Quantum and Classical Strong Direct Product Theorems and Optimal Time-Space Tradeoffs

Hartmut Klauck[*]
University of Calgary
klauckh@cpsc.ucalgary.ca

Robert Špalek[†]
CWI, Amsterdam
sr@cwi.nl

Ronald de Wolf[†]
CWI, Amsterdam
rdewolf@cwi.nl

Abstract

A strong direct product theorem says that if we want to compute k independent instances of a function, using less than k times the resources needed for one instance, then our overall success probability will be exponentially small in k. We establish such theorems for the classical as well as quantum query complexity of the OR function. This implies slightly weaker direct product results for all total functions. We prove a similar result for quantum communication protocols computing k instances of the Disjointness function.

Our direct product theorems imply a time-space tradeoff $T^2 S = \Omega(N^3)$ for sorting N items on a quantum computer, which is optimal up to polylog factors. They also give several tight time-space and communication-space tradeoffs for the problems of Boolean matrix-vector multiplication and matrix multiplication.

1 Introduction

1.1 Direct product theorems

For every reasonable model of computation one can ask the following fundamental question:

> How do the resources that we need for computing k independent instances of f scale with the resources needed for one instance and with k?

Here the notion of "resource" needs to be specified. It could refer to time, space, queries, communication etc. Similarly we need to define what we mean by "computing f", for instance whether we allow the algorithm some probability of error, and whether this probability of error is average or worst-case.

In this paper we consider two kinds of resources, queries and communication, and allow our algorithms some error probability. An algorithm is given k inputs $x^1, \ldots, x^k$, and has to output the vector of k answers $f(x^1), \ldots, f(x^k)$. The issue is how the algorithm can optimally distribute its resources among the k instances it needs to compute. We focus on the relation between the total amount T of resources available and the best-achievable success probability σ (which could be average or worst-case). Intuitively, if every algorithm with t resources must have some constant error probability when computing *one* instance of f, then for computing k instances we expect a constant error on each instance and hence an exponentially small success probability for the k-vector as a whole. Such a statement is known as a *weak* direct product theorem:

If $T \approx t$, then $\sigma = 2^{-\Omega(k)}$

However, even if we give our algorithm roughly kt resources, on average it still has only t resources available per instance. So even here we expect a constant error per instance and an exponentially small success probability overall. Such a statement is known as a *strong* direct product theorem:

If $T \approx kt$, then $\sigma = 2^{-\Omega(k)}$

Strong direct product theorems, though intuitively very plausible, are generally hard to prove and sometimes not even true. Shaltiel [Sha01] exhibits a general class of examples where strong direct product theorems fail. This applies for instance to query complexity, communication complexity, and circuit complexity. In his examples, success probability is taken under the uniform probability distribution on inputs. The function is chosen such that for most inputs, most of the k instances can be computed quickly and without any error probability. This leaves enough resources to solve the few hard instances with high success probability. Hence for his functions, with $T \approx tk$, one can achieve average success probability close to 1.

Accordingly, we can only establish direct product theorems in special cases. Examples are Nisan et al. 's [NRS94] strong direct product theorem for "decision forests", Parnafes et al.'s [PRW97] direct product theorem for "forests" of communication protocols, Shaltiel's strong direct product theorem for "fair" decision trees and his discrepancy bound for communication complexity [Sha01]. In the quantum case, Aaronson [Aar04, Theorem 10] established a result for the unordered search problem that lies in between the weak and the strong theorems: every T-query quantum algorithm for searching k marked items among $N = kn$ input bits will have success probability $\sigma \leq O\left(T^2/N\right)^k$. In particular, if $T \ll \sqrt{kn}$, then $\sigma = 2^{-\Omega(k)}$.

Our main contributions in this paper are strong direct product theorems for the OR-function in various settings. First consider the case of classical randomized algorithms. Let OR_n denote the n-bit OR-function, and let $f^{(k)}$ denote k independent instances of a function f. Any randomized algorithm with less than, say, $n/2$ queries will have a constant error probability when computing OR_n. Hence we expect an exponentially small success probability when computing $\mathrm{OR}_n^{(k)}$ using $\ll kn$ queries. We prove this in Section 3:

SDPT for classical query complexity:
Every randomized algorithm that computes $\mathrm{OR}_n^{(k)}$ using $T \leq \alpha kn$ queries has worst-case success probability $\sigma = 2^{-\Omega(k)}$ (for $\alpha > 0$ a sufficiently small constant).

For simplicity we have stated this result with σ being *worst-case* success probability, but the statement is also valid for the *average* success probability under a hard k-fold product distribution that is implicit in our proof.

This statement for OR actually implies a somewhat weaker DPT for all *total* functions f, via the notion of *block sensitivity $bs(f)$*. Using techniques of Nisan and Szegedy [NS94], we can embed $\mathrm{OR}_{bs(f)}$ in f (with the promise that the weight of the OR's input is 0 or 1), while on the other hand we know that the classical bounded-error query complexity $R_2(f)$ is upper bounded by $bs(f)^3$ [BBC+01]. This implies:

Every randomized algorithm that computes $f^{(k)}$ using $T \leq \alpha k R_2(f)^{1/3}$ queries has worst-case success probability $\sigma = 2^{-\Omega(k)}$.

This theorem falls short of a true strong direct product theorem in having $R_2^{1/3}(f)$ instead of $R_2(f)$ in the resource bound. However, the two most important aspects of a SDPT remain valid: the linear dependence of the resources on k and the exponential decay of the success probability.

