, Q. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *SODA* (2012), pp. 486–501.

[44] Renner, R., and Wolf, S. Simple and tight bounds for information reconciliation and privacy amplification. In *ASIACRYPT* (Berlin, Heidelberg, 2005), Springer-Verlag, pp. 199–216.

[45] Scarcello, F., Greco, G., and Leone, N. Weighted hypertree decompositions and optimal query plans. In *PODS* (2004), pp. 210–221.

[46] Tarjan, R. E., and Yannakakis, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput. 13*, 3 (1984), 566–579.

[47] Thomas, R. A menger-like property of tree-width: The finite case. *J. Comb. Theory, Ser. B 48*, 1 (1990), 67–76.

[48] Tiwari, P. Lower bounds on communication complexity in distributed computer networks. *J. ACM 34*, 4 (1987), 921–938.

[49] Vadhan, S. P. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science 7*, 1-3 (2012), 1–336.

[50] Woodruff, D., and Zhang, Q. Tight bounds for distributed functional monitoring. In *STOC* (2012), pp. 941–960.

[51] Yu, C. T., and Ozsoyoglu, M. Z. An algorithm for tree-query membership of a distributed query. In *The IEEE Computer Society's Third International Computer Software and Applications Conference, COMPSAC 1979, 6-8 November, 1979, Chicago, Illinois, USA* (1979), pp. 306–312.

[52] Zuckerman, D. Simulating BPP using a general weak random source. *Algorithmica 16*, 4/5 (1996), 367–391.

A COMPARISON WITH RELEVANT MODELS

A.1 Basic MPC model

We formally define the MPC model used in [9] and the model adopted by [2] here, both in the language of our model

(Model 2.1). We consider the MPC with no replication, which is known as the basic MPC model in the literature.

Model A.1 (MPC(0)). *We are given a query q and its underlying hypergraph $\mathcal{H} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ with input functions f_e having at most N non-zero values for every $e \in \overline{\mathcal{E}}$. We consider the network topology G' with $p+k$ nodes, defined as follows. There are k nodes, each assigned a function f_e for every $e \in \overline{\mathcal{E}}$. We call this set K. There are no edges between any pair of nodes in K. All nodes in K are directly connected by an edge to every node in a clique with p nodes that are disjoint from K (also a part of G'). Each vnode in K has capacity N and all the remaining nodes have capacity L. The capacity of a node bounds the number of bits it can receive in each round. Given this setup, we would like to compute* BCQ *(and more generally an* FAQ*) of $\mathcal{H}$ on G'.*
To design a protocol for this computation, we can assume that every node in G' has the knowledge of q and G'. At the end of the protocol, a pre-determined player in K knows the answer to q.
Finally, given the above setup, our goal is to design protocols that minimize the number of rounds to compute q on G. This model does not take into account the internal computation done by the $p+k$ nodes and assumes the nodes co-operatively compute the answer to q.

We now summarize the differences of this model from ours.

A.1.1 Differences from our Model.

- MPC(0) assumes a specific choice of a network topology G' as opposed to general topology G in our model.
- MPC(0) assumes a specific assignment of functions in $\overline{\mathcal{E}}$ to players in G'. Our upper bound techniques can handle any assignment but our lower bounds are for a specific class of assignments. We would like to mention here that this is true for the models in [2, 9] as well (i.e., upper bounds can handly any assignment whereas lower bounds are for a specific class of assignments) and we consider one such assignment in Model A.1.
- MPC(0) assumes node capacities whereas ours assumes edge capacities.
- The models in [2, 9] design protocols wherein the number of rounds is either constant or a function of k. The number of rounds in our model are a function of N.
- The models in [2, 9] generally prove results for computing natural join whereas we look at BCQ (and more generally FAQs). We note that the results of [2, 9] for natural join apply to BCQ as well. This is true for both upper and lower bounds in [9] and only for upper bounds in [2].

We consider two instantations of this model – one by [9] and the other by [2].

