Bounding Threshold Formulas using Multiparty Communication Complexity Techniques

Mark Chen*, Shiv Kampani†‡

Abstract

Recently, [KP22] showed that the communication complexity of a multiparty generalization of the Karchmer-Widgerson game is related to the depth complexity of formulas and circuits consisting of threshold gates. Their work, however, focuses only on upper-bounds for threshold formulas and circuits. In this work, we summarize the techniques of [KP22] and make progress towards proving lower-bounds for threshold formulas and circuits using lifting theorems. We also discuss the application of these results in the field of secure multiparty computation.

1 Introduction

Communication complexity, loosely, refers to the minimum amount of information that collaborating parties must exchange in order to "win" some well-defined game. There is a surprising connection between communication complexity and circuit complexity. Specifically, Karchmer and Widgerson ([KW88]) defined a two-party communication game, for a given boolean function f, whose communication complexity is **equal** to the formula depth complexity of f. An analogous result was also proved for monotone functions. The study of DAG-like communication protocols, instead of tree-like protocols, established a connection between communication complexity and the size complexity of boolean circuits. These results allow bounds on communication complexity to be carried over to formula and circuit complexity.

The line of work initiated by Karchmer and Widgerson remained focused on two-party communication games. Kozachinsky and Podolskii ([KP22]) recently studied a generalization

*yc3879@columbia.edu
†svk2118@columbia.edu
‡In alphabetical order of last names.

of these games to a multi-party setting. Interestingly, they proved that the multi-party communication complexity is equal (upto a constant factor) to the depth complexity of **threshold formulas**. They also showed a similar result for the size complexity of threshold circuits by considering DAG-like protocols.

Cohen et al. ([CDI+13]) outlined the connection of threshold circuits with the design of protocols for secure multiparty computation in the presence of adversaries. Upper bounds on threshold circuit complexity imply efficient multiparty computation protocols and lower bounds imply "types" of adversaries in the presence of which it is impossible to have secure protocols.

The objective of our project is to provide an overview of literature on threshold formula and circuit complexity as well as summarize the techniques and results of [KP22]. While their paper focus mainly on upper-bound results, we focus on lower-bounds and provide an approach for proving results via lifting theorems.

1.1 Threshold gates, formulas, and circuits

Definition 1 (Support)**.** *For any boolean input $x \in \{0,1\}^*$, its support is* $\text{supp}(x)$

In loose terms, a threshold gate outputs a 1 if at least a certain 'threshold' or fraction of its inputs are set to 1's, and 0 otherwise. Threshold gates compute threshold functions:

Definition 2 (Threshold function)**.** *A threshold function, denoted $\mathsf{THR}_a^b : \{0,1\}^b \to \{0,1\}$ is defined as follows: $\forall x \in \{0,1\}^b : \mathsf{THR}(x) = 1 \iff |\text{supp}(x)| \geq a$ and 0 otherwise.*

Definition 3 (Majority function)**.** *A majority function, denoted MAJ_{2n+1} is defined as $\mathsf{THR}_{n+1}^{2n+1}$. A special case is MAJ_3 or a 3-bit majority function.*

We will focus on threshold gates of the form THR_2^{k+1} for $k \geq 2$. For $k = 2$, this is MAJ_3. We define threshold formulas and circuits on this single-gate basis.

Definition 4 (Threshold formula)**.** *A threshold formula defined on n variables is defined as a full binary-tree for which every internal node has a THR_2^{k+1}-gate and every leaf node is a variable x_i for $i \in [n]$. The output of the formula is the output at the root of the tree.*

It is important to note that threshold formulas consist of variables and not literals at leaf nodes. We do not consider any negations. Threshold circuits are defined similarly, with the exception that the full binary tree is replaced by a directed acyclic graph (DAG). That is, THR_2^{k+1}-gates have unbounded fan-out.

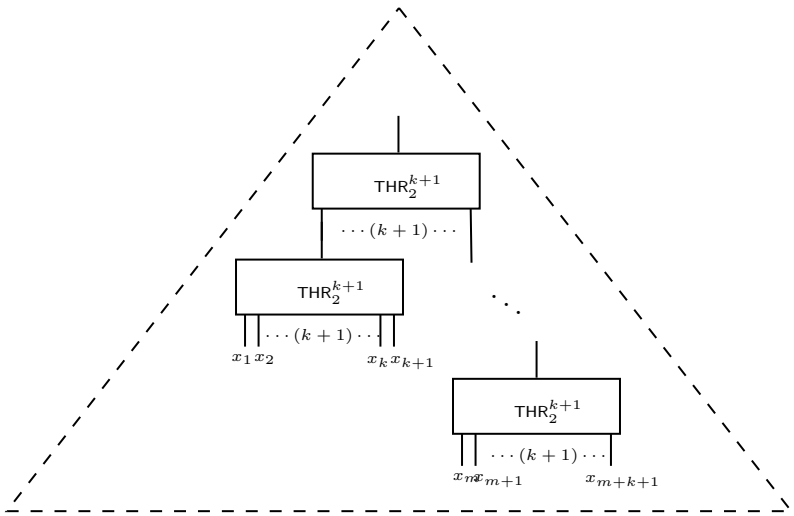


Figure 1: Abstraction of a large THR_2^{k+1}-formula

1.2 Player emulation in Secure Multiparty Computation

As mentioned previously, threshold circuits are connected with the design of secure multiparty computation protocols. We will provide a brief overview of this field to provide motivation for bounding threshold circuit complexity.

We consider the following problem: let $\mathcal{N}$ be a set of n parties who wish to compute a function f on their private inputs securely with respect to an adversary.

Definition 5 (Adversaries and structures)**.** *An adversary $A \subseteq \mathcal{N}$ "corrupts" a subset A of parties. A structure $\mathcal{S}$ is a set of adversaries.*

We will not formally define what it means for an adversary to "corrupt" a set of parties. The general idea is that when an adversary "corrupts" a set of parties A, it is able to access the computation or steps performed by all of the parties $p \in A$. The literature distinguishes between **active** adversaries who are able to alter the computations performed by the parties so as to ruin the overall protocol result and **passive** adversaries who are permitted only to observe the computations performed by the parties so as to learn other uncorrupted players' private inputs.

Definition 6 (Trusted party and ideal protocols)**.** *A trusted party denoted τ is one that cannot be corrupted by any adversary A. It is trivial to design n-party protocols in the presence of a trusted party τ. Such protocols are known as ideal protocols, denoted P_0.*

An example of a trivial n-party ideal protocol for computing a given function f is for all players to send their private inputs to τ, who then computes the function on all players'

inputs and outputs the result. Since no adversary can corrupt τ, the ideal protocol always computes f correctly and securely with respect to any adversary or structure.

Definition 7 (Security)**.** *Protocol P is A-secure if the adversary "can do no better" in the absence of a trusted party τ. A protocol P is $\mathcal{S}$-secure if it is A-secure $\forall A \in \mathcal{S}$.*

The problem of constructing secure n-party protocols for computing any function in the absence of any τ is challenging. However, Hirt and Maurer solved this problem for small n.

Theorem 1 (Passive adversaries)**.** *There exists an explicit 3-party computation protocol for computing any function securely against any passive adversary that corrupts at most 1 player.*

Theorem 2 (Active adversaries)**.** *There exists an explicit 4-party computation protocol for computing any function securely against any passive adversary that corrupts at most 1 player.*

We can apply a technique called **player emulation** to construct n-party protocols using Hirt and Maurer's results for small n. The idea is to start with the ideal protocol P_0 and recursively emulate the computational steps of the trusted party τ using the small n protocol. Figure 2 illustrates this process using the protocol mentioned in Theorem 1.

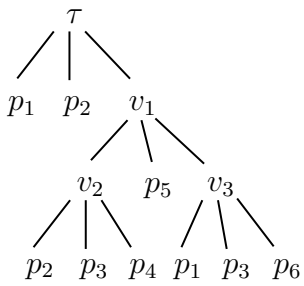


Figure 2: Protocol via player emulation

Each time a recursive player emulation occurs, we can choose either real players p or virtual players v (which are again emulated further). The emulation process ends once all leaves of the resulting protocol tree have real players p.

What adversary structure is this protocol secure against? Clearly, the overall protocol is secure if emulation of the trusted party is secure, which depends on the security of the virtual party v_1 and players p_1, p_2. The security of any virtual party then depends on whether a majority of its "emulators" are secure or not.