Next we turn our attention to *quantum* algorithms. Buhrman et al. [BNRW03] actually proved that roughly k times the resources for one instance suffices to compute $f^{(k)}$ with success probability *close to 1*, rather than exponentially small: $Q_2(f^{(k)}) = \mathrm{O}(kQ_2(f))$, where $Q_2(f)$ denotes the quantum bounded-error query complexity of f (such a result is not known to hold in the classical world). For instance, $Q_2(\mathrm{OR}_n) = \Theta(\sqrt{n})$ by Grover's search algorithm, so $\mathrm{O}(k\sqrt{n})$ quantum queries suffice to compute $\mathrm{OR}_n^{(k)}$ with high success probability. The constant in the $\mathrm{O}(\cdot)$ is rather large, though. In Section 4 we show that if we make the number of queries slightly smaller, the best-achievable success probability suddenly becomes exponentially small:

> **SDPT for quantum query complexity:**
> Every quantum algorithm that computes $\mathrm{OR}_n^{(k)}$ using $T \le \alpha k \sqrt{n}$ queries has worst-case success probability $\sigma = 2^{-\Omega(k)}$ (for $\alpha > 0$ a sufficiently small constant).

Our proof uses the polynomial method [BBC+01] and is completely different from the classical proof. The polynomial method was also used by Aaronson [Aar04] in his proof of a weaker quantum direct product theorem for the search problem, mentioned above. Our proof takes its starting point from his proof, analyzing the degree of a single-variate polynomial that is 0 on $\{0, \ldots, k-1\}$, at least σ on k, and between 0 and 1 on $\{0, \ldots, kn\}$. The difference between his proof and ours is that we partially factor this polynomial, which gives us some nice extra properties over Aaronson's approach of differentiating the polynomial, and we use a strong result of Coppersmith and Rivlin [CR92]. In both cases (different) extremal properties of Chebyshev polynomials finish the proofs.

Again, using block sensitivity we can obtain a weaker result for all total functions:

> Every quantum algorithm that computes $f^{(k)}$ using $T \le \alpha k Q_2(f)^{1/6}$ queries has worst-case success probability $\sigma = 2^{-\Omega(k)}$.

The third and last setting where we establish a strong direct product theorem is quantum communication complexity. Suppose Alice has an n-bit input x and Bob has an n-bit input y. These x and y represent sets, and $\mathrm{DISJ}_n(x, y) = 1$ iff those sets are disjoint. Note that DISJ_n is the negation of $\mathrm{OR}_n(x \wedge y)$, where $x \wedge y$ is the n-bit string obtained by bitwise AND-ing x and y. In many ways, DISJ_n has the same central role in communication complexity as OR_n has in query complexity. In particular, it is "co-NP complete" [BFS86]. The communication complexity of DISJ_n has been well studied: it takes $\Theta(n)$ bits of communication in the classical world [KS92, Raz92] and $\Theta(\sqrt{n})$ in the quantum world [BCW98, HW02, AA03, Raz03]. For the case where Alice and Bob want to compute k instances of Disjointness, we establish a strong direct product theorem in Section 5:

> **SDPT for quantum communication complexity:**
> Every quantum protocol that computes $\mathrm{DISJ}_n^{(k)}$ using $T \le \alpha k \sqrt{n}$ qubits of communication has worst-case success probability $\sigma = 2^{-\Omega(k)}$.

Our proof uses Razborov's [Raz03] lower bound technique to translate the quantum protocol to a polynomial, at which point the polynomial results established for the quantum query SDPT take over. We can obtain similar results for other symmetric predicates.

One may also consider algorithms that compute the *parity* of the k outcomes instead of the vector of k outcomes. This issue has been well studied, particularly in circuit complexity, and generally goes under the name of *XOR lemmas* [Yao82, GNW95]. In this paper we focus mostly on the vector version, but we can prove similar strong bounds for the parity version. In particular, we state a classical strong XOR lemma in Section 3.3 and can get similar strong XOR lemmas for the quantum case using the technique of Cleve et al. [CDNT98, Section 3]. They show how the

ability to compute the parity of any subset of k bits with probability $1/2 + \varepsilon$, suffices to compute the full k-vector with probability $4\varepsilon^2$. Hence our strong quantum direct product theorems imply strong quantum XOR lemmas.

1.2 Time-space and communication-space tradeoffs

Apart from answering a fundamental question about the computational models of (quantum) query complexity and communication complexity, our direct product theorems also imply a number of new and optimal time-space tradeoffs.

First, we consider the tradeoff between the time T and space S that a branching program (or sufficiently similar other model) needs for *sorting* N numbers. Classically, it is well known that $TS = \Omega(N^2)$ and that this tradeoff is achievable [Bea91]. In the quantum case, Klauck [Kla03] constructed a bounded-error quantum algorithm algorithm that runs in time $T = O((N \log N)^{3/2} / \sqrt{S})$ for all $(\log N)^3 \leq S \leq N / \log N$. He also showed[1] a lower bound $TS = \Omega(N^{3/2})$, which is close to optimal for small S but not for large S. We use our strong direct product theorem to establish the tradeoff $T^2 S = \Omega(N^3)$. This is tight up to polylogarithmic factors.

Secondly, we consider time-space and communication-space tradeoffs for the problems of *Boolean matrix-vector product* and *Boolean matrix product*. In the first problem there are an $N \times N$ matrix A and a vector b of dimension N, and the goal is to compute the vector $c = Ab$, where $c_i = \vee_{j=1}^{N} (A[i,j] \wedge b_j)$. In the setting of time-space tradeoffs, the matrix A is fixed and the input is the vector b. In the problem of matrix multiplication two matrices have to be multiplied with the same type of Boolean product, and both are inputs.

Time-space tradeoffs for Boolean matrix-vector multiplication have been analyzed in an average case scenario by Abrahamson [Abr90], whose results give a worst case lower bound of $TS = \Omega(N^{3/2})$ for classical algorithms. He conjectured that a worst case lower bound of $TS = \Omega(N^2)$ holds. Using our classical direct product result we are able to confirm this, i.e., there is a matrix A, such that computing Ab requires $TS = \Omega(N^2)$. We also show a lower bound of $T^2 S = \Omega(N^3)$ for this problem in the quantum case. Both bounds are tight (the second within a logarithmic factor) if T is taken to be the number of queries to the inputs. We also get a lower bound of $T^2 S = \Omega(N^5)$ for the problem of multiplying two matrices in the quantum case. This bound is close to optimal for small S; it is open whether it is close to optimal for large S.

