Choosing, Agreeing, and Eliminating in Communication Complexity*

Amos Beimel[1], Sebastian Ben Daniel[1], Eyal Kushilevitz[2], and Enav Weinreb[2]

[1] Dept. of Computer Science, Ben Gurion University, Beer Sheva, Israel.
[2] Dept. of Computer Science, Technion, Haifa, Israel.

Abstract. We consider several questions inspired by the direct-sum problem in (two-party) communication complexity. In all questions, there are k fixed Boolean functions $f_1, \ldots, f_k$ and Alice and Bob have k inputs $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$, respectively. In the *eliminate* problem, Alice and Bob should output a vector $\sigma_1, \ldots, \sigma_k$ such that $f_i(x_i) \neq \sigma_i$ for at least one i (i.e., their goal is to eliminate one of the 2^k output vectors); in *choose*, Alice and Bob should return $(i, f_i(x_i, y_i))$ and in *agree* they should return $f_i(x_i, y_i)$, for some i. The question, in each of the three cases, is whether one can do better than solving one (say, the first) instance. We study these three problems and prove various positive and negative results.

1 Introduction

A basic question in complexity theory is how the complexity of computing k independent instances relates to the complexity of computing one instance. Such problems, called *direct sum problems*, have been studied for a variety of computational models. Broadly, the direct sum question asks (with respect to an arbitrary computational model and any complexity measure):

Question 1. Can "solving" k functions $f_1, \ldots, f_k$ on k independent inputs $x_1, \ldots,$ x_k (respectively) be done more "efficiently" than just "solving" each $f_i(x_i)$?

(Of particular interest is the special case where all functions are identical.) Since the inputs are independent, it is tempting to conjecture that in reasonable models the answer is negative. Indeed, it was proved that in several models no significant saving can be obtained; e.g., for decision trees [7, 22, 27, 14]. However, for other models, some savings are possible despite the independence of the inputs, e.g., non-deterministic communication complexity and randomized communication complexity [16, 10], deterministic communication complexity of relations [10], and distributional communication complexity [27]. For other models, the answer is still unknown; e.g., in circuit complexity [11, 23, 29]. Direct sum results are important for understanding the power of a computational model. For example,

* The first author is supported by ISF grant 938/09. The second author is partially supported by the Frankel Center for Computer Science. The third and fourth authors are supported by ISF grant 1310/06.

it was shown in [17] that a negative answer for a variant of this question implies circuit lower bounds, in particular, $\mathcal{NC}^1 \neq \mathcal{NC}^2$. Furthermore, direct sum results on the information complexity have been used to prove lower bounds on the communication complexity of functions [4].

To better understand direct sum questions, a simpler task has been proposed – *eliminating* a vector of answers. More precisely, for k fixed Boolean functions $f_1, \ldots, f_k$, given k inputs $x_1, \ldots, x_k$, find a vector $\sigma_1, \ldots, \sigma_k$ such that $f_i(x_i) \neq \sigma_i$ for at least one i. In other words, given $x_1, \ldots, x_k$, a-priori there are 2^k possible vectors of outputs for the k instances. Solving the direct sum problem is finding the correct vector of outputs; eliminating means returning one of the $2^k - 1$ vectors which is not the vector of outputs. Clearly, if we solve one instance and obtain, say, $a_1 = f_1(x_1)$, then we can eliminate the vector $\overline{a}_1, \sigma_2, \ldots, \sigma_k$ for any $\sigma_2, \ldots, \sigma_k$. The question is if we can do better; that is,

Question 2. Can solving `eliminate`, on k independent instances $x_1, ..., x_k$ of k functions $f_1, \ldots, f_k$, be "easier" than solving the "easiest" function?

(Again, of a particular interest is when all functions are identical.) This question and related ones were studied in the context of polynomial-time computation [1, 3, 8, 9, 13, 19, 24–26, 28] and computation in general [5, 6, 12, 15, 20]. The question was explored for communication complexity by Ambainis et al. [2].

In this work, we introduce two new problems related to the direct sum question, `choose` and `agree`. As before, there are k fixed Boolean functions $f_1, \ldots, f_k$, and we are given k instances $x_1, \ldots, x_k$. In the `choose` problem, the task is to solve one instance of our choice; i.e., return $(i, f_i(x_i))$, for some i. Intuitively, `choose` is allowed to pick an "easy" input and answer it. The question is:

Question 3. Can solving `choose`, on k independent instances $x_1, ..., x_k$ of k functions $f_1, \ldots, f_k$, be "easier" than solving the "easiest" function?

In the `agree` problem, the task is to return $f_i(x_i)$ for some i (possibly without knowing for which instance i it corresponds). That is, if all outputs agree, i.e., $f_1(x_1) = f_2(x_2) = \cdots = f_k(x_k) = \sigma$, then $\texttt{agree}(x_1, \ldots, x_k)$ must return σ, otherwise it may return either 0 or 1. The question is:

Question 4. Can solving `agree`, on k independent instances $x_1, ..., x_k$ of k functions $f_1, \ldots, f_k$, be "easier" than solving the "easiest" function?

Comparing the three tasks, `choose` is the hardest and `eliminate` is the easiest, as solving `choose` implies solving `agree` which, in turn, implies solving `eliminate` (if we get an answer σ for $\texttt{agree}(x_1, \ldots, x_n)$, then we can eliminate the output vector $(\overline{\sigma}, \ldots, \overline{\sigma})$). However, eliminating may potentially be much easier than `choose` and `agree`; for example, if $f_1 = f_2$ and $x_1 = x_2$, then solving `agree` implies solving $f_1(x_1)$, while for `eliminate` one may return $(0, 1)$ (or $(1, 0)$) without any computation. Furthermore, if we can solve `choose` efficiently, then we can solve the direct sum efficiently (use `choose` to solve one instance and solve the other instances independently). We do not know of any connections between the direct sum problem and `agree` or `eliminate`.