From this intuition, it is clear that the security of an n-party protocol constructed via player emulation is equivalent to some threshold or majority computation. This intuition is made clearer when we choose to replace players at the protocol leaves with variables x_i (corresponding to player p_i) and virtual parties and τ with MAJ_3 gates. The result is a majority formula with the property:

Theorem 3. *Let P be an n-party computation protocol constructed via player emulation. Let F be the associated majority formula. P is secure against an adversary A if and only if $F(x_A) = 0$. x_A is the binary encoding of the adversary A, defined as $\forall i, x_i = 1 \iff p_i \in A$ (p_i is "corrupted").*

Corollary 1. *Let P be an n-party computation protocol secure against adversary structure $\mathcal{S}$. Let F be the associated majority formula with P. Then, $\forall A \in \mathcal{S}, F(x_A) = 0$. $\mathcal{S}$ is equivalent to the zero-set $F^{-1}(0)$.*

Corollary 1 tells us how to construct an n-party computation protocol secure against any given adversary structure. Suppose an adversary structure $\mathcal{S}$ is specified. Let $f : \{0,1\}^n \to \{0,1\}$ be the function associated with $\mathcal{S}$, that is, $A \in \mathcal{S} \iff f(x_A) = 0$. We can construct a majority formula F that "lower-bounds" f (see Definition 8) and interpret it as a protocol P_F. Then by Corollary 1, P_F is an n-party computation protocol secure against $\mathcal{S}$.

A particular kind of well-studied adversary structure is a **threshold-$1/k$** adversary structure, which consists of all adversaries that corrupt less than a $1/k$-fraction of all players. The function f corresponding to a threshold-$1/k$ adversary structure (for $kn+1$ players) is $\mathsf{THR}^{kn+1}_{n+1}$. A special case is $k = 2$, which is known as an **honest majority** adversary structure in the literature. The corresponding function is MAJ_{2n+1}.

Definition 8. *$f, F : \{0,1\}^n \to \{0,1\}$. F lower bounds F if $\forall x \in \{0,1\}^n,\ f(x) = 0 \implies F(x) = 0$.*

Now, we will formally establish the link between secure multiparty computation protocols and threshold formulas and circuits.

Definition 9 (Protocol Length)**.** *For a depth-d protocol P constructed via player emulation, the length of the description of the protocol P is $O(2^d)$.*

Thus, given the existence of explicit secure constant-party protocols (as in Theorems 1, 2), **efficient threshold formulas can be used to construct short-length multiparty computation protocols** for given adversary structures. This provides motivation for upper-bounding threshold formulas. Additionally, lower bounds on the depth complexity of threshold formulas imply that short multiparty computation protocols do not exist for certain adversary structures.

1.3 Literature review

Based on the motivation from secure multiparty computation, literature has focused on upper bound results. Specifically, it has focused on answering the following question:

Question 1. *Does there exist a $O(\log n)$-depth THR^{k+1}_2-formula for computing $\mathsf{THR}^{kn+1}_{n+1}$?*

This is still an **open problem** for arbitrary k. The best result we have so far is $O(\log^2 n)$-depth, which was proved using multiparty communication complexity techniques by [KP22]. The following list is a summary of the progress on this problem:

1. Exponential-time computable $O(n)$-depth THR_2^{k+1}-formula for $\mathsf{THR}_{n+1}^{kn+1}$ [CDI$^+$13].

2. Poly-time computable $O(\log n)$-depth THR_2^{k+1}-formula for $\mathsf{THR}_{n+1}^{kn+1}$ $\forall x \in \{0,1\}^{kn+1} : |\mathrm{wt}(x) - \frac{1}{k}| \geq \Omega(\frac{1}{\sqrt{\log n}})$ [CDI$^+$13].

3. Poly-time computable $O(\log^2 n)$-depth THR_2^{k+1}-formula for $\mathsf{THR}_{n+1}^{kn+1}$ [KP22]

For the special case of $k = 2$, this problem was solved by [KP22] using the same techniques. The following list is a summary of the progress on Question 1 with $k = 2$, that is, the problem of computing MAJ_{2n+1} using an $O(\log n)$-depth MAJ_3-formula:

1. $O(\log n)$-depth monotone formula for MAJ_{2n+1} [AKS83].

2. Poly-time computable $O(n)$-depth MAJ_3-formula for MAJ_{2n+1} [HM99].

3. Poly-time computable $O(\log n)$-depth MAJ_3-formula for MAJ_{2n+1} $\forall x \in \{0,1\}^{kn+1} : |\mathrm{wt}(x) - \frac{1}{2}| \geq 2^{-O(\sqrt{\log n})}$ [CDI$^+$13].

4. Poly-time computable $O(\log n)$-depth MAJ_3-formula for MAJ_{2n+1} [KP22].

In Section 3, we will prove the (underlined) results of [KP22] mentioned above.

1.4 Structure of this report

- (In section 2) We define the key preliminaries for understanding the core reduction in [KP22], including relevant communication games, monotone functions, multiparty mKW games and the key functions of [KP22]'s concerns, namely $\mathsf{THR}_{n+1}^{kn+1}$ and MAJ_3.

- For $\mathsf{THR}_{n+1}^{kn+1}$ and MAJ_3, we discuss the following:

  - (In section 3) We discuss formula depth upper bound results.
  - (In section 5) We discuss formula depth lower bound results. In addition, in section 4, we cite more literature about two-party deterministic lifting and give some examples of how people have been using lifting to show lower bounds. Taking this idea of lifting from section 4, we discuss potential ways to generalize lifting to multiparty as our NEWLY proposed directions in sections 5.2 and 5.3
  - (In section 6) We discuss circuit size results (by considering DAG-like communication protocols as circuits).

- (In section 7) We conclude by summarizing open problems to state-of-the-art results and how to expand and use multiparty lifting.

2 Preliminaries

2.1 Communication complexity

In *two-party* communication complexity, we will define a **communication game** between two players: Alice and Bob.

Definition 10 (Two-party Game)**.** *Let X, Y, Z be finite sets. We define a relation $R \subseteq X \times Y \times Z$. Alice and Bob are given $x \in X$ and $y \in Y$ respectively. The goal is to output $z \in Z : (x, y, z) \in R$.*

Definition 11 (Protocol)**.** *A communication protocol π for a given communication game satisfies the property: $\forall (x, y) \in X \times Y,\ (x, y, \pi(x, y)) \in R$. Its communication complexity, $\mathrm{CC}(\pi)$, is the number of bits that the players communicate.*

We will initially focus on protocols that look like binary trees (where at each node either Alice or Bob communicates). Later, we will also consider DAG-like protocols.

Definition 12 (Communication Complexity)**.** *The communication complexity of a two-player game with relation R, $\mathrm{CC}(R) = \min_\pi \mathrm{CC}(\pi)$, or the minimum number of bits that Alice and Bob must communicate in any protocol π for the game.*

Definition 13 (Multiparty (NIH) Game)**.** *A k-player communication game is a straightforward generalization of the two-player case. Let $X_1, \ldots, X_k, Z$ be $(k+1)$ finite sets. We define a relation $R \subseteq X_1 \times \cdots \times X_k \times Z$. $\forall i \in [k]$, player i is given an input $x_i \in X_i$. The goal is to output $z \in Z : (x_1, \ldots, x_k, z) \in R$.*

Note that any given player can see only their input. Thus, this model is known as Number-In-Hand or NIH. A more commonly studied model for multiparty communication is Number-On-Forehead (NOF), in which a player is able to see all but their own input. We will focus solely on multiparty communication in the NIH model.

2.2 Monotone formulas and mKW_f games

In this section, we will discuss the result of Karchmer and Widgerson mentioned in the introduction for **monotone boolean** functions.

Definition 14 (Monotone Functions)**.** *$f : \{0,1\}^n \to \{0,1\}$ is monotone if $\forall x, y \in \{0,1\}^n \to \{0,1\}$ we have $\forall i \in [n]\ x_i \leq y_i \implies f(x) \leq f(y)$*

Theorem 4 (Folklore)**.** *Every monotone function f can be computed by a formula F consisting only of $\{\mathsf{AND}, \mathsf{OR}\}$ gates and variables (no negations). Such a formula is known as a **monotone formula**. $d(f)$, the monotone depth of f, is the minimum depth of any such formula F.*

Consider the monotone Karchmer-Widgerson (mKW_f) communication game for a function f:

Definition 15 (mKW_f game)**.** *For monotone function $f : \{0,1\}^n \to \{0,1\}$, we define a two-player communication game as follows: Alice gets $x \in f^{-1}(0)$, Bob gets $y \in f^{-1}(1)$. The goal is to output any $i : 0 = x_i \neq y_i = 1$.*

Theorem 5 ([KW88])**.** $CC(\mathsf{mKW_f}) = d(f)$ *(monotone depth of f)*

Proof. We will only prove one direction. Specifically, we will show that any d-depth monotone formula F for a function f implies the existence of a d-bit communication game for the associated mKW_f game. This is sufficient to show that $\mathrm{CC}(\mathsf{mKW}_f) \leq \mathrm{d}(f)$. The other direction can be found in [KW88]. We will not show it since we do not use it in subsequent proofs.

Assume that we have a monotone formula F of depth-d that computes f. We will construct a communication protocol π for the $\mathsf{mKW_f}$ game.