Research on communication-space tradeoffs in the classical setting has been initiated by Lam et al. [LTT92] in a restricted setting, and by Beame et al. [BTY94] in a general model of space-bounded communication complexity. In the setting of communication-space tradeoffs, players Alice and Bob are modeled as space bounded circuits, and we are interested in the communication cost when given particular space bounds. For the problem of computing the matrix-vector product Alice receives the matrix A (now an input) and Bob the vector b. Beame et al. gave tight lower bounds e.g. for the matrix-vector product and matrix product over GF(2), but stated the complexity of Boolean matrix-vector multiplication as an open problem. Using our direct product result for quantum communication complexity we are able to show that any quantum protocol for this problem satisfies $C^2 S = \Omega(N^3)$. This is tight within a polylogarithmic factor. We also get a lower bound of $C^2 S = \Omega(N^5)$ for computing the product of two matrices, which again is tight.

Note that no classical lower bounds for these problems were known previously, and that finding better classical lower bounds than these remains open. The possibility to show good quantum bounds comes from the deep relation between quantum protocols and polynomials implicit in Razborov's lower bound technique [Raz03].

---

[1]Unfortunately there is an error in the proof presented in [Kla03], namely Lemma 5 appears to be wrong.

2 Preliminaries

2.1 Quantum query algorithms

We assume familiarity with quantum computing [NC00] and sketch the model of quantum query complexity, referring to [BW02] for more details, also on the close relation between query complexity and degrees of multivariate polynomials. Suppose we want to compute some function f. For input $x \in \{0,1\}^N$, a *query* gives us access to the input bits. It corresponds to the unitary transformation

$$O : |i, b, z\rangle \mapsto |i, b \oplus x_i, z\rangle.$$

Here $i \in [N] = \{1, \ldots, N\}$ and $b \in \{0, 1\}$; the z-part corresponds to the workspace, which is not affected by the query. We assume the input can be accessed only via such queries. A T-query quantum algorithm has the form $A = U_T O U_{T-1} \cdots O U_1 O U_0$, where the U_k are fixed unitary transformations, independent of x. This A depends on x via the T applications of O. The algorithm starts in initial S-qubit state $|0\rangle$ and its *output* is the result of measuring a dedicated part of the final state $A|0\rangle$. For a Boolean function f, the output of A is obtained by observing the leftmost qubit of the final superposition $A|0\rangle$, and its *acceptance probability* on input x is its probability of outputting 1.

One of the most interesting quantum query algorithms is Grover's search algorithm [Gro96, BBHT98]. It can find an index of a 1-bit in an n-bit input in expected number of $O\left(\sqrt{n/(|x|+1)}\right)$ queries, where $|x|$ is the Hamming weight (number of ones) in the input. If we know that $|x| \leq 1$, we can solve the search problem exactly using $\frac{\pi}{4}\sqrt{n}$ queries [BHMT02].

2.2 Communicating quantum circuits

In the model of quantum communication complexity, two players Alice and Bob compute a function f on distributed inputs x and y. The complexity measure of interest in this setting is the amount of communication. The players follow some predefined protocol that consists of local unitary operations, and the exchange of qubits. The communication cost of a protocol is the maximal number of qubits exchanged for any input. In the standard model of communication complexity Alice and Bob are computationally unbounded entities, but we are also interested in what happens if they have bounded memory, i.e., they work with a bounded number of qubits. To this end we model Alice and Bob as communicating quantum circuits, following Yao [Yao93].

A pair of communicating quantum circuits is actually a single quantum circuit partitioned into two parts. The allowed operations are local unitary operations and access to the inputs that are given by oracles. Alice's part of the circuit may use oracle gates to read single bits from her input, and Bob's part of the circuit may do so for his input. The communication C between the two parties is simply the number of wires carrying qubits that cross between the two parts of the circuit. A pair of communicating quantum circuits uses space S, if the whole circuit works on S qubits.

In the problems we consider, the number of outputs is much larger than the memory of the players. Therefore we use the following output convention. The player who computes the value of an output sends this value to the other player at a predetermined point in the protocol. In order to make the model as general as possible, we furthermore allow the players to do local measurements, and to throw qubits away as well as pick up some fresh qubits. The space requirement only demands that at any given time no more than S qubits are in use in the whole circuit.

A final comment regarding upper bounds: Buhrman et al. [BCW98] showed how to run a query algorithm in a distributed fashion with small overhead in the communication. In particular, if there

is a T-query quantum algorithm computing N-bit function f, then there is a pair of communicating quantum circuits with $O(T \log N)$ communication that computes $f(x \wedge y)$ with the same success probability. We refer to the book of Kushilevitz and Nisan [KN97] for more on communication complexity in general, and to the surveys [Kla00, Buh00, Wol02] for more on its quantum variety.

3   Strong Direct Product Theorem for Classical Queries

In this section we prove a strong direct product theorem for randomized algorithms computing k independent instances of OR_n. By Yao's principle, it is sufficient to prove it for deterministic algorithms under a fixed hard input distribution. We consider the product distribution ν^k, where $\nu(0^n) = 1/2$ and $\nu(e_i) = 1/2n$ for e_i an n-bit string that contains a 1 only at the i-th position. It is simple to prove that the best algorithm with αn queries works in the following way: it queries αn bits, outputs 1 if it has found a 1, and 0 otherwise. Its success probability is $\frac{\alpha+1}{2}$.

3.1   Non-adaptive algorithms

An algorithm is *non-adaptive* if it decides before the first query which bits it will query. Let $Suc_{t,\mu}(f)$ be the success probability of the best deterministic algorithm for f under μ that queries at most t input bits. We first establish a strong direct product theorem for non-adaptive algorithms.