A.1.2 Fixing p and Determining L [9]. This model assumes N is larger than the size of G' and all the functions f_e are matchings (i.e., skew-free). In other words, for each variable $v \in e$, each of the values $x_v \in \mathrm{Dom}(v)$ can occur in at most one tuple in f_e. Using Proposition 3.2 and Theorem 3.3 in [9], it can be shown that there exists an optimal one round protocol to solve BCQ of any star $\mathcal{H}$ on G' with $L = \Omega\left(\frac{k \cdot N}{p}\right)$. Further, when $\mathcal{H}$ is a forest, BCQ of $\mathcal{H}$ on G' takes $\Theta(\log(D'))$ rounds for the same L (where D' is the diameter of $\mathcal{H}$). We would like to mention here that a follow up work [10] handled input functions with specific types of skew and proved upper and lower bounds for the queries considered above. Since each node in the p-clique can have different capacities in this scenario, we do not discuss it further here.

A.1.3 Fixing L and Determining p [2]. This model assumes the size of G' is much larger than N. Assuming $L = (k \cdot N)^{\frac{1}{\delta}}$ for a fixed small constant $\delta > 1$, we can use the Main Results 1 and 2 from [2] and show that there exists a protocol to solve BCQ of any star $\mathcal{H}$ in: (1) $O(k)$ rounds with $p = (k \cdot N)^{2-\frac{2}{\delta}}$ and (2) $O(\log_2(k))$ rounds with $p = (k \cdot N)^{6-\frac{2}{\delta}}$.

Before we instantiate our model for a comparison with the above models, we would like to state that while our model can handle the constraint where the size of G' can be larger than N, our techniques cannot. Hence, we restrict our comparison to the model in Section A.1.2. We now instantiate our model (Model 2.1) with G' and assume that each edge in G' has capacity

$$L' = \frac{L}{k} = \frac{N}{p}. \tag{13}$$

Note that this is a weaker version of Model A.1 since node capacities don't necessarily translate to equal edge capacities when the goal is to compute q on G'. We take this route as it helps us make a fair comparison with Section A.1.2.

A.1.4 Our Results in Model A.1. We show how our upper bound techniques apply for solving BCQ of any star $\mathcal{H}$ on G'. We can instantiate Corollary 4.2 with capacity $\widetilde{\Theta}(1)$ to get $O\left(\min_{\Delta \in [|V(G')|]} \left(\frac{N}{\mathrm{ST}(G', K, \Delta)} + \Delta\right)\right)$ rounds. We claim that $\min_{\Delta \in [|V(G')|]} \left(\frac{N}{\mathrm{ST}(G', K, \Delta)} + \Delta\right) = O\left(\frac{N}{p}\right)$. To see this, we show such a Steiner tree packing containing p trees with diameter 2 – each node in the p-clique in G' along with all its k edges incident on K forms a Steiner tree. Since there are p such nodes, we can obtain such a packing. Recall that when each edge in G' has capacity L' (instead of the $O(\log_2(D))$ capacity in Model 2.1), our upper bound gets divided by L'. Thus, we have an upper bound of $O\left(\frac{N}{L' \cdot p}\right) = O(1)$ (using (13)) i.e., a constant number of rounds.
Note that a lower bound of one round on the number of rounds is trivial. Hence, we can obtain a tight bound of $\widetilde{\Theta}(1)$

for any star $\mathcal{H}$, resulting in an one round protocol matching results in Section A.1.2.

Given the tight results for the star case, there is a natural generalization for our protocol and bounds when $\mathcal{H}$ is a forest using ideas from the proof of Lemma 4.4. We start by noting that all stars at the same level in $\mathcal{H}$ can be computed simultaneously since each node in K is directly connected to each node in the (p)-clique. In particular, we can run the star protocol used above on all these stars simultaneously but we still need to be able to uniquely identify the stars computed. It's not too hard to see that this can be done with $O(\log(y(\mathcal{H})))$ additional information for each internal node v. This results in an upper bound of

$$O\left(D' \cdot \log(y(\mathcal{H})) \cdot \min_{\Delta \in [|V|]} \left(\frac{N}{\mathrm{ST}(G', K, \Delta)} + \Delta\right)\right) = O\left(\frac{D' \cdot \log(y(\mathcal{H})) \cdot N}{p}\right),$$

where $\mathrm{ST}(G', K, 2) = p$ and D' is the diameter of $\mathcal{H}$. If we divide our upper bound by L' and substitute its value from (13), we can use ideas similar to those used in the star case to obtain a protocol with $O(D')$ rounds. However, our lower bound techniques do not work for the assignment of functions to K in Model A.1. We would like to mention that the model in Section A.1.2 takes $\Theta(\log(D'))$ rounds for this case (though the upper bound only holds for the special case of matching databases).