We start by mentioning some related work. The direct sum question in communication complexity for deterministic protocols of functions is still open. There is an example of a relation in which saving of $O(k \cdot \log n)$ bits is possible for k instances by a deterministic protocol [10]. For non-deterministic protocols and randomized protocols, some saving is possible for some functions [10, 16]. However, the best possible "generic" upper bound for the direct sum of randomized protocols is not known (while in the non-deterministic case, an additive $O(\log n)$ savings is the best possible [10, 16]). Ambainis et al. [2] study the communication complexity of `eliminate`. They conjecture that, for functions, no saving is possible for deterministic protocols. To support their conjecture they supply, in addition to other results, lower bounds for deterministic, non-deterministic, and randomized protocols for `eliminate` of specific functions.

Our Results. We define the `choose` and `agree` problems and study their properties, as well as the properties of `eliminate`.

- For randomized public-coin protocols, we show that saving of $O(\log k)$ bits is possible for `eliminate` of some functions (e.g., the inner-product function).
- On the negative side, we prove that the randomized communication complexity of solving a function f is a lower bound on the randomized communication complexity of $\texttt{eliminate}_{f^k}$, i.e., on computing eliminate on k instances of f. This implies, together with a trivial upper bound, that the randomized communication complexity of computing a function f characterizes the randomized communication complexity of computing $\texttt{eliminate}_{f^k}$, up-to a factor of $2^{O(k)}$. In particular, our results show that eliminate of IP requires $n - O(k)$ bits even for randomized protocols. This improves a lower bound of $n/(2^{O(k)} \log n \log\log n)$ for randomized protocols for IP proved in [2].
- We relate the complexity of $\texttt{choose}_{f,g}$ to the non-deterministic and deterministic complexity of solving f and g. In particular, we show that if the non-deterministic communication complexity of f and g are high, then the deterministic communication complexity of $\texttt{choose}_{f,g}$ is high. This implies that for most functions the communication complexity of $\texttt{choose}_{f,g}$ is $\Omega(n)$. We prove a similar, however weaker, lower bound for $\texttt{agree}_{f,g}$.

To better understand if saving is possible for `eliminate` in communication complexity, we explore a restriction of eliminate, called `r-eliminate`. In this case, Alice has k inputs $x_1, \ldots, x_k$, however, Bob has *one* input y. The goal of Alice and Bob is to find a vector $\sigma_1, \ldots, \sigma_k$ such that $f_i(x_i, y) \neq \sigma_i$, for at least one i. In the rest of this section, we assume that $f_1 = f_2 = \ldots = f_k = f$. In this model significant saving is possible even for $k = 2$. For example for `r-eliminate` of the equality function, if Alice holds two inputs $x_1 = x_2$, she can eliminate, say, $(0, 1)$ without any communication, and if $x_1 \neq x_2$, she can eliminate $(1, 1)$ without any communication. We show that for some other functions `r-eliminate` is hard and we also give additional examples where some saving is possible:

- `r-eliminate` of the disjointness function on k instances can be computed using a deterministic protocol sending $(n \log k)/k$ bits; that is, better than

the deterministic complexity of solving one instance of disjointness (which is n). By [2], `eliminate` of k instances of disjointness requires $n - O(\log n)$ bits deterministically [2]. Using this result, we prove that `r-eliminate` of disjointness on k instances requires $\Omega(n/k)$ bits deterministically. That is, our protocol is optimal up to a factor of $\log k$.
- `r-eliminate` of the inner-product function on k instances can be solved deterministically by sending $n - k + 2$ bits. Thus, some saving is possible for large k's. We show that our lower bound for $\mathtt{eliminate}_{\mathrm{IP}^k}$ can be translated to a lower bound of $\Omega(n/k)$ for `r-eliminate` on k instances of IP for randomized protocols.
- For most functions, `r-eliminate` of two instances requires at least $n - 5$ bits. Thus, the naive protocol where Bob sends his input to Alice is nearly optimal.

In the full version of this paper, we also consider decision trees and circuit complexity. We show that for decision trees, no saving can be obtained for `choose`, `agree`, and `eliminate`. That is, any decision tree solving $\mathtt{eliminate}_{f,g}$ can be converted into a decision tree of the same size and depth that either computes f or computes g. This generalizes the results that no saving can be achieved for decision trees in the direct sum problem [7, 22, 27, 14]. We also prove that the size of the smallest circuit solving $\mathtt{agree}_{f,g}$ is equal to the deterministic communication complexity of $\mathtt{choose}_{R_f,R_g}$, where R_f denotes the Karchmer-Wigderson relation related to f [18]. This is a generalization of results of [18] on the relation between circuit size of a function f and the communication complexity of R_f.

2 Preliminaries

We consider the two-party communication complexity model of Yao [30]. In this model, there are three finite sets X, Y, and Z, and a relation $R \subseteq X \times Y \times Z$. Two players, Alice and Bob, get $x \in X$ and $y \in Y$ respectively. Their goal is to compute z such that $(x, y, z) \in R$ by exchanging bits according to some protocol (we assume that for every x, y there is some z such that $(x, y, z) \in R$). Let $\mathcal{D}(R)$ be the deterministic communication complexity of solving R, $\mathcal{N}(R)$ be the nondeterministic communication complexity, $\mathcal{N}^0(R)$ and $\mathcal{N}^1(R)$ be the one-sided nondeterministic communication complexities, and $\mathcal{R}_\epsilon(R)$ and $\mathcal{R}_\epsilon^{\mathrm{pub}}(R)$ be its ϵ-error randomized communication complexity with private and public random string, respectively. For formal definitions, see [21].

Next, we define the three problems inspired by the direct sum question. For the three problems there are several possible outputs, i.e., they are relations.