Lemma 1 *Let $f : \{0,1\}^n \to \{0,1\}$ and μ be an input distribution. Every non-adaptive deterministic algorithm for $f^{(k)}$ under μ^k with $T \leq kt$ queries has success probability $\sigma \leq Suc_{t,\mu}(f)^k$.*

Proof. The proof has two steps. First, we prove by induction that non-adaptive algorithms for $f^{(k)}$ under general product distribution $\mu_1 \times \ldots \times \mu_k$ that spend t_i queries in x^i have success probability $\leq \prod_{i=1}^{k} Suc_{t_i, \mu_i}(f)$. Second, we argue that, when $\mu_i = \mu$, the value is maximal for $t_i = t$.

Following [Sha01, Lemma 7], we prove the first part by induction on $T = t_1 + \ldots + t_k$. If $T = 0$, then the algorithm has to guess k independent random variables $x^i \sim \mu_i$. The probability of success is equal to the product of the individual success probabilities, i.e. $\prod_{i=1}^{k} Suc_{0, \mu_i}(f)$.

For the induction step $T \Rightarrow T + 1$: pick some $t_i \neq 0$ and consider two input distributions $\mu'_{i,0}$ and $\mu'_{i,1}$ obtained from μ_i by fixing the queried bit x^i_j. By the induction hypothesis, for each value $b \in \{0,1\}$, there is an optimal non-adaptive algorithm A_b that achieves the success probability $Suc_{t_i-1, \mu'_{i,b}}(f) \cdot \prod_{j \neq i} Suc_{t_j, \mu_j}(f)$. We construct a new algorithm A that calls A_b as a subroutine after it has queried x^i_j with b as an outcome. A is optimal and it has success probability

$$\left(\sum_{b=0}^{1} Pr_{\mu_i}[x^i_j = b] \cdot Suc_{t_i-1, \mu'_{i,b}}(f) \right) \cdot \prod_{j \neq i} Suc_{t_j, \mu_j}(f) = \prod_{i=1}^{k} Suc_{t_i, \mu_i}(f).$$

For symmetry reasons, if all k instances x^i are independent and identically distributed, then the optimal distribution of queries $t_1 + \ldots + t_k = kt$ is uniform, i.e. $t_i = t$. In such a case, the algorithm achieves the success probability $Suc_{t,\mu}(f)^k$. □

3.2   Adaptive algorithms

In this section we prove a similar statement also for adaptive algorithms. However, the strong direct product theorem is not true for every function and input distribution.

Example. Following [Sha01], we define $h : \{0,1\} \times \{0,1\}^n \to \{0,1\}$ as follows: $h(0,x) = 0$ and $h(1, x_1 \ldots x_n) = x_1 \oplus \ldots \oplus x_n$. Let μ be a uniform input distribution. It is simple to prove that $\mathrm{Suc}_{\frac{2}{3}n,\mu}(h) = \frac{3}{4}$ and $\mathrm{Suc}_{\frac{2}{3}nk,\mu^k}(h^{(k)}) = 1 - 2^{-\Omega(k)}$. This is because approximately half of the blocks can be solved with certainty using just 1 query and the unused queries can be used to answer exactly also the other half of the blocks. By a Chernoff bound, the probability that we do not have enough unused queries is exponentially small.

However, the strong direct product theorem is valid for $\mathrm{OR}_n^{(k)}$ under ν^k. Here there are k input blocks, each being an n-bit string distributed according to ν. Let us first state some assumptions (without loss of generality) about the best adaptive algorithm for $\mathrm{OR}_n^{(k)}$ under ν^k:

1. The algorithm is deterministic. By Yao's principle [Yao77], if there exists a randomized algorithm with success probability σ under some input distribution, then there exists a deterministic algorithm with success probability σ under that distribution.

2. Whenever the algorithm finds a 1 in some input block, it stops querying that block.

3. The algorithm spends the same number of queries in all blocks where it does not find a 1. This is optimal due to the symmetry between the blocks, and implies that the algorithm will spend at least as many queries in each "empty" input block as in each "non-empty" block.

Now, by upper bounding the number of queries in each block, we prove that no adaptive algorithm can be much better than a non-adaptive algorithm with slightly more queries.

Lemma 2 *If there is an adaptive T-query algorithm A computing $\mathrm{OR}_n^{(k)}$ under ν^k with probability σ, then there is a non-adaptive $3T$-query algorithm A' computing it with probability $\sigma - 2^{-\Omega(k)}$.*

Proof. The expected number of empty blocks (i.e. 0^n-blocks) is $\frac{1}{2}k$. By a Chernoff bound, the probability that the number Z of empty blocks is smaller than $\frac{1}{3}k$ is $\delta = 2^{-\Omega(k)}$. If there are at least $\frac{1}{3}k$ empty blocks, then the algorithm spends at most $\frac{T}{k/3}$ queries in each of them. Define the following non-adaptive algorithm A': it spends $3T/k$ queries in each block regardless of whether it finds a 1 there or not. It is simple to prove that $\Pr[A' \text{ succeeds} \mid Z \geq k/3] \geq \Pr[A \text{ succeeds} \mid Z \geq k/3]$. Let us compare the overall success probabilities of A and A':

$$
\begin{aligned}
\sigma_{A'} &= \Pr[Z < k/3] \cdot \Pr[A' \text{ succeeds} \mid Z < k/3] + \Pr[Z \geq k/3] \cdot \Pr[A' \text{ succeeds} \mid Z \geq k/3] \\
&\geq \Pr[Z \geq k/3] \cdot \Pr[A \text{ succeeds} \mid Z \geq k/3], \\
\sigma_A &= \Pr[Z < k/3] \cdot \Pr[A \text{ succeeds} \mid Z < k/3] + \Pr[Z \geq k/3] \cdot \Pr[A \text{ succeeds} \mid Z \geq k/3] \\
&\leq \delta \cdot 1 + \Pr[Z \geq k/3] \cdot \Pr[A \text{ succeeds} \mid Z \geq k/3] \leq \delta + \sigma_{A'}.
\end{aligned}
$$