- If the top-gate of F is AND, let Alice speak. Let $F = F_L \wedge F_R$, where F_L, F_R are depth $d-1$ monotone formulas. $F(x) = 0 \implies F_L(x) = 0 \vee F_R(x) = 0$. Alice communicates '0' if $F_L(x) = 0$, otherwise, '1'. For Bob, $F(y) = 1 \implies F_L(y) = F_R(y) = 1$. Thus, both parties now know a sub-formula F' of depth $d-1$ with $F'(x) = 0, F'(y) = 1$.
- If the top-gate of F is OR, let Bob speak. Let $F = F_L \vee F_R$, where F_L, F_R are depth $d-1$ monotone formulas. For Alice, $F(x) = 0 \implies F_L(x) = F_R(x) = 0$. $F(y) = 1 \implies F_L(y) = 1 \vee F_R(y) = 1$. Bob communicates '0' if $F_L(y) = 1$, otherwise, '1'. Thus, both parties now know a sub-formula F' of depth $d-1$ with $F'(x) = 0, F'(y) = 1$.

Repeating this procedure, we are left at depth-0 formulas, or simply, variables. When a variable (say index i) is reached, Alice and Bob output i since our construction guarantees that $0 = x_i \neq y_i = 1$. This is possible in d bits of communication (one bit per iteration). ■

2.3 Threshold formulas and Q_k functions

Before bounding threshold formulas, it is necessary to determine what functions they can compute. The basis $\{\mathsf{THR}_2^{k+1}\}$ (with no variable negations) is **not** universal. The Q_k functions are a useful characterization for what functions threshold formulas can compute.

Definition 16 (Q_k functions)**.** $f : \{0,1\}^n \to \{0,1\}$ *is a Q_k function if* $\forall x^1, x^2, \ldots, x^k \in f^{-1}(0)$ $\exists i \in [n] : x_i^1 = x_i^2 = \cdots = x_i^k = 0$.

Example 1. $\mathsf{THR}_{n+1}^{kn+1} \in Q_k$

Proof. $\forall x \in (\mathsf{THR}_{n+1}^{kn+1})^{-1}(0)$, $|\mathrm{supp}(x)| \leq n$. Thus, by a union bound, for any k-inputs drawn from $(\mathsf{THR}_{n+1}^{kn+1})^{-1}(0)$, $|\mathrm{supp}(x^1) \cup \cdots \cup \mathrm{supp}(x^k)| \leq kn \implies \exists i \in [kn+1] : i \notin \mathrm{supp}(x^1) \cup \cdots \cup \mathrm{supp}(x^k) \implies \exists i \in [kn+1] : x_i^1 = x_i^2 = \cdots = x_i^k = 0 \implies \mathsf{THR}_{n+1}^{kn+1} \in Q_k$. ■

Definition 17 (Self-dual functions)**.** $f : \{0,1\}^n \to \{0,1\}$ *is* ***self-dual*** *if $\forall x \in \{0,1\}^n$ we have $f(\neg x) = \neg f(x)$.*

Theorem 6 (Folklore characterization for Q_2)**.** $f \in Q_2 \iff f$ *is monotone and self-dual.*

2.4 Q_k communication game

It is not a straightforward task to generalize the mKW_f game to multiple parties. To do so, we consider monotone self-dual (equivalently, Q_2) functions and frame an equivalent game (in terms of communication complexity) that is easier to generalize to multiple parties.

Definition 18 (Q_2 communication game)**.** *For Q_2 function f, we define a two-player communication game as follows: Alice, Bob get $x, y \in f^{-1}(0)$. The goal is to output any $i : x_i = y_i = 0$.*

This game is equivalent (in terms of communication complexity) to the mKW_f game for $f \in Q_2$. This is easy to see when considering what would happen if, in the original game, Bob were to negate all the bits in his input. We then generalize this to the Q_k communication game in the NIH model.

Definition 19 (Q_k communication game)**.** *For some Q_k function f, we define a k-player communication game (in the NIH model) as follows: $(p_1, p_2, \ldots, p_k) \leftarrow (z^1, z^2, \ldots, z^k) \in (f^{-1}(0))^k$. The goal is to output any $i : z^1_i = z^2_i = \cdots = z^k_i = 0$.*

Is this communication game useful? More specifically, does it have the desirable Karchmer-Widgerson property as in Theorem 15? [KP22] answer in the affirmative.

Theorem 7 (Main result of [KP22])**.**

Theorem 7 allows us to bound the depth complexity of threshold formulas by bounding the communication complexity of associated Q_k games. Proving this theorem is equivalent to showing that depth-d communication protocols imply depth-$O(d)$ threshold formulas and vice versa. We will present both directions of the proof with the following organizational structure:

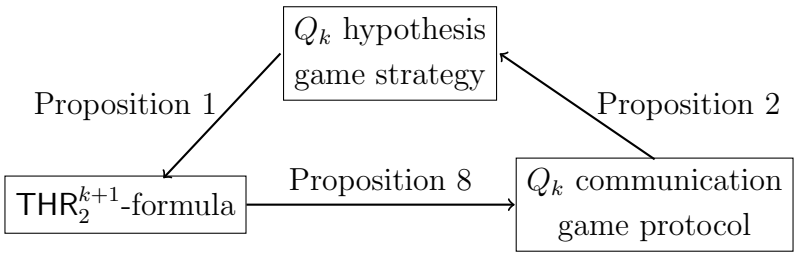


Figure 3: Proof structure for Theorem 7 in [KP22]

Remark 1. *In Theorem 7 and the remainder of our work, the number of players k is regarded as a constant, i.e. $O(1)$.*

2.5 Q_k hypothesis game

Since it is difficult to transform protocols to formulas directly, [KP22] introduce the notion of an intermediary **hypothesis game**. The reason for its introduction is that strategies in Q_k hypothesis games can easily be converted to THR_2^{k+1} formulas.

Definition 20 (Q_k hypothesis game)**.** *For some Q_k function f, we define the Q_k hypothesis game as follows. There are two players: Nature and Learner. Nature chooses some $z \in f^{-1}(0)$ that is kept private from Learner. In each round, Learner makes the following $k+1$ hypotheses:*

$$z \in \mathcal{H}_0, z \in \mathcal{H}_1, \ldots, z \in \mathcal{H}_k$$

If $< k$ of these hypotheses are satisfied, Learner loses the game. Else, Nature communicates the index j of some true hypothesis $\mathcal{H}_j$ to Learner and the game continues. Learner's goal is to output an index $i : z_i = 0$.

3 Upper bounds

3.1 Q_k protocol $\implies$ threshold formula

We will show that depth-d protocols for the Q_k game of a function f imply a depth-$O(d)$ formula F that lower bounds f.

Proposition 1. *Any d-round hypothesis strategy for a Q_k function f can be converted to a depth-d THR_2^{k+1}-formula F that lower-bounds f.*

Proof. Assume that we have a d-round hypothesis strategy for f. We can interpret the strategy as a $(k+1)$-ary tree by considering each round as a node and the 'outputs' corresponding to the bits sent by Nature. We can transform this tree to a formula by placing a THR_2^{k+1}-gate at each node and the variable x_i at each terminal where Learner outputs index i. We claim that this formula lower bounds f. This is because, for any path p from root s to a node v, $\forall z$ 'alive' after p, $F_v(z) = 0$. ■

Proposition 2. *Any d-round hypothesis strategy for a Q_k function f can be converted to a depth-d THR_2^{k+1}-formula F that lower-bounds f.*

Proof. The proof of this proposition is involved. It has been left out given that its mechanical details do not contribute meaningfully to the overall idea presented. A complete proof can be found in [KP22]. It is important to note that the constant associated with this transformation is $2^{k^k} \cdot k$, which is practical only for a small, constant number of players (see Remark 1). ■

3.2 Result for MAJ_3

Theorem 8. *There exists an explicit $O(\log n)$-depth monotone formula for computing MAJ_{2n+1}.*

Proof. Such a formula can be constructed from the AKS sorting network presented in [AKS83]. ■

Theorem 9. *There exists an explicit $O(\log n)$-depth MAJ_3-formula for computing MAJ_{2n+1}.*

Proof. By Theorem 5, there exists explicit $O(\log n)$-depth monotone formula for computing MAJ_{2n+1}. By Theorem 15, we can construct an explicit communication protocol for the $\mathsf{mKW}_{\mathsf{MAJ}_{2n+1}}$ game. Since $\mathsf{MAJ}_{2n+1} \in Q_2$, this protocol can be modified to give a protocol for the Q_2 game for MAJ_{2n+1}. Finally, by Theorem 7, we can construct an explicit $O(\log n)$-depth monotone formula for computing MAJ_{2n+1}. [KP22] also prove that this formula can be constructed in polynomial time. ■

3.3 Result for $\mathsf{THR}^{kn+1}_{n+1}$

Theorem 10. *There exists an explicit $O(\log^2 n)$-depth THR^{k+1}_2-formula for computing $\mathsf{THR}^{kn+1}_{n+1}$.*

Proof. By Theorem 7, it suffices to show an efficient $O(\log^2 n)$-bit communication protocol for Q_k game for $\mathsf{THR}^{\mathsf{kn+1}}_{\mathsf{n+1}}$. The idea is that we can perform a binary search over indices. Let all players maintain indices $\ell, r \in [kn+1]$.