Definition 1 (Choose, Agree, and Eliminate). *Let $f_1, \ldots, f_k : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be functions. In all three problems, the input of Alice and Bob are a k-tuple $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ respectively. In $\mathtt{choose}_{f_1,\ldots,f_k}$ an output is $(i, f_i(x_i, y_i))$ where $i \in \{1, \ldots, k\}$. In $\mathtt{agree}_{f_1,\ldots,f_k}$, an output is $f_i(x_i, y_i)$ where $i \in \{1, \ldots, k\}$. Finally, in $\mathtt{eliminate}_{f_1,\ldots,f_k}$, an output is any vector $\sigma_1, \ldots, \sigma_k \in \{0,1\}$ for which there exists i such that $\sigma_i \neq f_i(x_i, y_i)$.*

If $f_1 = ... = f_k = f$ we abbreviate $f_1, \ldots, f_k$ by f^k. Note that $\mathcal{R}_\epsilon(\texttt{eliminate}_{f^k}) = O(1)$ for $\epsilon \geq 1/2^k$ as a random k-bit output will err with probability $1/2^k$. As we shall see that, for $\epsilon < 1/2^k$, $\mathcal{R}_\epsilon(\texttt{eliminate}_{f^k})$ is large for some functions.

Next, we define a restricted version of `eliminate` where Bob gets the same y for all k functions. For simplicity, assume that all k functions are equal.

Definition 2 (R-Eliminate). *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. In* $\texttt{r-eliminate}_{f^k}$*, Alice gets a k-tuple $x_1, \ldots, x_k$ and Bob gets a* single *input y; an output is any vector $\sigma_1, \ldots, \sigma_k \in \{0,1\}$ for which there exists i such that $\sigma_i \neq f(x_i, y)$.*

The next theorem relates the randomized communication complexity and the distributional communication complexity of f.

Theorem 1 (Yao's min-max Theorem [30]). *Let $f : X \times Y \to \{0,1\}$. Then, $\mathcal{R}_\epsilon^{\text{pub}}(f) = \max_\mu \mathcal{D}_\epsilon^\mu(f)$, where $\mathcal{D}_\epsilon^\mu$ denotes the ϵ-error* distributional *complexity with respect to μ and the maximum is taken over all distributions μ on $X \times Y$.*

3 General Bounds for Choose, Agree, and Eliminate

In this section, we show that saving is possible in public-coin randomized protocols for eliminate of some functions. We then prove lower bounds on the complexity of $\texttt{choose}_{f_1,f_2}$, $\texttt{agree}_{f_1,f_2}$, and $\texttt{eliminate}_{f^k}$.

Theorem 2. *There exist a randomized protocol for* $\texttt{eliminate}_{f^k}$ *with complexity $\max\{n - 0.5\log k, k\log k\} + O(1)$ for $\epsilon = 2^{-k}/e$, in which at least one of the parties knows the answer at the end.*

Proof. We describe a protocol with the desired complexity. In the first step of the protocol, Bob checks if he has at least $\sqrt{k}$ distinct inputs (among his k inputs). If so, both Alice and Bob treat the public random string r as a sequence of blocks $r_1, r_2, \ldots$, each of length n. If there exists i and j such that $y_i = r_j$, among the first $2^n/\sqrt{k}$ blocks of r, Bob sends j to Alice. In this case, Alice computes $\sigma_\ell = 1 - f(x_\ell, r_j)$ for $1 \leq \ell \leq k$ and outputs $\sigma_1, \ldots, \sigma_k$. If Bob cannot find such index, he sends 0 to Alice, who outputs a random k-bit string. If Bob has less than $\sqrt{k}$ distinct inputs, and Alice has at least $\sqrt{k}$ distinct inputs, they reverse roles. In this case, Bob gets the output. In both cases, the communication complexity is $O(1) + \log(2^n/\sqrt{k}) = n - 0.5\log k + O(1)$.

If both Alice and Bob have less than $\sqrt{k}$ distinct input values (they discover this fact using O(1) communication) then they do the following. Since Bob has less than $\sqrt{k}$ values, there is a value that appears more than $\sqrt{k}$ times. Bob sends to Alice $\sqrt{k}$ indices in which his inputs are the same. Since Alice has less than $\sqrt{k}$ values, there are two indices i, j among the indices that Bob sent such that $x_i = x_j$. Alice outputs an arbitrary vector $(\sigma_1, \ldots, \sigma_k)$ such that $\sigma_i \neq \sigma_j$ which is always correct, and the communication complexity is $\sqrt{k}\log k + O(1)$.

To complete the analysis of the protocol, we bound the error probability for the cases that Bob (or Alice) has at least $\sqrt{k}$ distinct input values. The probability that $y_i \neq r_j$ for all $1 \leq i \leq k$ and $1 \leq j \leq 2^n/\sqrt{k}$ is less than $(1-1/2^n)^{\sqrt{k}2^n/\sqrt{k}} \leq 1/e$. In this case, the protocol errs with probability $1/2^k$. If $y_i = r_j$ then the protocol never errs. Thus, the error probability of the protocol is $2^{-k}/e$. □

Next we state lower bounds for $\texttt{choose}_{f_1,f_2}$ and $\texttt{agree}\, f_1, f_2$ in terms of the deterministic and non-deterministic communication complexity of f_1 and f_2, the proof for `choose` appears in the final version.

Theorem 3. $\mathcal{D}(\texttt{choose}_{f_1,f_2}) \geq \min\{\mathcal{D}(f_1), \mathcal{D}(f_2), \max\{\mathcal{N}(f_1), \mathcal{N}(f_2)\}\}$ *for every two functions* f_1, f_2.

Theorem 4. *For every two functions* $f_1, f_2 : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,
$\mathcal{D}(\texttt{agree}_{f_1,f_2}) \geq \max\left\{\min\left\{\mathcal{N}^1(f_1), \mathcal{N}^0(f_2)\right\}, \min\left\{\mathcal{N}^0(f_1), \mathcal{N}^1(f_2)\right\}\right\} - \log(2n)$.

Proof. Let $\mathcal{P}$ be a (deterministic) protocol for $\texttt{agree}_{f_1,f_2}$ whose complexity is $\mathcal{D}(\texttt{agree}_{f_1,f_2})$. We construct a non-deterministic protocol with communication complexity $\mathcal{D}(\texttt{agree}_{f_1,f_2}) + \log(2n)$ either for proving $f_1(x,y) = 0$ or for proving $f_2(x,y) = 1$. Let $S_1 \subseteq f_1^{-1}(0)$ and $S_2 \subseteq f_2^{-1}(1)$. A pair of inputs $(x_2, y_2) \in S_2$ is an f_1-loser if $\mathcal{P}(x_1, x_2, y_1, y_2) = 0$ for at least $|S_1|/2$ pairs $(x_1, y_1) \in S_1$.