We conclude that $\sigma_{A'} \geq \sigma_A - \delta$. (*Remark.* By replacing the $k/3$-bound on Z by a βk-bound for some $0 < \beta < \frac{1}{2}$, we can obtain an arbitrary constant $0 < \gamma < 1$ in the exponent $\delta = 2^{-\gamma k}$, while the number of queries of A' becomes T/β.) $\quad\square$

Finally, we combine the two results. Lemma 1 says that non-adaptive algorithm have exponentially small acceptance probability and Lemma 2 says that adaptive algorithms are not much better. Recall that $\mathrm{Suc}_{\alpha n, \nu}(\mathrm{OR}_n) = \frac{\alpha+1}{2}$. We get that non-adaptive algorithms for $\mathrm{OR}_n^{(k)}$ under ν^k with $\alpha k n$ queries have success probability $\leq \mathrm{Suc}_{\alpha n, \nu}(\mathrm{OR}_n)^k = (\frac{\alpha+1}{2})^k = 2^{-\log(\frac{2}{\alpha+1})k}$. Note that we can achieve any constant $\gamma < 1$ in the exponent by choosing α sufficiently small. In sum:

Theorem 3 (SDPT for OR) *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that every randomized algorithm for $\mathrm{OR}_n^{(k)}$ with $T \leq \alpha k n$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

3.3 A bound for the parity instead of the vector of results

Here we give a strong direct product theorem for the *parity* of k independent instances of OR_n. The parity is a Boolean variable, hence we can always guess it with probability at least $\frac{1}{2}$. However, we prove that the advantage (instead of the success probability) of our guess must be exponentially small.

Let X be a random bit with $\Pr[X = 1] = p$. We define the *advantage of X* by $\mathrm{Adv}(X) = |2p - 1|$. Note that a uniformly distributed random bit has advantage 0 and a bit known with certainty has advantage 1. It is well known that if $X_1, \ldots, X_k$ are independent random bits, then $\mathrm{Adv}(X_1 \oplus \ldots \oplus X_k) = \prod_{i=1}^{k} \mathrm{Adv}(X_i)$. Compare this with the fact that the probability of guessing correctly the complete vector $(X_1, \ldots, X_k)$ is the product of the individual probabilities.

We have proved a lower bound for the computation of $OR_n^{(k)}$ (vector of OR's). By the same technique, replacing the success probability by the advantage in all claims and proofs, we can also prove a lower bound for the computation of $OR_n^{\oplus k}$ (parity of OR's).

Theorem 4 (SDPT for parity of OR's) *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that every randomized algorithm for $OR_n^{\oplus k}$ with $T \leq \alpha k n$ queries has advantage $\tau \leq 2^{-\gamma k}$.*

3.4 A bound for all functions

Here we show that the strong direct product theorem for OR actually implies a weaker direct product theorem for all functions. In this weaker version, the success probability of computing k instances still goes down exponentially with k, but we need to start from a polynomially smaller bound on the overall number of queries.

Definition 1 *For $x \in \{0, 1\}^n$ and $S \subseteq [n]$, we use x^S to denote the n-bit string obtained from x by flipping the bits in S. Consider a (possibly partial) function $f : \mathcal{D} \to Z$, with $\mathcal{D} \subseteq \{0, 1\}^n$. The block sensitivity $bs_x(f)$ of $x \in \mathcal{D}$ is the maximal b for which there are disjoint sets $S_1, \ldots, S_b$ such that $f(x) \neq f(x^{S_i})$. The block sensitivity of f is $\max_{x \in \mathcal{D}} bs_x(f)$.*

Block sensitivity is closely related to deterministic and bounded-error classical query complexity:

Theorem 5 ([Nis91, BBC+01]) $R_2(f) = \Omega(bs(f))$ *for all f, $D(f) \leq bs(f)^3$ for all total Boolean f.*

Nisan and Szegedy [NS94] showed how to embed a $bs(f)$-bit OR-function (with the promise that the input has weight ≤ 1) into f. Combined with our strong direct product theorem for OR, this implies a direct product theorem for all functions in terms of their block sensitivity:

Theorem 6 *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that for every f, every classical algorithm for $f^{(k)}$ with $T \leq \alpha k\, bs(f)$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

This is optimal whenever $R_2(f) = \Theta(bs(f))$, which is the case for most functions. For total functions, the gap between $R_2(f)$ and $bs(f)$ is not more than cubic, hence

Corollary 7 *For every $0 < \gamma < 1$, there exists an $\alpha > 0$ such that for every total Boolean f, every classical algorithm for $f^{(k)}$ with $T \leq \alpha k R_2(f)^{1/3}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

4 Strong Direct Product Theorem for Quantum Queries

In this section we prove a strong direct product theorem for quantum algorithms computing k independent instances of OR. Our proof relies on the polynomial method of [BBC+01].

4.1 Bounds on polynomials

We use three results about polynomials, also used in [BCWZ99]. The first is by Coppersmith and Rivlin [CR92, p. 980] and gives a general bound for polynomials bounded by 1 at integer points:

Theorem 8 (Coppersmith & Rivlin [CR92]) *Every polynomial p of degree $d \leq n$ that has absolute value*

$$|p(i)| \leq 1 \text{ for all integers } i \in [0, n],$$

satisfies

$$|p(x)| < ae^{bd^2/n} \text{ for all real } x \in [0, n],$$

where $a, b > 0$ are universal constants (no explicit values for a and b are given in [CR92]).