Initially, $\ell \leftarrow 1, r \leftarrow kn+1$. In each round, players share count of 1's on both halves ($O(\log n)$ bits) and choose the half that satisfies the invariant:

$$\sum_{i=1}^{k} |\mathrm{supp}_{\ell:r}(z^i)| < |r - \ell + 1|.$$

After $O(\log n)$ rounds, the range ℓ, r narrows to an index i, which players output. The total number of bits communicated is $O(\log^2 n)$, as required. ■

4 More literature: lifting for two-party KW games

One key take-away from [KP22] is that it generalizes to multiparty the originally 2-party result published in [KW88] for the first time citing Yannakakis (though only in one direction, because even though there is only a constant-time blow up from MAJ_3- and THR^{k+1}_2-bases to the monotone basis, it is not known if all monotone circuits can be reduced to MAJ_3- and THR^{k+1}_2-bases with constant-depth blow-up in general):

Theorem 11. *The* mKW *game communication complexity corresponding to f is asymptotically equal to the monotone depth of f. In particular, if we call the* mKW *game corresponding to f, R_f, we have*

$$CC(R_f) = \textsc{Mono}\text{-}D(f).$$

The multiparty equivalent of this is really what [KP22] shows mainly and uses to show the new upper bound for $\mathsf{THR}^{kn+1}_{n+1}$. Clearly, the proof for the multiparty case is a lot more difficult than the 2-party case (which just uses an induction on a fairly straightforward invariant).

Now, since 2-party mKW games are a very well-studied and widely-used kind of communication games, a natural question to ask is whether there are more things about 2-party mKW games that we can *usefully* generalize to *multiparty*. Turns out, one such technique with potential is what's

known as the "lifting theorem." Here, we summarize the classic lifting theorem being used on 2-party mKW games called DART games (section 4.1) and its application to show lower bounds for GEN and PYRGEN (section 4.2) and for st-connectivity (section 4.3).

4.1 DART games, structured protocols and mKW games

One interesting idea we can also generalize into the multi-party is the lifting theorem (called DART games) initiated by [RM97]. This general technique has now been used to show complexity in quantum, proof complexity, query complexity, KRW conjecture and so on [Raz03, MNP14, GLM$^+$16, GPW17, PR18, DRMN$^+$20].

4.1.1 DART games

In the use case of the first pivotal paper that coins the concept of lifting calls the process of lifting a DART game, and here it goes.

We can define any 2-party communication in the typical sense as a relation. Let X, Y, Z be finite sets. In particular, let A be the input space of Alice and B be the input space of Bob. Then, a protocol $\mathcal{P}$ over $A \times B$ is the one such that, given $(x, y) \in A \times B$, Alice and Bob exchange bits to decide on a $\mathcal{P}(x, y) \in Z$ s.t. $(x, y, \mathcal{P}(x, y)) \in R$.

Definition 21 (DART(m, n))**.** *Now, consider such a relation, R.* DART(m, n) *is a special case of that, where we specify X, Y as follows:*

(1) $X = [m]^n$, $Y = (\{0,1\}^m)^n$. *Let $x \in X$ and $y \in Y$ be the inputs to Alice and Bob.*

(2) R only depends on

$$(y_1(x_1), y_2(x_2) \ldots, y_n(x_n)),$$

where $y_i(x_i)$ chooses the $(x_i \in [m])$-th bit of $y_i \in \{0,1\}^m$. This is what later literature would call the "pointer gadget" (while more modern results would use other gadgets like Inner-Product *to achieve better results).*

Remark 2. *What does it mean to "only depend on" this? It means that, if $y_i(x_i) = y_i'(x_i'), \forall i \in [l]$, then $R(x, y, z) = R(x', y', z), \forall z$.*

Remark 3. *Since R only depends on*

$$(y_1(x_1), y_2(x_2) \ldots, y_n(x_n)),$$

we can rewrite each element as

$$(e_1, e_2 \ldots, e_n).$$

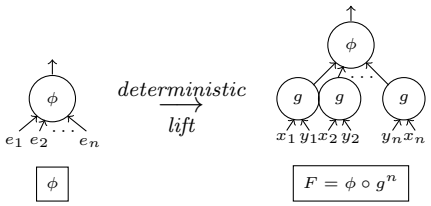
Thus, we can equivalently write

$$R(x, y, z) \equiv R((e_1, \ldots, e_n), z).$$

(3) $R((e_1, \ldots, e_n), z)$ *is a* DNF*-search problem. That is,* $\exists$DNF *tautology* $\phi(e_1, \ldots, e_n)$ *(thus* $\exists i$ *such that the* i*-th clause* $\phi_i(e_1, \ldots, e_n) = 1$*). We let* $z \in Z$ *encode some clause* ϕ_i*, and* $R((e_1, \ldots, e_n), z)$ *being a* DNF*-search problem means that*

$$R((e_1, \ldots, e_n), z) = \mathit{TRUE} \iff z(e_1, \ldots, e_n) = 1.$$

Remark 4. *Using more modern terms, requirement* (2) *is where lifting happens, and each* $g = y_i(x_i)$ *is a gadget (again, this is not the only possible gadget) and* e_i *would be the inputs to the original function before lifting:*



Definition 22 ($R \in DARR(m, n)$)**.** *We say that relation* R *is a* DART *relation if the corresponding game is in* DART(m, n)*.*

4.1.2 Structured protocols

Definition 23 (Structured Protocol)**.** *For a specific gadget,* g*, the structured protocol is the simple one: Alice sends her input* x_i *for a specific* i *for Bob to compute and send back the result of a specific* $g(x_i, y_i)$ *(WLOG, they do not need extra bits to communicate which* i *it is, since they can pre-agree to an order of communication).*

- *In the case of a* DART *game,* $R \in$ DART(m, n)*,* $g(x_i, y_i) = y_i(x_i)$*.*
- x_i *takes* $\log(m)$ *bits and* $y_i(x_i)$ *takes one bit to send.*

We denote the structured complexity of some $R \in$ DART(m, n) *as*

$$SC(R) = \min_{\mathcal{P}}\{\#\textit{rounds that solves } R\}.$$

Remark 5. *Structured protocol is equivalent to a decision tree for the* DNF*-search problem [LNNW95]. So, if* R *is the associated relation of* f *and* F *denotes formula for* f*, then the following is true, up to constants:*

$$SC(R) = D(F).$$

Theorem 12. *Lifting theorem for this special case states that*

$$CC(R) = SC(R) \cdot \Theta(\log(m)) = D(F) \cdot \Theta(\log(m)).$$

Note that the $\log(n)$ *comes from the fact that, in each round of the structured protocol, Alice sends* $\log(m)$ *bits for index and Bob responds with* 1 *bit. So, if it goes on for some constant* k *rounds, the total number of bits communicated is*

$$k \cdot \Theta(\log(m)).$$

Proof. In general, $CC(R) = SC(R) \cdot O(\log(m))$ is trivial, because given some protocol that uses this many bits, i.e. the structured protocol, there could only be protocol that does better.

However, with this special case of gadget, [RM97] shows that $CC(R) = SC(R) \cdot \Omega(\log(m))$ is also true for $m \geq n^{20}$ (this has been improved by later works, but they are the same up to constants). ■

4.1.3 How is it related to multiparty mKW games (definition 15)?

Recall what we have seen many times now, a two-party mKW game can be written as the following relation ($f : \{0,1\}^l \to \{0,1\}$):

$$R_f = \{(x, y, i) \in f^{-1}(1) \times f^{-1}(0) \times [l] : x_i = 1, y_i = 0\}.$$

But, if f is not only monotone, but also self-dual (these two conditions together are equivalent to say that $f \in Q_2$), we further have the following

$$R'_f = \{(x, y, i) \in f^{-1}(0) \times f^{-1}(0) \times [l] : x_i = y_i = 0\}.$$

The R'_f is really a special case of R_f with more restriction on what f could be.

Corollary 2 (Yannakakis Theorem, [KW88])**.** *This is a recall and slight tweak of theorem 11. The* mKW *game communication complexity corresponding to* f *is asymptotically equal to the monotone depth of* f*. In particular, if we call the* mKW *game corresponding to* f*,* R_f*, we have*

$$CC(R_f) = \text{Mono-}D(f).$$

Thus, as a corollary, we can say the same about the special case, R'_f*,*

$$CC(R'_f) = \text{Mono-}D(f).$$

Remark 6. *Here is an important place to note that, with the result in [KP22], we also have, for a general* $f \in Q_k$*, in which case*

$$R_f = \{(x^1, x^2, \ldots, x^k, i) \in \underbrace{f^{-1}(0) \times f^{-1}(0) \times \cdots \times f^{-1}(0)}_{k \text{ many}} \times [l] : x^1_i = x^2_i = \cdots = x^k_i = 0\},$$

we also have

$$CC(R_f) = \text{Mono-}D(f),$$

as they can first be reduced to MAJ_3*- or* THR_2^{k+1}*-basis depending on what* k *is and then reduce to monotone basis with constant-factor blow-up. So, lifting should naturally serve as another tool to show lower bound for* R_f *when* $f \in Q_k$*.*

4.2 Lower bound for GEN & PYRGEN as an example use of lifting

This is a pivotal example to show how lifting through DART games and mKW games play together to show both upper bounds and lower bounds of certain monotone functions.