Proposition 1. *For every two sets* $S_1 \subseteq f_1^{-1}(0)$ *and* $S_2 \subseteq f_2^{-1}(1)$*, either there is a pair* $(x_2, y_2) \in S_2$ *that is an* f_1*-loser or a pair* $(x_1, y_1) \in S_1$ *that is an* f_2*-loser.*

Proposition 2. *Either there is a sequence of* $2n$ *pairs* $(x^1, y^1), \ldots, (x^{2n}, y^{2n}) \in f_2^{-1}(1)$ *s.t., for every* $(x,y) \in f_1^{-1}(0)$*, it holds that* $\mathcal{P}(x, x^i, y, y^i) = 0$ *for at least one* i*, or there is a sequence of* $2n$ *pairs* $(x^1, y^1), \ldots, (x^{2n}, y^{2n}) \in f_1^{-1}(0)$ *s.t., for every* $(x,y) \in f_2^{-1}(1)$*, it holds that* $\mathcal{P}(x^i, x, y^i, y) = 1$ *for at least one* i*.*

Assume that the first case of Proposition 2 holds. We construct a non-deterministic protocol for proving that $f_1(x,y) = 0$ (if the second case holds, we would construct a non-deterministic protocol for proving $f_2(x,y) = 1$). Let $(x^1, y^1), \ldots, (x^{2n}, y^{2n})$ be the sequence guaranteed by the proposition. The first idea is, given inputs x, y to f_1, to execute $\mathcal{P}(x, x^i, y, y^i)$ for $i = 1, \ldots, 2n$. If in at least one of the executions, the output of $\mathcal{P}$ is 0 then clearly $f_1(x,y) = 0$. Furthermore, if $f_1(x,y) = 0$, then at least one of the executions will return 0. This protocol activates $\mathcal{P}$ $2n$ times, and is, thus, extremely inefficient. However, Alice can guess an index i such that $\mathcal{P}(x, x^i, y, y^i) = 0$, send it to Bob, and Alice and Bob execute $\mathcal{P}(x, x^i, y, y^i)$, and output 0 iff $\mathcal{P}$ outputs 0. Therefore, $\mathcal{D}(\texttt{agree}_{f_1,f_2}) + \log 2n \geq \min\left\{\mathcal{N}^0(f_1), \mathcal{N}^1(f_2)\right\}$. Similarly, $\mathcal{D}(\texttt{agree}_{f_1,f_2}) + \log 2n \geq \min\left\{\mathcal{N}^1(f_1), \mathcal{N}^0(f_2)\right\}$. □

Theorem 4 does not rule out an exponential gap between $\mathcal{D}(\texttt{agree}_{f_1,f_2})$ and $\min\{\mathcal{D}(f_1), \mathcal{D}(f_2)\}$. For the special case, $\texttt{agree}_{f,\overline{f}}$, Theorem 4 implies that the gap is at most a quadratic; i.e., $\mathcal{D}(\texttt{agree}_{f,\overline{f}}) \geq \mathcal{N}(f) - \log(2n) - 1 \geq \Omega(\sqrt{\mathcal{D}(f)}) - \log(2n)$. This should be compared, on one hand, to $\texttt{choose}_{f,\overline{f}}$, which is as hard as computing f and, on the other hand, to solving $\texttt{eliminate}_{f,\overline{f}}$, which is equal to solving $\texttt{eliminate}_{f^2}$. Furthermore, $\texttt{agree}_{f,f}$ is equivalent to computing f.

We end this section by stating a lower bound for $\texttt{eliminate}_{f^k}$ using the randomized communication complexity of f.

Theorem 5. *$\mathcal{R}^{\text{pub}}_{\epsilon}(\texttt{eliminate}_{f^k}) \geq \mathcal{R}^{\text{pub}}_{\epsilon'}(f)$, where $\epsilon' = \frac{1}{2} - \frac{1/2 - \epsilon 2^{k-1}}{2^k - 1}$.*

Proof. We prove the lower bound using Yao's min-max Theorem (Theorem 1). Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function and μ be a distribution on $\{0,1\}^n \times \{0,1\}^n$ such that $\mathcal{R}^{\text{pub}}_{\epsilon}(f) = \mathcal{D}^{\mu}_{\epsilon}(f)$. We start with a randomized protocol $\mathcal{P}$ for $\texttt{eliminate}_{f^k}$ with complexity $\mathcal{R}_{\epsilon}(\texttt{eliminate}_{f^k})$. We construct a deterministic protocol $\mathcal{P}''$ with the same complexity as $\mathcal{P}$ such that the probability over the inputs, according to μ, that $\mathcal{P}''$ computes $f(x,y)$ correctly is at least ϵ'. Thus, $\mathcal{R}_{\epsilon}(\texttt{eliminate}_{f^k}) \geq \mathcal{D}^{\mu}_{\epsilon'}(f) = \mathcal{R}^{\text{pub}}_{\epsilon'}(f)$. The construction of $\mathcal{P}''$ is done in stages. We first define a protocol $\mathcal{P}'$ and use it to define $\mathcal{P}''$.

For the construction of $\mathcal{P}''$, we use constants $q_0, \ldots, q_k$ such that $1 = q_0 \geq q_1 \geq \cdots \geq q_k = \epsilon$. We also use random variables $Z_1, \ldots, Z_k$ defined as follows. Pick inputs $x_1, y_1, \ldots, x_k, y_k$ according to the distribution μ^k, execute $\mathcal{P}(x_1, \ldots, x_k, y_1, \ldots, y_k)$ and let $\sigma_1, \ldots, \sigma_k$ be its output. Finally, for $1 \leq i \leq k$, define $Z_i = T$ if $\sigma_i = f(x_i, y_i)$ and $Z_i = F$ otherwise.