The other two results concern the Chebyshev polynomials T_d, defined as in [Riv90]:

$$T_d(x) = \frac{1}{2}\left(\left(x + \sqrt{x^2 - 1}\right)^d + \left(x - \sqrt{x^2 - 1}\right)^d\right).$$

T_d has degree d and its absolute value $|T_d(x)|$ is bounded by 1 if $x \in [-1, 1]$. On the interval $[1, \infty)$, T_d exceeds all others polynomials with those two properties ([Riv90, p.108] and [Pat92, Fact 2]):

Theorem 9 *If q is a polynomial of degree d such that $|q(x)| \leq 1$ for all $x \in [-1, 1]$ then $|q(x)| \leq |T_d(x)|$ for all $x \geq 1$.*

Paturi [Pat92, before Fact 2] proved

Lemma 10 (Paturi [Pat92]) $T_d(1 + \mu) \leq e^{2d\sqrt{2\mu + \mu^2}}$ *for all $\mu \geq 0$.*

Proof. For $x = 1 + \mu$: $T_d(x) \leq (x + \sqrt{x^2 - 1})^d = (1 + \mu + \sqrt{2\mu + \mu^2})^d \leq (1 + 2\sqrt{2\mu + \mu^2})^d \leq e^{2d\sqrt{2\mu + \mu^2}}$ (using that $1 + z \leq e^z$ for all real z). $\qquad\square$

The following key lemma is the basis for all our direct product theorems:

Lemma 11 *Suppose p is a degree-D polynomial such that for some $\delta \geq 0$*

$$-\delta \leq p(i) \leq \delta \text{ for all } i \in \{0, \ldots, k-1\},$$
$$p(k) = \sigma,$$
$$p(i) \in [-\delta, 1 + \delta] \text{ for all } i \in \{0, \ldots, N\}.$$

Then for every integer $1 \leq C < N - k$ and $\mu = 2C/(N - k - C)$ we have

$$\sigma \leq a\left(1 + \delta + \frac{\delta(2N)^k}{(k-1)!}\right) \cdot \exp\left(\frac{b(D-k)^2}{(N-k-C)} + 2(D-k)\sqrt{2\mu + \mu^2} - k\ln(C/k)\right) + \delta k 2^{k-1},$$

where a, b are the constants given by Theorem 8.

Before establishing this gruesome bound, let us reassure the reader by noting that we will apply this lemma with δ negligibly small, $D = \alpha\sqrt{kN}$ for sufficiently small α, and $C = ke^{\gamma+1}$, giving

$$\sigma \leq \exp\left((b\alpha^2 + 4\alpha e^{\gamma/2+1/2} - 1 - \gamma)k\right) \leq e^{-\gamma k} \leq 2^{-\gamma k}.$$

Proof of Lemma 11. Divide p with remainder by $\prod_{j=0}^{k-1}(x-j)$ to obtain

$$p(x) = q(x)\prod_{j=0}^{k-1}(x-j) + r(x),$$

where $d = \deg(q) = D - k$ and $\deg(r) \le k - 1$. We know that $r(x) = p(x) \in [-\delta, \delta]$ for all $x \in \{0, \dots, k-1\}$. Decompose r as a linear combination of polynomials e_i, where $e_i(i) = 1$ and $e_i(x) = 0$ for $x \in \{0, \dots, k-1\} - \{i\}$:

$$r(x) = \sum_{i=0}^{k-1} p(i)e_i(x) = \sum_{i=0}^{k-1} p(i) \prod_{\substack{j=0 \\ j\neq i}}^{k-1} \frac{x-j}{i-j}.$$

We bound the values of r for all real $x \in [0, N]$ by

$$|r(x)| \ \le \ \sum_{i=0}^{k-1} \frac{|p(i)|}{i!(k-1-i)!} \prod_{\substack{j=0 \\ j\neq i}}^{k-1} |x-j| \le \frac{\delta}{(k-1)!} \sum_{i=0}^{k-1} \binom{k-1}{i} N^k \le \frac{\delta(2N)^k}{(k-1)!},$$

$$|r(k)| \ \le \ \delta k 2^{k-1}.$$

This implies the following about the values of the polynomial q:

$$|q(k)| \ge (\sigma - \delta k 2^{k-1})/k!$$

$$|q(i)| \le \frac{(i-k)!}{i!}\left(1 + \delta + \frac{\delta(2N)^k}{(k-1)!}\right) \text{ for } i \in \{k, \dots, N\}$$

In particular:

$$|q(i)| \le C^{-k}\left(1 + \delta + \frac{\delta(2N)^k}{(k-1)!}\right) = A \text{ for } i \in \{k+C, \dots, N\}$$

Theorem 8 implies that there are constants $a, b > 0$ such that:

$$|q(x)| \le A \cdot a e^{bd^2/(N-k-C)} = B \text{ for all real } x \in [k+C, N]$$

We now divide q by B to normalize it, and rescale the interval $[k+C, N]$ to $[1, -1]$ to get a degree-d polynomial t satisfying

$$|t(x)| \le 1 \text{ for all } x \in [-1, 1]$$
$$t(1+\mu) = q(k)/B \text{ for } \mu = 2C/(N-k-C)$$

Since t cannot grow faster than the degree-d Chebyshev polynomial, we get

$$t(1+\mu) \le T_d(1+\mu) \le e^{2d\sqrt{2\mu+\mu^2}}.$$

Combining our upper and lower bounds on $t(1+\mu)$, we obtain

$$\frac{(\sigma - \delta k 2^{k-1})/k!}{C^{-k}\left(1 + \delta + \frac{\delta(2N)^k}{(k-1)!}\right) a e^{bd^2/(N-k-C)}} \le e^{2d\sqrt{2\mu+\mu^2}}.$$

Rearranging gives the bound. $\qquad\square$

4.2 Consequences for quantum algorithms

The previous result about polynomials implies a strong tradeoff between queries and success probability for quantum algorithms that have to find k ones in an N-bit input. A *k-threshold algorithm with success probability* σ is an algorithm on N-bit input x, that outputs 0 with certainty if $|x| < k$, and outputs 1 with probability at least σ if $|x| = k$.