Finally, for general simple graphs $\mathcal{H}$, we decompose $\mathcal{H}$ into a forest and a core by Definition 2.7. We use the *trivial protocol* on the core, which is basically sending all functions to one player in K and is independent of the induced query in the core. We would like to mention that this is worse than existing protocols [2, 9] for $\mathcal{H}$ with non-constant degeneracy d since we do not exploit any information about the query.

Before we move to the next model, we would like to mention here that the results of Section A.1.2 and our results match up to a constant factor for the case when $\mathcal{H}$ is a star. The upper bounds match since the protocols in both cases split the input functions the same way – the model in Section A.1.2 uses hashes to achieve this and we use Steiner tree packings for the same. The results however start diverging even when $\mathcal{H}$ is a tree of small depth.

A.2 General MPC model

We now perform our second and final comparison. We formally define the model from [36], which is a followup of [9, 10] and performs a *worst-case analysis* of the communication cost for join queries. All the three models are described in [37]. We define it in the language of our model like we did for Model A.1.

Model A.2 (MPC(ϵ)). *Let ϵ be a fixed value s.t. $0 \le \epsilon < 1$. We are given a query q and its underlying hypergraph $\mathcal{H} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ with input functions f_e having at most N non-zero values for every $e \in \overline{\mathcal{E}}$. We consider the network topology G'', which is a clique on p. The input of size $k \cdot N$ is uniformly partitioned across the p nodes. Let $K = V(G'')$. It follows that $|K| = p$. All nodes in G have capacity $L(\epsilon)$. The capacity of a vertex bounds the number of bits it can receive in each round. Given this setup, we would like to compute* BCQ *(and more generally an* FAQ*) of $\mathcal{H}$ on G''.*

To design a protocol for this computation, we can assume that every node in G'' has the knowledge of q and G''. At the end of the protocol, a pre-determined player in K knows the answer to q.

Finally, given the above setup, our goal is to design protocols that minimize the number of rounds to compute q on G''. This model does not take into account the internal computation done by the p nodes and assumes the nodes co-operatively compute the answer to q.

We now summarize the differences of this model from ours.

A.2.1 *Differences from our Model.*

- MPC(ϵ) assumes a specific choice of a network topology G'' as opposed to general topology in our model.
- MPC(ϵ) assumes a uniform distribution of the input across the p nodes instead of one function being completely assigned to a specific node in G'' in our model.
- MPC(ϵ) works with node capacities like Model A.1, whereas ours works on edge capacities.
- The model in [36] designs protocols wherein the number of rounds either constant or a function of k. The number of rounds in our model are a function of N.
- The model in [36] proves results for computing natural join whereas we look at BCQ (and more generally FAQs). Note that their upper results for natural join apply for BCQ as well (but lower bound results do not transfer).

We consider the instantiation of this model by [36]. We would like to mention here that the models studied in [2, 9, 10] can all be instantiated in this setting only for proving upper bounds.

A.2.2 *Fixing p and Determining L [36].*

This model assumes N is larger than the size of G' and there are no restrictions in the input functions. Using Theorems 3.1 and 3.3 of [36], it can be shown that there exists an optimal one round protocol to solve BCQ of any star $\mathcal{H}$ on G'' with $L\left(\epsilon = 1 - \frac{1}{k}\right) = \Omega\left(\frac{N}{p^{1-\epsilon}}\right) = \Omega\left(\frac{N}{p^{\frac{1}{k}}}\right)$. Further, when $\mathcal{H}$ is a forest, BCQ of

$\mathcal{H}$ on G'' takes $O(k)$ rounds[23] with $L\left(\epsilon = 1 - \frac{1}{\rho^*(\mathcal{H})}\right) = \Omega\left(\frac{N}{p^{1-\epsilon}}\right) = \Omega\left(\frac{N}{p^{\frac{1}{\rho^*(\mathcal{H})}}}\right)$ using ideas in Section 4 of [36]. Here, $\rho^*(\mathcal{H})$ denotes the edge cover number of $\mathcal{H}$ (i.e., size of the minimum edge cover of $\mathcal{H}$).
We now instantiate our model (Model 2.1) with G'' and assume that each edge in G'' has capacity

$$L'' = \frac{L(\epsilon)}{p}. \tag{14}$$

Further, we assume that the input functions are not distributed uniformly but rather based on some pre-determined hash functions. Note that this certainly makes our model (Model A.1) more restrictive since node capacities don't necessarily translate to equal edge capacities when the goal is to compute q on G'' and the hash-based split (see Appendix H in the full paper [38]) restricts the way in which input functions can be distributed across nodes in G''. We opt for this since it helps us make a fair comparison with Section A.2.2.