By the correctness of $\mathcal{P}$, $\Pr[Z_1 = \cdots = Z_k = T] \leq \epsilon = q_k$, where the probability is taken over the choice of inputs, according to the distribution μ^k, and over the randomness of $\mathcal{P}$. Thus, there must be an index i, where $1 \leq i \leq k$, such that $\Pr[Z_i = T | Z_1 = \cdots = Z_{i-1} = T] \leq q_i/q_{i-1}$. Let i be the smallest such index. That is, $\Pr[Z_j = T | Z_1 = \cdots = Z_{j-1} = T] \geq q_j/q_{j-1}$ for $1 \leq j \leq i-1$. In particular, $p \stackrel{\text{def}}{=} \Pr[Z_1 = \cdots = Z_{i-1} = T] = \prod_{1 \leq j \leq i-1} \Pr[Z_j = T | Z_1 = \cdots = Z_{j-1} = T] \geq (q_1/q_0) \cdot (q_2/q_1) \cdot \ldots \cdot (q_{i-1}/q_{i-2}) = q_{i-1}$.

Using this index i, we construct a randomized protocol $\mathcal{P}'$ for computing f. Intuitively, $\mathcal{P}'$ will use the fact that with a noticeable probability it can take the i-th output of $\mathcal{P}$ and invert it and obtain a correct output for the i-th pair of inputs (assuming they are distributed according to μ). On input, x, y, the protocol $\mathcal{P}'$ samples $x_1, y_1, \ldots, x_{i-1}, y_{i-1}, x_{i+1}, y_{i+1}, \ldots, x_k, y_k$ according to μ^{k-1} and gives these inputs both to Alice and to Bob (we will see later how to implement this step without communication). Alice and Bob execute $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} = (x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k)$ and $\boldsymbol{y} = (y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_k)$; let $\sigma_1, \ldots, \sigma_k$ be the output of $\mathcal{P}$. If $\sigma_j = f(x_j, y_j)$ for $1 \leq j \leq i-1$, then Alice and Bob output $\overline{\sigma}_i$. Otherwise, Alice chooses a random bit b with uniform distribution and outputs this bit.

We next claim that if the inputs x, y are distributed according to μ, then the error of $\mathcal{P}'$ is at most ϵ'. Protocol $\mathcal{P}'$ succeeds in two cases: (1) $\sigma_j = f(x_j, y_j)$ for $1 \leq j \leq i-1$ and $\sigma_i \neq f(x,y)$; by our choice of i, this happens with probability at least $p \cdot (1 - q_i/q_{i-1})$ and (2) $\sigma_j \neq f(x_j, y_j)$ for some $1 \leq j \leq i-1$ and $b = f(x,y)$; this happens with probability $0.5(1-p)$. All together the success probability of $\mathcal{P}'$ is $p \cdot (1 - q_i/q_{i-1}) + 0.5(1-p)$, where the probability is taken over the choice of the inputs x, y according to μ, the choice of the other $k-1$ pairs of inputs for $\mathcal{P}$ according to μ^{k-1}, the randomness of $\mathcal{P}$, and the choice of b. Since $p \leq q_{i-1}$ and by choosing q_i, q_{i-1} such that $q_i/q_{i-1} \leq 0.5$, the success probability is at least $q_{i-1} \cdot (1 - q_i/q_{i-1}) + 0.5(1 - q_{i-1}) = 0.5 + 0.5q_{i-1} - q_i$. We choose $q_0 \stackrel{\text{def}}{=} 1$ and $q_j \stackrel{\text{def}}{=} \frac{2^{j-1}}{2^j-1}\epsilon' - (1 - \frac{1}{2^{j-1}})$ for $1 \le j \le k$. Notice that $q_1 = \epsilon'$, $q_k = \epsilon$, and the success probability of $\mathcal{P}'$ is at least $1 - \epsilon'$.

Protocol $\mathcal{P}'$ is randomized and we want a deterministic protocol $\mathcal{P}''$. Furthermore, we need to explain how we can assume that Alice and Bob know all the other $k-1$ pairs of inputs. The derandomization of $\mathcal{P}'$ is done using a simple averaging argument: there exists a random string for $\mathcal{P}'$ such that the success probability of $\mathcal{P}'$ with this random string is at least $1 - \epsilon'$, where now the success probability is taken over the choice of x, y according to μ. We fix such random string to obtain $\mathcal{P}''$. As this random string contains $x_1, y_1, \ldots, x_{i-1}, y_{i-1}, x_{i+1}, y_{i+1}, \ldots, x_k, y_k$, and $\mathcal{P}''$ executes $\mathcal{P}'$ with the fixed random string, both Alice and Bob know these inputs. Thus, as the communication complexity of $\mathcal{P}''$ is the same as the communication complexity of $\mathcal{P}$, the theorem follows. $\square$

Together with a simple protocol for eliminate, which solves one instance and guesses the other coordinates, we have that $\mathcal{R}_{\epsilon \cdot 2^{k-1}}(f) \ge \mathcal{R}_\epsilon(\texttt{eliminate}_{f^k}) \ge \mathcal{R}_{1/2 - \frac{1/2 - \epsilon 2^{k-1}}{2^k - 1}}(f) \ge 2^{-O(k)}\mathcal{R}_\epsilon(f)$ for $\epsilon < 1/2^k$. In other words, $\mathcal{R}_\epsilon(f)$ characterizes $\mathcal{R}_\epsilon(\texttt{eliminate}_{f^k})$ up to a factor of $2^{O(k)}$.

4 Eliminate and R-Eliminate

We consider the `r-eliminate` task, where Alice gets a sequence of inputs $x_1, \ldots, x_k$ and Bob gets a single input y, and their goal is to compute some impossible outcome for $f(x_1, y), \ldots, f(x_k, y)$. In this model, we have the largest gap possible between $\mathcal{D}(f)$ and $\mathcal{D}(\texttt{r-eliminate}_{f^2})$.