Theorem 12 *For every $\gamma > 0$, there exists an $\alpha > 0$ such that every quantum k-threshold algorithm with $T \leq \alpha\sqrt{kN}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

Proof. Fix $\gamma > 0$ and consider a T-query k-threshold algorithm. By [BBC+01], its acceptance probability is an N-variate polynomial of degree $D \leq 2T \leq 2\alpha\sqrt{kN}$ and can be symmetrized to a single-variate polynomial p with the properties

$p(i) = 0$ if $i \in \{0, \ldots, k-1\}$
$p(k) \geq \sigma$
$p(i) \in [0,1]$ for all $i \in \{0, \ldots, N\}$

Choosing $\alpha > 0$ sufficiently small and $\delta = 0$, the result follows from Lemma 11. □

This implies a strong direct product theorem for k instances of the n-bit search problem:

Theorem 13 (SQDPT for Search) *For every $\gamma > 0$, there exists an $\alpha > 0$ such that every quantum algorithm for* $\mathrm{Search}_n^{(k)}$ *with $T \leq \alpha k\sqrt{n}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

Proof. Set $N = kn$, fix a $\gamma > 0$ and a T-query algorithm A for $\mathrm{Search}_n^{(k)}$ with success probability σ. Now consider the following algorithm that acts on an N-bit input x:

1. Apply a random permutation π to x.

2. Run A on $\pi(x)$.

3. Query each of the k positions that A outputs, and return 1 iff at least $k/2$ of those bits are 1.

This uses $T + k$ queries. We will show that it is a $k/2$-threshold algorithm. First, if $|x| < k/2$ it always outputs 0. Second, consider the case $|x| = k/2$. The probability that π puts all $k/2$ ones in distinct n-bit blocks is

$$\frac{N}{N} \cdot \frac{N-n}{N-1} \cdots \frac{N - \frac{k}{2}n}{N - \frac{k}{2}} \geq \left(\frac{N - \frac{k}{2}n}{N}\right)^{k/2} = 2^{-k/2}.$$

Hence our algorithm outputs 1 with probability at least $\sigma 2^{-k/2}$. Choosing α sufficiently small, the previous theorem implies $\sigma 2^{-k/2} \leq 2^{-(\gamma+1/2)k}$, hence $\sigma \leq 2^{-\gamma k}$. □

Our bounds are quite precise for $\alpha \ll 1$. We can choose $\gamma = 2\ln(1/\alpha) - O(1)$ and ignore some lower-order terms to get roughly $\sigma \leq \alpha^{2k}$. On the other hand, it is known that Grover's search algorithm with $\alpha\sqrt{n}$ queries on an n-bit input has success probability roughly α^2 [BBHT98]. Doing such a search on all k instances gives overall success probability α^{2k}.

Theorem 14 (SQDPT for OR) *There exist $\alpha, \gamma > 0$ such that every quantum algorithm for* $\mathrm{OR}_n^{(k)}$ *with $T \leq \alpha k\sqrt{n}$ queries has success probability $\sigma \leq 2^{-\gamma k}$.*

Proof. An algorithm A for $\mathrm{OR}_n^{(k)}$ with success probability σ can be used to build an algorithm A' for $\mathrm{Search}_n^{(k)}$ with slightly worse success probability:

1. Run A on the original input and remember which blocks contain a 1.

2. Run simultaneously (at most k) binary searches on the nonzero blocks. Iterate this $s = 2\log(1/\alpha)$ times. Each iteration is computed by running A on the parts of the blocks that are known to contain a 1, halving the remaining instance size each time.

3. Run the exact version of Grover's algorithm on each of the remaining parts of the instances to look for a one there (each remaining part has size $n/2^s$).

This new algorithm A' uses $(s+1)T + \frac{\pi}{4}k\sqrt{n/2^s} = O(\alpha\log(1/\alpha)k\sqrt{n})$ queries. With probability at least σ^{s+1}, A succeeds in all iterations, in which case A' solves $\mathrm{Search}_n^{(k)}$. By the previous theorem, for every $\gamma' > 0$ of our choice we can choose $\alpha > 0$ such that

$$\sigma^{s+1} \leq 2^{-\gamma' k},$$

which implies the theorem with $\gamma = \gamma'/(s+1)$. $\qquad\square$

Choosing our parameters carefully, we can actually show that for every $\gamma < 1$ there is an $\alpha > 0$ such that $\alpha k\sqrt{n}$ queries give success probability $\sigma \leq 2^{-\gamma k}$. Clearly, $\sigma = 2^{-k}$ is achievable without any queries by random guessing.

4.3   A bound for all functions

As in Section 3.4, we can extend the strong direct product theorem for OR to a slightly weaker theorem for all total functions. Block sensitivity is closely related to bounded-error quantum query complexity:

Theorem 15 ([BBC$^+$01]) $Q_2(f) = \Omega\left(\sqrt{bs(f)}\right)$ for all f, $D(f) \leq bs(f)^3$ for all total Boolean f.

By embedding an OR of size $bs(f)$ in f, we obtain

Theorem 16 *There exist* $\alpha, \gamma > 0$ *such that for every* f, *every quantum algorithm for* $f^{(k)}$ *with* $T \leq \alpha k\sqrt{bs(f)}$ *queries has success probability* $\sigma \leq 2^{-\gamma k}$.

This is close to optimal whenever $Q_2(f) = \Theta\left(\sqrt{bs(f)}\right)$. For total functions, the gap between $Q_2(f)$ and $\sqrt{bs(f)}$ is no more than a 6th power, hence

Corollary 17 *There exist* $\alpha, \gamma > 0$ *such that for every total Boolean* f, *every quantum algorithm for* $f^{(k)}$ *with* $T \leq \alpha k Q_2(f)^{1/6}$ *queries has success probability* $\sigma \leq 2^{-\gamma k}$.

5   Strong Direct Product Theorem for Quantum Communication

In this section we establish a strong direct product theorem for quantum communication complexity, specifically for protocols that compute k independent instances of the Disjointness problem. Our proof relies crucially on the beautiful technique that Razborov introduced to establish a lower bound on the quantum communication complexity of (one instance of) Disjointness [Raz03]. It allows us to translate a quantum communication protocol to a single-variate polynomial that represents, roughly speaking, the protocol's acceptance probability as a function of the size of the intersection of x and y. Once we have this polynomial, the results from Section 4.1 suffice to establish a strong direct product theorem.