A.2.3 Our Results in Model A.2. We now show how our upper bound techniques apply in this model for solving BCQ of any star $\mathcal{H}$ on G'. We do not compare lower bounds here since (1) [36] lower bounds do not hold for BCQ (or at least it does not follow immediately from their lower bounds for the join quries) and (2) Our lower bounds for the case when the functions are uniformly distributed over the players are quantitatively very weak. For the upper bound, we can instantiate Corollary H.7 in the full paper [38] with capacity $\widetilde{\Theta}(1)$ to get $O\left(\min_{\Delta\in[p]}\left(\frac{N}{\mathrm{ST}(G'',K,\Delta)} + p\cdot\Delta\right)\right)$ rounds. We claim that $\min_{\Delta\in[p]}\left(\frac{N}{\mathrm{ST}(G'',K,\Delta)} + p\cdot\Delta\right) = O\left(\frac{N}{p} + p\right)$. To see this, we show a Steiner tree packing containing $\frac{p-1}{2}$ trees with diameter 1 – we can greedily keep picking and throwing out paths of length $p-1$ from G'' that contain all the p vertices. Each such path forms a Steiner tree. Since we can identify $\frac{p-1}{2}$ such paths, we can obtain such a packing. Recall that when each edge in G'' has capacity $L(\epsilon)$ instead of the standard $O(\log_2(N))$, our upper bound gets divided by L''. Thus, we have an upper bound of $O\left(\frac{\frac{N}{p}+p}{L''}\right)$. Using (14) and the fact $N \geq p^2$ (from Model 2.1), we get a $O\left(p^{\frac{1}{k}}\right)$ round protocol. Note that this is worse than the one round protocol by Section A.2.2.
For the case when $\mathcal{H}$ is a forest, we can instantiate Corollary H.7 in the full paper [38] with capacity $\widetilde{\Theta}(1)$ to get a bound of

$$O\left(y(\mathcal{H})\cdot\min_{\Delta\in[|V|]}\left(\frac{N\cdot r}{\mathrm{ST}(G'',K,\Delta)} + p\cdot\Delta\right)\right) = O\left(\frac{D'\cdot\log(y(\mathcal{H}))\cdot N}{p}\right),$$

where $\mathrm{ST}(G'',K,1) = p$ and D' is the diameter of $\mathcal{H}$. We can use ideas from Section A.1.4 and from those used in the star case to obtain a protocol with $O\left(D'\cdot p^{\frac{1}{\rho^*(\mathcal{H})}}\right)$ rounds. In particular, to get this bound, we divide our upper bound by L'' and substitute its value form (14). Note that this is worse than the $O(k)$ round protocol by Section A.2.2.
Finally, for general simple graphs $\mathcal{H}$, we decompose $\mathcal{H}$ into a forest and a core by Definition 2.7. We use the *trivial protocol* on the core, which is basically sending all functions to one player in K and is independent of the induced query in the core. As stated in Section A.1.4, this is worse than existing protocols [36] for $\mathcal{H}$ with non-constant degeneracy d since we do not exploit any information about the query.

A.3 Scope for Future Work

Many open questions arise out of this comparison. We summarize them here and leave them for future work.

- Can we modify our model to handle node failures like Models A.1 and A.2 do, using replication?
- Can we improve over our *trivial protocol* for cyclic queries using ideas from [2, 9, 10, 31, 36, 37]?
- Can our algorithmic ideas for set intersection be plugged into the Models A.1 and A.2?
- Can we extend our techniques to handle arbitrary distributions of input functions to nodes in the topology?

[23] For the case when we are interested in computing the join query of $\mathcal{H}$, then there is also a matching $\Omega(k)$ lower bound.