4.2.1 GEN

GEN is also known as the PATH-SYSTEMS, which was formulated as early as in [Coo70]. It has the following recursive definition:

Definition 24 (GEN Function)**.** *Let* $t_{ijk} \in \{0,1\}$ *be a bit and* $\{t_{ijk}\}_{1\le i,j,k\le l}$ *has a total of* l^3 *such bits. We say that* 1 *generates* k *if:*

- *(Base Case)* $k = 1$*, or*
- *(Recursive Case)* $(t_{ijk} = 1$ *for some* $i, j) \wedge (1$ *generates* $i) \wedge (1$ *generates* $j)$*.*

Here is what GEN *does: it takes* $\{t_{ijk}\}_{1\le i,j,k\le l}$ *as input, and determine if* 1 *generates* l*. Notice that* $\mathsf{GEN}(t_{111}, \ldots, t_{lll}) = 1 \iff 1$ *generates* l*.*

Proposition 3. GEN *is a monotone function.*

Proof. One cannot make GEN evaluate to 0 if it was already evaluating to 1, by flipping any t_{ijk} from 0 to 1 (as the previous path of recursion would still exist from l to 1). ■

Definition 25 (GEN Function, Alternative)**.** *We can also phrase the recursion from the other direction:* $\mathsf{GEN}(t_{111}, \ldots, t_{lll}) = 1 \iff l$ *belongs to the smallest set* $X \subseteq [l]$ *such that:*

- $1 \in X$*.*
- $i, j \in X \wedge t_{ijk} = 1 \implies k \in X$*.*

4.2.2 PYRGEN

We talk about PYRGEN because it can be formulated as a $\mathsf{DART}(m,n)$ game. PYRGEN will later be related to GEN, though this relation may not presently seem obvious.

Definition 26. *Recall definition 21 for a* $\mathsf{DART}(m,n)$ *game. We define the* $\mathsf{PYRGEN}(m,d)$ *game by matching the definition to each of the required items of a* $\mathsf{DART}(m,n)$ *game:*

(1) Let $\{x_{ij}\}_{1\le j\le i\le d}$ *be the input to Alice and* $\{y_{ij}\}_{1\le j\le i\le d}$ *be the input to Bob.* $x_{ij} \in [m]$ *and* $y_{ij} \in \{0,1\}^m$*. Notice that it is just a different way to enumerate, they are nonetheless tuples of elements as required. The inputs are both* $1+2+\cdots+d = \frac{(d+1)d}{2} = \binom{d+1}{2} =: n$*-dimensional.*

(2) The index gadget still works, $x_{ij} \in [m]$ indexes the position of $y_{ij} \in \{0,1\}^m$ so $y_{ij}(x_{ij}) \in \{0,1\}$ just like before. Similarly, we denote $e_{ij} = y_{ij}(x_{ij})$.

(3) It must produce 1 in agreement with some DNF *tautology, and here it is*

$$\phi = [e_{11} = 0] \vee \left[\bigvee_{1 \le j \le i \le d-1} \left(e_{ij} = 1 \wedge e_{(i+1)j} = 0 \wedge e_{i(j+1)} = 0 \right) \right] \vee \left[\bigvee_{j} e_{dj} = 1 \right].$$

Intuitively, this is always true if we think of it is a pyramidal path from $(1,1)$ to some (d,j) where each level i has at most i elements for $1 \le i \le d$. Then, all this DNF *is saying is:*

- *Either it never made out of the origin, $(1,1)$, or*
- *it hits a dead-end somewhere on the way before hitting the d-th level, or*
- *it hits the d-th level which is the final level.*

So, we define the accepting state of PYRGEN *according to this* DNF*, ϕ. In particular, we want $R((e_{11}, e_{21}, e_{22} \ldots, e_{dd}), z) = 1 \iff z$ is a satisfied clause of ϕ.*

4.2.3 The exact structured communication complexity of PYRGEN

Proposition 4. $SC(\mathsf{PYRGEN}(m,d)) = O(d)$.

Proof. Simple induction on the steps of the game:

- (Base Case) One round to check e_{11}. If the bit sent back is 0, then we are done. If not, we keep going.
- (Induction) At this step we already know that $e_{ij} = 1$, then we use two rounds at most to check if $e_{(i+1)j} = 1$ and $e_{i(j+1)} = 1$. If both are 0, then we are done here too. If not, we go to any one that is 1 for the next round.
- (End) If we reach the bottom level and $e_{dj} = 1$ too, we also end there without further queries.

This is for a total of at most $1 + (k-1)2 = O(k)$ rounds of communications, i.e.

$$SC(\mathsf{PYRGEN}(m,d)) = O(d).$$

■

Proposition 5. $SC(\mathsf{PYRGEN}(m,d)) = \Omega(d)$.

Proof. We consider an adversarial argument where Bob tries to make the game run as long as possible (namely, his input values y_{ij} could change accordingly in order to generate $y_{ij}(x_{ij})$ he sees fit). If $d \le 2$, then this lower bound is trivial, so let's assume WLOG that $d \ge 3$. Let's define the following two terms for convenience:

Definition 27 (Legal Paths)**.** *It is a path from vertex* $(1,1)$ *to* (d,j) *for some* j *that contains edges only of the types* $((i,j),(i+1,j))$ *and* $((i,j),(i,j+1))$*.*

Definition 28 (Good Paths)**.** *A good path is a legal path where each vertex* (i,j) *on it has* $e_{ij}=1$ *or the* e_{ij} *is not known yet.*

Claim 1. *If the adversarial Bob always responds* $e_{ij}=0$ *for any query unless there will be no good paths left with such an answer, Alice must keep asking questions until she received a* 1 *answer to one of the* e_{dj}*'s.*

Proof. Suppose Alice were able to determine the correct answer earlier, at (i,j) with $i<d$, we have two case:

- Alice gets an answer of 1 at (i,j) where $i<d$. By Bob's adversarial strategy, it relies with 1 only if he must do so to keep a good path. But this also means that one of $e_{i+1,j}$ and $e_{i,j+1}$ must be 1, otherwise there wouldn't be any legal paths there. So, Alice won't be able to terminate using the second condition of having a dead-end neither at the origin or the d-th level.
- Alice gets an answer of 0 at (i,j) where $i<d$. Then, to determine the game there we must have either $(e_{i-1,j-1}=1\wedge e_{i,j-1}=0\wedge e_{i,j}=0)$ or $(e_{i-1,j}=1\wedge e_{i,j}=0\wedge e_{i,j+1}=0)$. WLOG, lets consider it is the case where $(e_{i-1,j-1}=1\wedge e_{i,j-1}=0\wedge e_{i,j}=0)$. But, by Bob's adversarial strategy, $e_{i-1,j-1}=1$ only if it has a good path under it, so there is a contradiction, too.

■

It is not hard to show that in order to make Bob answer $e_{dj}=1$, Alice will need to ask $d-1$ other questions to get there. Therefore,

$$SC(\mathsf{PYRGEN}(m,d))=\Omega(d).$$

■

Remark 7. *Combining the two directions just shown, we have*

$$SC(\mathsf{PYRGEN}(m,d))=\Theta(d),$$

or, by theorem 12,

$$CC(\mathsf{PYRGEN}(m,d))=\Theta(d\cdot\log(m)).$$

4.2.4 Monotone depth lower bound of GEN

This is done by reducing the mKW-game for GEN to PYRGEN, thus showing that PYRGEN is at least as hard as GEN, effectively carrying over PYRGEN's lower bound to GEN.

Recall definition 24, a GEN game is defined on a l^3-bit input, and each t_{ijk} recursively defines if k can be generated by 1. We associate this definition with PYRGEN.

Firstly, we let

$$l = m \cdot \binom{d+1}{2} + 2,$$

because of the following. We let each t_{ijk} be defined as a tuple

$$\text{GEN-elements} = \{((i,j),k)\}_{1 \le j \le i \le d \text{ and } k \in [m]}.$$

In lieu of the pyramid that we discussed in this section, this is saying that we are associating m values to each vertex of the pyramid. Here is why l is defined to be such a value:

- m distinct values on each of the $1 + 2 + \cdots + d = \binom{d+1}{2}$ vertices in the d-level pyramid.
- One vertex above $(1,1)$ as the target (which we denote by l).
- One vertex below the (d,j)'s as the source (which we denote by 1).