Example 1. Consider the equality function, where $\mathrm{EQ}(x, y) = 1$ iff $x = y$. It is known that $\mathcal{D}(\mathrm{EQ}) = n$. However, $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{EQ}^2}) = O(1)$: if $x_1 = x_2$ Alice outputs $(0, 1)$ (if $x_1 = y$ then so is x_2). Otherwise, if $x_1 \ne x_2$, Alice outputs $(1, 1)$ (since x_1, x_2 cannot both equal y). By [2], $\mathcal{D}(\texttt{eliminate}_{\mathrm{EQ}^k}) = \Omega(n)$, thus we also get the largest possible separation between `r-eliminate` and `eliminate`.

Eliminate vs. R-Eliminate. We next identify a simple property of functions that enables proving that the communication complexity of `r-eliminate` and `eliminate` can differ by a factor of at most $1/k$ for these functions.

Definition 3. *A function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is padable with respect to a function $g : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$, where $m > n$, if there exists a value $b \in \{0,1\}$ such that $f(x, y) = g(b^i \circ x \circ b^{m-n-i}, a \circ y \circ a')$, for every $i \le m - n$, every $x, y \in \{0,1\}^n$, and every $a \in \{0,1\}^i$, $a' \in \{0,1\}^{m-n-i}$.*

We shall see that natural functions, e.g., disjointness and inner-product, are padable with respect to themselves (by taking $b = 0$).

Lemma 1. *If a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is padable with respect to $g : \{0,1\}^{nk} \times \{0,1\}^{nk} \to \{0,1\}$, then $\mathcal{D}(\texttt{r-eliminate}_{g^k}) \ge \mathcal{D}(\texttt{eliminate}_{f^k})$ and $\mathcal{R}_\epsilon(\text{r-}\,\texttt{eliminate}_{g^k}) \ge \mathcal{R}_\epsilon(\texttt{eliminate}_{f^k})$.*

Proof. Let $x_1, \ldots, x_k \in \{0,1\}^n$ and $y_1, \ldots, y_k \in \{0,1\}^n$ be the inputs of Alice and Bob (respectively) for $\texttt{eliminate}_{f^k}$. Let $y' = y_1 \circ \cdots \circ y_k$. As f is padable with respect to g (with some b), we can pad each input x_i of Alice to get $x'_i = b^{n(i-1)} x_i b^{n(k-i)}$ (of length nk). We execute an optimal deterministic protocol for $\texttt{r-eliminate}_{g^k}$ on inputs $(x'_1, \ldots x'_k, y')$, and let τ be the answer of the eliminate protocol. Since f is padable with respect to g, we have $f(x_i, y_i) = g(x'_i, y')$ and so the answer τ is a possible answer for $\texttt{eliminate}_{f^k}(x_1, \ldots, x_k, y_1, \ldots, y_k)$ as well. Hence, $\mathcal{D}(\texttt{eliminate}_{f^k}) \leq \mathcal{D}(\texttt{r-eliminate}_{g^k})$. The same reduction applies for the randomized case. □

R-Eliminate for the Disjointness Function. Let DISJ denote the disjointness function, namely $\mathrm{DISJ}(S,T) = 1$ iff $S \cap T = \emptyset$ for $S, T \subseteq [n]$ (the inputs sets S, T are represented by their n-bit characteristic vectors).

Theorem 6. $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{DISJ}^k}) = O(\frac{n}{k} \cdot \log k)$.

Proof. Let $S_1, \ldots, S_k$ be the inputs of Alice and T be the input of Bob. For $1 \leq i \leq k$ define $A_i \stackrel{\text{def}}{=} S_i \setminus \cup_{i \neq j} S_j$; that is, A_i contains the elements that appear only in S_i. Let A_j be a smallest set among $A_1, \ldots, A_k$; that is, $|A_j| \leq |A_i|$ for $1 \leq i \leq k$. Since $A_1, \ldots, A_k$ are disjoint, $|A_j| \leq n/k$. To solve $\texttt{r-eliminate}_{\mathrm{DISJ}^k}$, Alice sends the set A_j to Bob. Bob computes $\mathrm{DISJ}(A_j, T)$ and sends the answer to Alice. Alice computes the output as follows: If $\mathrm{DISJ}(A_j, T) = 0$, then A_j intersects T, and, in particular, S_j intersects T. Therefore, in this case, Alice may return any vector whose j-th coordinate is 1. If $\mathrm{DISJ}(A_j, T) = 1$, then either S_j and T are disjoint, or $S_j \setminus A_j$ intersects T, and therefore S_i intersects T for at least one $i \neq j$ (since any element in $S_j \setminus A_j$ belongs to some S_i). Therefore, the vector whose j-th coordinate is 0 and all other coordinates are 1 is not possible and Alice outputs this vector.

The number of sets of size at most n/k is at most $(ek)^{n/k}$. Therefore, communicating the set A_j requires at most $\frac{n}{k} \log(ek) \leq (n/k)(\log k + 2)$ bits. □

Ambainis et al. [2] showed that $\mathcal{D}(\texttt{eliminate}_{\mathrm{DISJ}^k}) = \Omega(n)$. Using the fact that DISJ with n/k-bit inputs is padable with respect to DISJ with n-bit inputs and Lemma 1, we can obtain lower bounds on the complexity of r-eliminate of disjointness, that is, $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{DISJ}^k}) = \Omega(n/k)$. In particular, we deduce that the protocol of Theorem 6 is optimal up to factor of $\log k$ for deterministic protocols. Theorem 5 and Lemma 1 imply the lower bound $\mathcal{R}_\epsilon(\text{r-}\,\texttt{eliminate}_{\mathrm{DISJ}^k}) = n/2^{O(k)}$ for $\epsilon < 1/2^{k+1}$.