Definition 29 (Consistent with Structure of the Pyramid)**.** *Let $v_1, v_2, v_3 \in$ GEN-elements. We say that (v_1, v_2, v_3) is consistent with the structure of the pyramid if:*

- *(At Source) $v_1 = v_2 = source$ and $v_3 = ((d,j),k)$ for some $j \in [d]$ and $k \in [m]$. This triple marks the beginning.*
- *(At Target) Quite analogously, $v_1 = v_2 = ((1,1), \langle k \in [m] \rangle)$ where k could be different values and $v_3 = target$. This triple marks the end.*
- *(Triangle) A general triple that is neither the first or the second cases: $v_1 = ((i+1,j),k_1), ((i+1,j+1),k_2), ((i,j),k_3)$.*

Remark 8. *Notice that, for each consistent tuple (v_1, v_2, v_3), v_3 is going up the pyramid based on the v_1, v_2.*

Theorem 13.

$$CC(\text{PYRGEN}(m,d)) = O\left(\text{Mono-KW-}CC(\text{GEN})\right).$$

Proof. As what we set out to do in the beginning of this section (section 4.2.4), we are going to take an instance of PYRGEN and use it to construct a GEN instance, which is pretty typical for some theorem like this. So, let $\{x_{ij}\}_{1 \le j \le i \le d}$ be the input to Alice and $\{y_{ij}\}_{1 \le j \le i \le d}$ be the input to Bob. We want to turn Alice's input into an instance U such that $\text{GEN}(U) = 1$, and Bob's into V such that $\text{GEN}(V) = 0$ (by definition of a mKW game of GEN).

To start with, we clean (v_1, v_2, v_3) triples that are not consistent with the pyramid by setting t_{v_1,v_2,v_3} to 0, so they can never be generated by 1 by definition.

Alice: It's simple, we want a path from the source GEN-element tuples to the target tuple. Notice that each $x_{ij} \in [m]$, so we set those as the viable values of each vertex of the pyramid. That is, we define $e_{ij} = ((i,j), x_{ij})$. Thus, we inductively define U as $\{t_{v_1,v_2,v_3}\}_{v_1,v_2,v_3 \in \text{GEN-elements}}$:

- (At Source) If $v_1, v_2 = source$, then they generate $e_{dj}, \forall j \in [d]$ (i.e. $t_{source,source,e_{dj}} = 1, \forall j \in [d]$).
- (At Target) If $v_1, v_2 = e_{11}$, then $v_3 = target$ (i.e. $t_{e_{11},e_{11},target} = 1$).
- (Triangle) If $v_1 = e_{i+1,j}$ and $v_2 = e_{i+1,j+1}$, then $v_3 = e_{i,j}$ is the one generated by them (i.e. $t_{v_1,v_2,v_3} = 1$).
- All other $t_{v_1,v_2,v_3} = 0$.

By construction, we specify exactly a path with $e_{ij} = 1$ generated layer by layer consistent with the pyramid from bottom up, based on the input $\{x_{ij}\}_{1 \le j \le i \le d}$, so, clearly $\mathsf{GEN}(U) = 1$.

Bob: Note each $y_{ij} \in \{0,1\}^m$. We consider vertex of the pyramid as GEN-elements, specified by:

- (At Source) Colored 0.
- (At Target) Colored 1.
- (Intermediate) Each $((i,j),k)$ is colored by $y_{ij}(k)$.

Here is how Bob assigns $V = \{t_{v_1,v_2,v_3}\}_{v_1,v_2,v_3 \in \mathsf{GEN}\text{-elements}}$:

- If v_1 and v_2 are both colored 0 and v_3 is colored 1, then $t_{v_1,v_2,v_3} = 0$.
- Otherwise, as long as v_1, v_2, v_3 are consistent with pyramid, they are set to 1 (the ones not consistent have been dealt with).

Thus, for v_3 at the next level be generated by the previous level, it must be colored 0, because the source is colored 0. But, *target* is colored 1, so it cannot be generated by the source, so $\mathsf{GEN}(V) = 0$.

KW-Game with U, V: Output (v_1, v_2, v_3) such that $t_{v_1,v_2,v_3} = 1$ by Alice and $t_{v_1,v_2,v_3} = 0$ by Bob. ■

Corollary 3. Monotone-NC $\neq$ Monotone-P. *It suffices to show that $\exists \epsilon > 0$ s.t. the monotone-depth of GEN is $\Omega(l^\epsilon)$, i.e. at least polynomial, where l is approximately the input size.*

Proof. By remark 7, $SC(\mathsf{PYRGEN}(m,d)) = \omega(d)$. Then, by theorem 12, the communication complexity of mKW-game for GEN is $\Omega(d \cdot \log(m))$. Recall that $l = m \cdot \binom{d+1}{2} + 2$, if we let $m = d^4 0$, we have

$$l = d^{40} \cdot \frac{d(d+1)}{2} + 2 = O(d^{42}),$$

so $\Omega(d \cdot \log(m)) \geq \Omega(l^{1/42})$. GEN is in Monotone-P; hence, this shows the corollary. ■

4.3 Lower bound for *st*-connectivity

Historically, [KW88] has a more involved proof for the $\log^2(n)$ lower bound for monotone circuits that compute *st*-connectivity through a reduction to FORK games. Here, we see how the concept of lifting can significantly simplify some of the proofs like this.

Definition 30 (*st*-Connectivity)**.** *Let there be $n+2$ vertices forming the set V - n regular vertices, and 2 special ones for source and target respectively. There are $(n+1)^2$ total potential edges on this graph. We use $\{x_{v,u}\}_{v,u\in V}$ to denote if an edge exists or not. Where, $x_{v,u}=1$ denotes a directed edge from v to u exists; the edge doesn't exist, otherwise. Let α be an assignment to $\{x_{v,u}\}_{v,u\in V}$, then*

$$\text{ST-CONN}(\alpha)=1 \iff G_\alpha \text{ has an } s\text{-}t \text{ path.}$$

Proposition 6. *The st-connectivity problem is clearly monotone, as flipping any 0 assignment to 1 (i.e. adding in more directed edges) won't disconnect previously connected vertices.*

Definition 31 ($CONN(m,n)$)**.** *$CONN(m,n)$ is a $\mathsf{DART}(m,n)$ game.*

- *Alice gets $\{x_i\}_{1\le i\le n}$ and Bob gets $\{y_i\}_{1\le i\le n}$, where $x_i\in[m]$ and $y_i\in\{0,1\}^m$.*
- *We use index gadget, denoted $e_i=y_i(x_i)$.*
- *Alice and Bob search for i and accepts i in one of the following situations:*

  (1) $i=1$ and $e_i=1$.

  (2) $i=n$ and $e_i=0$.

  (3) $i\le n-1$ and $(e_i=0)\wedge(e_{i+1}=1)$.

  (4) $i\le n-1$ and $(e_i=1)\wedge(e_{i+1}=0)$.

  These give us the DNF tautology:

$$\left[\bigvee_{i\le n-1}(e_i\neq e_{i+1})\right]\vee(e_1=1)\vee(e_n=0).$$

Proposition 7.

$$SC(CONN(m,n))=\Theta(\log(n))$$

Proof. **Upper bound** can be achieved by binary search. First, we use 2 rounds to find out if (1) or (2) is true. If not, we proceed by finding where e_i and e_{i+1} differ. For example, since if we proceed, we know $e_1=0$ and $e_n=1$, so if $e_{\frac{n}{2}}=1$, condition (3) must be met in the first half; if $e_{\frac{n}{2}}=0$, condition (3) must be met in the second half. Continue binary search this way to find the i. **Lower bound** is by (1)-(3) contain $n+1$ distinct cases, meaning $n+1$ leaves, which implies a depth of at least $\log(n)$ to differentiate these options. ■

Corollary 4. *Thus, by theorem 12, we have, letting* $m = n^{20}$,

$$CC(CONN(m,n)) = \Theta(\log(n) \cdot \log(m)) = \Theta(\log^2(n)).$$

Now, all that's left to do is to reduce the mKW-game for *st*-connectivity to $CONN(m,n)$. We start with an instance of $CONN(m,n)$, where Alice gets $\{x_i\}_{1 \le i \le n}$ and Bob gets $\{y_i\}_{1 \le i \le n}$. Similar to the GEN case we have seen, we want to construct U for Alice such that s and t are connected, and U for Alice such that s and t are not connected:

- We let U set exactly the following edges to 1:

$$(s,(1,x_1)),((1,x_1),(2,x_2)),\ldots,((n-1,x_{n-1}),(n,x_n)),((n,x_n),t).$$

- For V, we first let s be colored 0, t be colored 1, and every (i,j) be colored according to $y_i(j)$. V contains exactly all pairs where both vertices are colored the same.

Thus, the lower bound carries over to the mKW-game simulated this way, and so its communication complexity is also $\Theta(\log^2(n))$.

4.4 Why generalize lifting to multiparty?

Sections 4.2 and 4.3 give classic examples of how two-party lifting theorem could lead to proofs for lower bounds of many natural and important problems even when it appeared for the first time and used quite naively. It is natural to consider its multiparty generalization and whether it can be used for probelms that fall into the general Q_k model, which we will point out some ways to do so in section 5.2.

5 Towards lower bounds

5.1 Threshold formula $\implies$ Q_k protocol

Proposition 8. *Let* $k \ge 2$, $f \in Q_k$, *and let* F *be a* THR_2^{k+1}*-formula that lower bounds* f. *Then,* $\exists$ *protocol* π *for the* Q_k*-communication game for* f *such that* $\mathrm{CC}(\pi) = O(\mathsf{depth}(F))$.

Proof. Given this F, we know the inputs are the vectors of the k parties, $x^1,\ldots,x^k \in f^{-1}(0)$, and the output is some $i_t = 0$. Now, let's start from the output and trace back, we notice the following invariant at any gate g of F: $g(x^1) = g(x^2) = \cdots = g(x^k) = 0$, which can be shown by a backward induction (because output corresponds to 0, so tracing back to each THR_2^{k+1}-gate gives 0).

Thus, $g(x^i) = \mathsf{THR}_2^{k+1}(g_1(x^i),\ldots,g_k(x^i)) = 1 \implies \exists$ at most one j s.t. $g_j(x^i) = 1$. So, each party only needs to tell every other party about this j bit for everyone to figure out the remaining common coordinate, l, such that $g_l(x^i) = 0, \forall i \in [k]$. So, the total communication needed is $O(1)$ at each gate from the output to inputs, for a total number of $\mathsf{depth}(F)$ many rounds, so we have $\mathrm{CC}(\pi) = O(\mathsf{depth}(F))$. ∎

5.2 Multiparty lifting theorems

Recall that, in theorem 12, the way [RM97] trivializes the upper bound is by picking a specific kind of protocol that communicates exactly $(\log(m)+1)$ bits in each round for $SC(R_f)$ many rounds (structured complexity is defined in number of rounds not bits transmitted). Thus, it claims that, since we are guaranteed such a communication protocol for such an R, it suffices as an upper bound.

Now, we want to show that, a similar property exists for multiparty DART game (to be defined in definition 32), i.e.

Theorem 14 (k-Party DART Game, Upper Bound)**.**

$$CC(R_f) = SC(R_f) \cdot O(\log(m)).$$

Definition 32. *[k-Party* DART *Games, Denoted* $\mathsf{DART}^k(m,n)$*] To show this generalization would be quite natural, we mostly stay true to the original 2-party* DART *games defined in definition 21, but just generalize in the number of players.*

(1) $X_1 = (\{0,1\}^m)^n, X_2 = X_3 = \cdots = X_{k-1} = ([m]^m)^n, X_k = [m]^n$*. Inputs to parties* $P_1, \ldots, P_k$ *are* $x^i \in X_i$*, respectively.*

(2) R *only depends on*

$$(x_1^1(x_1^2(\ldots(x_1^k)\ldots)), x_2^1(x_2^2(\ldots(x_2^k)\ldots))\ldots, x_n^1(x_n^2(\ldots(x_n^k)\ldots))),$$

where $x_i^1(x_i^2(\ldots(x_i^k)\ldots))$ *works as such:* $x_i^k \in [m]$ *chooses the* x_i^k*-th* ***number*** *of* x_i^{k-1} *(as compared to* x_i^k*-th bit, but it doesn't matter since communication only concerns with the number of bits transmitted, not the computation actually done by each party. That party receiving* x_i^k *can do whatever computation on whatever partition of whatever encoding it has for* m *choices of numbers in* $[m]$*). So, we keep choosing in such a way until we reach* $x_i^2 \in [m]^m$ *at which point we also choose a number in* $[m]$ *based on the previous number in* $[m]$ *we got. Let that number be* i'*, then we output the* (i')*-th number of* x_i^1*, which would be in* $\{0,1\}$*.*

Remark 9. *Similar to remark (2), we define our new k-party gadget g to be*

$$g(x_i^1, x_i^2, \ldots, x_i^k) = x_i^1(x_i^2(\ldots(x_i^k)\ldots)),$$

and we denote $e_i = g(x_i^1, x_i^2, \ldots, x_i^k)$*. Thus, we can equivalently write*

$$R(x^1, x^2, \ldots, x^k, z) \equiv R((e_1, \ldots, e_n), z).$$

Remark 10. *This is the main place that changed from definition 21, by generalizing the index gadget from 2-party to k-party.*

(3) $R((e_1, \ldots, e_n), z)$ *is a* DNF*-search problem. That is,* $\exists$DNF *tautology* $\phi(e_1, \ldots, e_n)$ *(thus* $\exists i$ *such that the i-th clause* $\phi_i(e_1, \ldots, e_n) = 1$*). We let* $z \in Z$ *encode some clause* ϕ_i*, and*

$R((e_1, \ldots, e_n), z)$ *being a* DNF*-search problem means that*

$$R((e_1, \ldots, e_n), z) = \mathit{TRUE} \iff z(e_1, \ldots, e_n) = 1.$$

Remark 11. *All definition 32 does is to generalize the notion of lifting to multiparty. In particular,* (1) *allows it to be k parties and* (2) *allows these inputs to do meaningful indexing under a k-party index gadget.* (3) *is actually entirely unchanged, so we can make use of lifting hard instance of* DNF *search problem to lower bound the lifted search problem as before.*

Proof (Theorem 14). Let $R \in \mathsf{DART}^k(m, n)$ be a k-party DART communication relation. As [RM97], we use the structured protocol to compute this and use it as an upper bound (as by construction, we guarantee a protocol of at most this communication complexity to exist). The worst one needs to do in each round is:

- A_k sends x_i^k to A_{k-1}.
- Keep going, by A_i sending its number indexed by the previous players to A_{i-1}.
- When A_2 picks its number this way, it sends its indexed number to A_1, who would know exactly what $i' \in \{0, 1\}$ to output, so it broadcasts and everyone else knows.

Following this process, in each round, all but A_1 sends $\log(m)$ bits that encodes its indexed number in $[m]$, and A_1 broadcasts by sending one bit. Thus, each round, the number of bits needed is $(k-1)\log(m) + 1$. Again, $SC(R)$ denotes the structured complexity which is the number of rounds needed. So, under this protocol,

$$CC(R) = SC(R) \cdot ((k-1)\log(m) + 1),$$

which means that, in general, we have an upper bound:

$$CC(R) = SC(R) \cdot O(log(m)).$$

■

5.3 Applications of the new multiparty lifting theorem

5.3.1 Direct generalization of two-party DART games shown as background

Definition 33 (PYRGEN^k)**.** *A natural generalization of* PYRGEN *to k parties is the following (similar to definition 26, we formulate it as we fill it into the requirements of a* $\mathsf{DART}^k(m, n)$*):*

(1) For $i \in [k]$, A_i gets input $\{x^i_{j_1,j_2,\ldots,j_k}\}_{1 \le j_2,\ldots,j_k \le j_1 \le d}$. In particular, $x^1_{j_1,j_2,\ldots,j_k} \in \{0,1\}^m$, $x^i_{j_1,j_2,\ldots,j_k} \in [m]^m$ for $i \in [k-1]\backslash\{1\}$ and $x^k_{j_1,j_2,\ldots,j_k} \in [m]$. To ensure that input for each

player is still n-bit/partitions, we notice the following: by the indexing pattern, we have a total positions of

$$1 + 2^{k-1} + 3^{k-1} + \cdots + d^{k-1} = \sum_{\alpha=1}^{d} \alpha^{k-1} =: n.$$

(2) We use the k-party index gadget defined in definition 32 [see item (2)]. Analogously, here, we have

$$\{e_{j_1,j_2,\ldots,j_k}\}_{1 \leq j_2,\ldots,j_k \leq j_1 \leq d}$$

(3) Defined exactly following the scenarios by which the two-party case was defined:

- *Either it never made out of the origin,* $(\underbrace{1,1,\ldots,1}_{\#=k})$*, or*
- *it hits a dead-end somewhere on the way before hitting the d-th level, or*
- *it hits the d-th level which is the final level.*

In particular:

$$[e_{1,1,\ldots,1}=0] \vee \left[\bigvee_{1 \leq j_2,\ldots,j_k \leq j_1 \leq d-1} \left(e_{j_1,\ldots,j_k} = 1 \wedge e'_{j_1,\ldots,j_k} = 0 \wedge e''_{j_1,\ldots,j_k} = 0 \right) \right] \vee \left[\bigvee_{j_2,\ldots,j_k} e_{d,j_1,\ldots,j_k} = 1 \right],$$

where $e'_{j_1,\ldots,j_k}$ *increases some* j_κ *by* 1 *compared to* $e_{j_1,\ldots,j_k}$ *and* $e''_{j_1,\ldots,j_k}$ *further increases some other* j_λ *by* 1 *compared to* $e'_{j_1,\ldots,j_k}$ *(in other words, compared to the* 2*-party pyramid which was* 2*-dimensional, here what we have is a k-dimensional pyramid).*