R-Eliminate for the Inner Product function. Let $\mathrm{IP} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the inner product mod 2 function, i.e. $\mathrm{IP}(x,y) = x[1] \cdot y[1] \oplus, \ldots, \oplus x[n] \cdot y[n]$, where $x[i]$ and $y[i]$ are the i-th bits of x and y respectively. In the full version of this paper, we show that some small saving is possible for `r-eliminate` of IP; namely, $\mathcal{D}(\texttt{r-eliminate}_{\mathrm{IP}^k}) \leq \max\{n - k + 2, 0\}$. We next prove a lower bound on the randomized communication complexity of $\texttt{eliminate}_{\mathrm{IP}^k}$. Specifically, we prove that $\mathcal{R}_\delta(\texttt{eliminate}_{\mathrm{IP}^k}) \geq n - O(k)$, for $\delta < \frac{1}{4k \cdot 2^{2k}}$. This improves over a lower bound of $n/2^{O(k)} \log n \log\log n$ from [2]. Notice that for $k \geq \log n$ their bound is $\Omega(1)$, while even for $k = o(n)$ our bound is $\Omega(n)$. The lower bound for IP can also be obtained from Theorem 5; we present the proof below since we believe that its ideas might be of interest.

Theorem 7. *$\mathcal{R}_\delta(\texttt{eliminate}_{\mathrm{IP}^k}) \geq n - O(k)$, for $\delta < 1/(4k^{3/2}2^k + 2)$.*

Proof. We will show that if there is a δ-error protocol $\mathcal{P}$ for $\texttt{eliminate}_{\mathrm{IP}^k}$ with complexity less than $n - O(k)$, then there is a randomized protocol for IP on inputs of length nk with error less than $1/2 - \epsilon$ with complexity less than $nk - \log O(1/\epsilon)$ contradicting the known lower bound for IP. Given $x, y \in \{0,1\}^{nk}$, the protocol for $\mathrm{IP}(x, y)$, denoted $\mathcal{P}'$, proceeds as follows:

1. Let $x = x_1, \ldots, x_k$ and $y = y_1, \ldots, y_k$, where $|x_i| = |y_i| = |n|$. Alice and Bob execute the protocol $\mathcal{P}$ for $\texttt{eliminate}_{\mathrm{IP}^k}$ on $(x_1, \ldots, x_k, y_1, \ldots, y_k)$. Let $(\sigma_1, \ldots, \sigma_k)$ be the output of $\mathcal{P}$. Denote $\alpha_i = \overline{\sigma_i}$ for $1 \leq i \leq k$ (with probability at least $1 - \delta$, $\mathrm{IP}(x_i, y_i) = \alpha_i$ for at least one i).
2. Alice chooses uniformly at random an index $1 \leq j \leq k$ and sends j and $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_k$ to Bob.
3. Bob computes $\beta_i = \mathrm{IP}(x_i, y_i)$, for $i \in \{1, \ldots, k\} \setminus \{j\}$, computes $a = \oplus_{i \neq j} \beta_i$, and $c = |\{i \neq j : \alpha_i = \beta_i\}|$. It computes the protocol's output as follows:
   - if $c = 0$ (that is, $\alpha_i \neq \beta_i$ for every $i \in \{1, \ldots, k\} \setminus \{j\}$), the output is $a \oplus \alpha_j$ (in this case if $\mathcal{P}$ returns a correct output, then it must be that $\alpha_j = \beta_j$ and the output of $\mathcal{P}'$ is correct).
   - if $c > 0$, then with probability $1/2 + \epsilon_c$ the output is $a \oplus \alpha_j$ and with probability $1/2 - \epsilon_c$ it is $a \oplus \overline{\alpha_j}$, where ϵ_c will be determined later.

We analyze the success probability of the protocol $\mathcal{P}'$. First, assume that $\mathcal{P}$ never errs. We will later remove this assumption. Let $m = |\{i : \alpha_i = \mathrm{IP}(x_i, y_i)\}|$; that is, m is the number of correct values among $\alpha_1, \ldots, \alpha_k$.

The case $m = 1$: If Alice chooses the unique j such that $\alpha_j = \mathrm{IP}(x_j, y_j)$, then $c = 0$ and $\mathcal{P}'$ always succeeds. If Alice chooses any other j, then $\alpha_j \neq \mathrm{IP}(x_j, y_j)$ and $c = 1$, and so the protocol $\mathcal{P}'$ succeeds with probability $1/2 - \epsilon_1$. All together, the success probability, in this case, is:

$$\frac{1}{k} + \frac{k-1}{k}\left(\frac{1}{2} - \epsilon_1\right) \geq \frac{1}{2} + \frac{1}{2k} - \epsilon_1.$$

The case $2 \leq m \leq k$: If Alice chooses an index j such that $\alpha_j = \mathrm{IP}(x_j, y_j)$ (this happens with probability m/k), then $c = m - 1$ and Bob outputs the correct value (i.e., $a \oplus \alpha_j$) with probability $1/2 + \epsilon_{m-1}$. If Alice chooses j such that $\alpha_j \neq \mathrm{IP}(x_j, y_j)$ (with probability $(1 - m/k)$), then $c = m$ and Bob outputs the correct value (i.e., $a \oplus \overline{\alpha}_j$) with probability $1/2 - \epsilon_m$. All together, the success probability, in this case, is

$$\frac{m}{k}\left(\frac{1}{2} + \epsilon_{m-1}\right) + \left(1 - \frac{m}{k}\right)\left(\frac{1}{2} - \epsilon_m\right) = \frac{1}{2} + \frac{m}{k}\epsilon_{m-1} - \frac{k-m}{k}\epsilon_m.$$

Set $\epsilon_m = 1/(4\binom{k}{m})$. Thus, for $2 \leq m \leq k$, the success probability is:

$$\frac{1}{2} + \frac{1}{k}\left(\frac{m}{4\binom{k}{m-1}} - \frac{k-m}{4\binom{k}{m}}\right) = \frac{1}{2} + \frac{1}{4k\binom{k}{m}}.$$

For $m = 1$ the success probability is greater than $1/2 + 1/(4k)$. All together, $\mathcal{P}'$ succeeds with probability greater than $1/2 + \frac{1}{4k\binom{k}{k/2}} \geq 1/2 + 1/(4k^{3/2}2^k)$ (since $\binom{k}{k/2} \leq 2^k/k^{1/2}$).

Next, assume that $\mathcal{P}$ errs with probability (at most) δ. In the worst case, $\mathcal{P}'$ fails whenever $\mathcal{P}$ fails. The success probability of $\mathcal{P}'$ is, therefore, at least $(1-\delta) \cdot (1/2 + 1/(4k^{3/2}2^k))$. Assuming that $\delta \leq 1/(4k^{3/2}2^k + 2)$, the success probability of $\mathcal{P}'$ is at least $1/2 + 1/(8k2^{2k})$.

The communication complexity of $\mathcal{P}'$ is the communication complexity of $\mathcal{P}$, on inputs of length n, plus $(1-1/k)nk$. By [21, Exercise 3.30], the communication complexity of IP with error $1/2 - \epsilon$ on inputs of length nk is at least $nk - O(\log 1/\epsilon)$. Thus, we get $\mathcal{R}_\delta(\texttt{eliminate}_{\mathrm{IP}^k}) + (1-1/k) \cdot nk \geq nk - O(k)$, which implies that $\mathcal{R}_\delta(\texttt{eliminate}_{\mathrm{IP}^k}) \geq n - O(k)$. $\square$

As IP with n/k-bit inputs is padable with respect to IP with n-bit inputs (using $b = 0$ in Definition 3) then, using Lemma 1, we get:

Corollary 1. $\mathcal{R}_\delta(\text{r-}\texttt{eliminate}_{\mathrm{IP}^k}) \geq \frac{n}{k} - O(1)$ *for every* $\delta < 1/(4k^{3/2}2^k + 2)$.

We know that $\mathcal{R}_{1/2^k}(\texttt{eliminate}_{\mathrm{IP}^k}) = O(1)$. Thus, the error that we allow in Theorem 7 (and Corollary 1) is nearly optimal.

Eliminate of Most Functions. In the full version of the paper, we prove that for most functions f `r-eliminate` cannot be computed efficiently; i.e., we prove that $\mathcal{D}(\texttt{r-eliminate}_{f^2}) \geq n - 5$ for most functions.

References

1. M. Agrawal and V. Arvind. Polynomial time truth-table reductions to P-selective sets. In *Structure in Complexity Theory Conference*, pages 24–30, 1994.
2. A. Ambainis, H. Buhrman, W. Gasarch, B. Kalyanasundaramd, and L. Torenvliete. The communication complexity of enumeration, elimination, and selection. *JCSS*, 63(2):148–185, 2001.
3. A. Amihood, R. Beigel, and W. I. Gasarch. Some connections between bounded query classes and non-uniform complexity. *ECCC*, 7(024), 2000.
4. Z. Bar-Yossef, T. S. Jayram, R Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *the Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 209–218, 2002.
5. R. Beigel, W. I. Gasarch, J. Gill, and J. C. Owings. Terse, superterse, and verbose sets. *Inf. Comput.*, 103(1):68–85, 1993.
6. R. Beigel, W. I. Gasarch, M. Kummer, G. Martin, T. McNicholl, and F. Stephan. The complexity of Odd_n^A. *J. Symb. Log.*, 65(1):1–18, 2000.

7. R. Beigel and T. Hirst. One help-bit doesn't help. In *the Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 124–130, 1998.
8. R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Information and Computation*, 120:12–23, 1994.
9. J. Cai and L. A. Hemachandra. Enumerative counting is hard. *Inf. Comput.*, 82(1):34–44, 1989.
10. T. Feder, E. Kushilevitz, M. Naor, and N. Nisan. Amortized communication complexity. *SIAM Journal on Computing*, 24(4):736–750, 1995.
11. G. Galbiati and M. J. Fischer. On the complexity of 2-output Boolean networks. *Theor. Comput. Sci.*, 16:177–185, 1981.
12. W. Gasarch and G. Martin. *Bounded queries in recursion theory.* 1998.
13. L. A. Hemaspaandra and L. Torenvliet. *Theory of Semi-Feasible Algorithms.* 2003.
14. R. Jain, H. Klauck, and M. Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. Technical Report 1004.0105v1, arxiv.org, 2010. `http://arxiv.org/abs/1004.0105v1`.
15. C. Jockusch. Semirecursive sets and positive reducibility. *Transactions of the American Mathematical Society*, 131(2):420–436, 1968.
16. M. Karchmer, E. Kushilevitz, and N. Nisan. Fractional covers and communication complexity. *SIAM J. on Discrete Mathematics*, 8(1):76–92, 1995.
17. M. Karchmer, R. Raz, and A. Wigderson. On proving superlogarithmic depth lower bounds via the direct sum in communication complexity. In *6th IEEE Structure in Complexity Theory*, pages 299–304, 1991.
18. M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. on Discrete Mathematics*, 3(2):255–265, 1990.
19. K. Ko. On self-reducibility and weak P-selectivity. *JCSS*, 26(2):209–221, 1983.
20. M. Kummer. A proof of Beigel's cardinality conjecture. *J. Symb. Log.*, 57(2):677–681, 1992.
21. E. Kushilevitz and N. Nisan. *Communication Complexity.* 1997.
22. N. Nisan, S. Rudich, and M. Saks. Products and help bits in decision trees. *SIAM J. Comput.*, 28(3):1035–1050, 1999.
23. W. J. Paul. Realizing Boolean functions on disjoint sets of variables. Technical report, Cornell University, Ithaca, NY, USA, 1974.
24. A. L. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. In *6th ICALP*, pages 546–555. Springer-Verlag, 1979.
25. A. L. Selman. Analogues of semicursive sets and effective reducibilities to the study of NP complexity. *Information and Control*, 52(1):36–51, 1982.
26. A. L. Selman. Reductions on NP and P-selective sets. *Theor. Comput. Sci.*, 19:287–304, 1982.
27. R. Shaltiel. Towards proving strong direct product theorems. In *Proc. of the 16th IEEE Conf. on Computational Complexity*, pages 107–119, 2001.
28. D. Sivakumar. On membership comparable sets. *J. Comput. Syst. Sci.*, 59(2):270–280, 1999.
29. D. Uhlig. On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. *Mat. Zametki*, 15:937–944, 1974.
30. A. C. Yao. Some complexity questions related to distributed computing. In *Proc. of the 11th ACM Symp. on the Theory of Computing*, pages 209–213, 1979.