Remark 12. *What this shows here is that many previous results in the two-party deterministic lifting case, especially ones that are based on index / pointer gadget, can be generalized to multiparty.*

Definition 34 (m-Party GEN Function, denoted GEN^m)**.** GEN^m *is defined recursively just like* GEN*, with the generalization happening this way: we concern with whether or not* $j_1, \ldots, j_m$ *generate* k*, instead. In other words, instead of the original* t_{ijk}*, now we have* $t_{j_1,\ldots,j_m,k}$*. Everything else are defined analogously.*

Proposition 9. *What we have discussed so far should mean that the communication complexity lifting results would carry over to computing* GEN^m*. Using definition 33, we can use a proof very similar to the two-party proof to also show the communication complexity of* GEN^m*.*

5.3.2 Bound $\mathsf{THR}^{kn+1}_{n+1}$ with multiparty deterministic lifting

This is the truly interesting part that we wish we could write out in more details given some more time, and the high-level procedure is as follows:

1. Define a $\mathsf{DART}^k(m,n)$ game, $THRESH^k(m,n)$.

   This should be quite doable just given how much the Q_k communication game resembles the graph games we have discussed so far for $\mathsf{DART}(m,n)$.

2. Take an instance of $THRESH^k(m,n)$, with inputs $x^1, x^2, \ldots, x^k$ and show that we can construct $X^1, X^2, \ldots, X^k$ accordingly as inputs to $\mathsf{THR}_{n+1}^{kn+1}$ and they all evaluate to 0 as a Q_k mKW game.

3. Then carry over the complexity by studying the structured protocol complexity of $THRESH^k(m,n)$.

If all of these could be done completely in details, we should be able to analyze and get some new lower bound results from this new multiparty lifting direction.

6 Circuit size upper bound

So far, we have only talked about formulas for which it makes sense to focus on depth bounds. In fact, we can also consider DAG-like communication protocols [Sok17], in which case size bounds also makes sense. We have so far considered formulas that compute Q_k functions, and now it is also helpful to talk about DAGs in general (in which case considering size is more meaningful). To do so, we formulate communication protocols in terms of a DAG.

Definition 35 (k-Party DAG-Like Communication Protocol, Full Form)**.** *A k-party DAG-like communication protocol π:*

- *Inputs: $\mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k$, which encodes the behavior of each party.*
- *Output: $y \in \mathcal{Y}$.*

π is encoded by $\langle G, P_1, P_1, \ldots, P_k, \phi_1, \phi_2, \ldots, \phi_k, l\rangle$, where there really are four groups of things:

- *G: defines the DAG. For this, it is a 2-ary graph.*
- *$P_1, P_1, \ldots, P_k$: partitions of $V(G)\backslash T(G)$ to assign to k parties, respectively.*
- *$\phi_1, \phi_2, \ldots, \phi_k$: each takes the form $\phi_i : P_i \times \mathcal{X}_i \to \{0,1\}$ and decides where to go next on the DAG, since the DAG is 2-ary [in communication complexity terms, dictates which bit to send for the party that the vertex is assigned to by the partition].*
- *l: it takes the form $T(G) \to \mathcal{Y}$ and maps a terminal node to an output value.*

Definition 36 (k-Party DAG-Like Communication Protocol, Light Form)**.** *To define computability, we take the light form of a DAG-like protocol, which is encoded by $\langle G, P_1, P_1, \ldots, P_k, l\rangle$, in particular without "$\phi_1, \phi_2, \ldots, \phi_k$," so that we can discuss which outputs a certain input **could** reach instead of a specific path that it **will definitely** take.*

Definition 37 (Computability of the Light Form)**.** *π computes $S \subseteq \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k \times \mathcal{Y}$, if $\forall (x^1, x^2, \ldots, x^k) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_k, \exists y \in \mathcal{Y}$ and $t \in T(G)$ s.t. $(x^1, \ldots, x^k)$ visits t, $l(t) = y$ and $(x^1, x^2, \ldots, x^k, y) \in S$.*

In the case of Q_k-communication game, the DAG computes

$$S = \left\{ (x^1, x^2, \ldots, x^k, j) \mid x^1_j = \ldots = x^k_j = 0 \right\} \subseteq \underbrace{f^{-1}(0) \times \cdots \times f^{-1}(0)}_{k \text{ many}} \times [n].$$

Definition 38 (π strongly computes $f \in Q_k$)**.** *If for every $t \in T(G)$ and $x \in f^{-1}(0)$: x is i-compatible with t for some $i \in [k] \implies x_{l(t)} = 0$.*

Theorem 15. *Let $k \geq 2$. Let $f \in Q_k$ and π, as a DAG protocol, strongly computes the Q_k communication game for f. Then, given the light form of π, $\exists$ a poly-time algorithm A that outputs a Q_k-circuit $C \leq f$ s.t.*

- *$D(C) =$ linear in $CC(\pi)$.*
- *$S(C) =$ polynomial in $S(\pi)$.*

Proof. These claims are the generalization of the depth proof by introducing DAG protocol and tweaking specific definitions at places. As upper-bound-type claims, check proofs for propositions 1 and 2. ■

7 Open problems

The following are important open problems:

1. $O(\log(n))$-depth (with **small constant**), deterministic poly-time computable MAJ_3-formula for computing MAJ_{2n+1}.
2. $o(\log^2(n))$-depth deterministic poly-time computable THR^{k+1}_2-formula for computing $\mathsf{THR}^{kn+1}_{n+1}$.
3. What we have shown in section 5.2 is a generalization of the index gadget to multiparty and some of its corresponding preservations of 2-party lifting properties. Index gadget shouldn't be too special, and we may be able to similarly generalize other better more modern gadgets such as Inner-Product and even gadgets with constant SC to CC blow-up for $f \in Q_k$ (as compared to the $\Theta(\log(m))$-blow-up in our case) for some special cases like [SZ07, She08] to multiparty, as well.
4. Eventual, the hopeful goal is to use the notion of multiparty lifting theorem to show better bounds for $\mathsf{THR}^{kn+1}_{n+1}$ in THR^{k+1}_2-basis. In particular,
   - if we can show that $\mathsf{THR}^{kn+1}_{n+1}$ needs a THR^{k+1}_2-circuit of depth $\Omega(\log^2(n))$ by lifting theorem lower bounding techniques illustrated in sections 4.2 and 4.3, then the current upper bound is tight up to a constant (see section 5.3.2 for ideas).

- if we can show that $\mathsf{THR}_{n+1}^{kn+1}$ can be computed by a THR_2^{k+1}-circuit of depth $O(\log(n))$ through lifting, then we also have a $O(\log(n))$ construction for $\mathsf{THR}_{n+1}^{kn+1}$.

References

[AKS83] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in c log n parallel steps. *Combinatorica*, 3(1):1–19, jan 1983.

[CDI+13] Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 185–202, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Coo70] Stephen A Cook. Path systems and language recognition. In *Proceedings of the second annual ACM symposium on Theory of computing*, pages 70–72, 1970.

[DRMN+20] Susanna F De Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, and Robert Robere. Krw composition theorems via lifting. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 43–49. IEEE, 2020.

[GLM+16] Mika Goos, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016.

[GPW17] Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for bpp. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 132–143. IEEE, 2017.

[HM99] Martin Hirt and Ueli Maurer. Player simulation and general adversary structures in perfect multiparty computation. 1999.

[KP22] Alexander Kozachinskiy and Vladimir Podolskii. Multiparty Karchmer-Wigderson Games and Threshold Circuits. *Theory of Computing*, 18(15):1–33, 2022.

[KW88] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 539–550, 1988.

[LNNW95] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM Journal on Discrete Mathematics*, 8(1):119–132, 1995.

[MNP14] Alexis Maciel, Phuong Nguyen, and Toniann Pitassi. Lifting lower bounds for tree-like proofs. *computational complexity*, 23(4):585–636, 2014.

[PR18] Toniann Pitassi and Robert Robere. Lifting nullstellensatz to monotone span programs over any field. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 1207–1219, New York, NY, USA, 2018. Association for Computing Machinery.

[Raz03] Alexander A Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya: Mathematics*, 67(1):145, 2003.

[RM97] Ran Raz and Pierre McKenzie. Separation of the monotone nc hierarchy. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 234–243. IEEE, 1997.

[She08] Alexander A Sherstov. Communication lower bounds using dual polynomials. *arXiv preprint arXiv:0805.2135*, 2008.

[Sok17] Dmitry Sokolov. Dag-like communication and its applications. In *International Computer Science Symposium in Russia*, pages 294–307. Springer, 2017.

[SZ07] Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *arXiv preprint arXiv:0710.0095*, 2007.