5.1 Razborov's technique

Razborov's technique relies on the following linear algebraic notions. The *operator norm* $\| A \|$ of a matrix A is its largest singular value σ_1. The *trace inner product* between A and B is $\langle A, B \rangle = \mathrm{Tr}(A^*B)$. The *trace norm* is $\| A \|_{tr} = \max\{|\langle A, B \rangle| : \| B \| = 1\} = \sum_i \sigma_i$. The *Frobenius norm* is $\| A \|_F = \sqrt{\sum_{ij} |A_{ij}|^2} = \sqrt{\sum_i \sigma_i^2}$. The following lemma is implicit in Razborov's paper.

Lemma 18 *Consider a Q-qubit quantum communication protocol on N-bit inputs x and y, with acceptance probabilities denoted by $P(x,y)$. Define $P(i) = \mathrm{Exp}_{|x|=|y|=N/4, |x \wedge y|=i}[P(x,y)]$, where the expectation is taken uniformly over all x, y that each have weight $N/4$ and that have intersection i. For every $d \leq N/4$ there exists a degree-d polynomial q such that $|P(i) - q(i)| \leq 2^{-d/4+2Q}$ for all $i \in \{0, \ldots, N/8\}$.*

Proof. We only consider the $\mathcal{N} = \binom{N}{N/4}$ strings of weight $N/4$. Let P denote the $\mathcal{N} \times \mathcal{N}$ matrix of the acceptance probabilities on these inputs. We know from Yao and Kremer [Yao93, Kre95] that we can decompose P as a matrix product $P = AB$, where A is an $\mathcal{N} \times 2^{2Q-2}$ matrix with each entry at most 1 in absolute value, and similarly for B. Note that $\| A \|_F, \| B \|_F \leq \sqrt{\mathcal{N} 2^{2Q-2}}$. Using Hölder's inequality we have:

$$\| P \|_{tr} \leq \| A \|_F \cdot \| B \|_F \leq \mathcal{N} 2^{2Q-2}.$$

Let μ_i denote the $\mathcal{N} \times \mathcal{N}$ matrix corresponding to the uniform probability distribution on $\{(x,y) : |x \wedge y| = i\}$. These "combinatorial matrices" have been well studied [Knu03]. Note that $\langle P, \mu_i \rangle$ is the expected acceptance probability $P(i)$ of the protocol under that distribution. One can show that the different μ_i commute, so they have the same eigenspaces $E_0, \ldots, E_{N/4}$ and can be simultaneously diagonalized by some orthogonal matrix U. For $t \in \{0, \ldots, N/4\}$, let $(UPU^T)_t$ denote the block of UPU^T corresponding to E_t, and $a_t = \mathrm{Tr}((UPU^T)_t)$ be its trace. Then we have

$$\sum_{t=0}^{N/4} |a_t| \leq \sum_{j=1}^{\mathcal{N}} \left| (UPU^T)_{jj} \right| \leq \| UPU^T \|_{tr} = \| P \|_{tr} \leq \mathcal{N} 2^{2Q-2},$$

where the second inequality is a property of the trace norm.

Let λ_{it} be the eigenvalue of μ_i in eigenspace E_t. It is known [Raz03, Section 5.3] that λ_{it} is a degree-t polynomial in i, and that $|\lambda_{it}| \leq 2^{-t/4}/\mathcal{N}$ for $i \leq N/8$ (the factor $1/4$ in the exponent is implicit in Razborov's paper). Consider the high-degree polynomial p defined by

$$p(i) = \sum_{t=0}^{N/4} a_t \lambda_{it}.$$

This satisfies

$$p(i) = \sum_{t=0}^{N/4} \mathrm{Tr}((UPU^T)_t) \lambda_{it} = \langle UPU^T, U\mu_i U^T \rangle = \langle P, \mu_i \rangle = P(i).$$

Let q be the degree-d polynomial obtained by removing the high-degree parts of p:

$$q(i) = \sum_{t=0}^{d} a_t \lambda_{it}.$$

13

Then P and q are close on all integers i between 0 and $N/8$:

$$|P(i) - q(i)| = |p(i) - q(i)| = \left| \sum_{t=d+1}^{N/4} a_t \lambda_{it} \right| \leq \frac{2^{-d/4}}{\mathcal{N}} \sum_{t=0}^{N/4} |a_t| \leq 2^{-d/4+2Q}.$$

$\square$

5.2 Consequences for quantum protocols

Combining Razborov's technique with our polynomial bounds we can prove

Theorem 19 (SQDPT for Disjointness) *There exist $\alpha, \gamma > 0$ such that every quantum protocol for $\mathrm{DISJ}_n^{(k)}$ with $Q \leq \alpha k \sqrt{n}$ qubits of communication has success probability $p \leq 2^{-\gamma k}$.*

Proof (sketch). By doing the same trick with $s = 2\log(1/\alpha)$ rounds of binary search as for Theorem 14, we can tweak a protocol for $\mathrm{DISJ}_n^{(k)}$ to a protocol that satisfies, with $P(i)$ defined as in Lemma 18, $N = kn$ and $\sigma = p^{s+1}$:

$P(i) = 0$ if $i \in \{0, \ldots, k-1\}$
$P(k) \geq \sigma$
$P(i) \in [0,1]$ for all $i \in \{0, \ldots, N\}$

(a subtlety: instead of exact Grover we use an exact version of the $\mathrm{O}(\sqrt{n})$-qubit Disjointness protocol of [AA03]; the [BCW98]-protocol would lose a $\log n$-factor). Lemma 18, using $d = 12Q$, then gives a degree-d polynomial q that differs from P by at most $\delta \leq 2^{-Q}$ on all $i \in \{0, \ldots, N/8\}$. This δ is sufficiently small to apply Lemma 11, which in turn upper bounds σ and hence p. $\square$

This technique also gives strong direct product theorems for symmetric predicates other than DISJ_n.

6 Time-Space Tradeoff for Quantum Sorting

We will now use our strong direct product theorem to get near-optimal time-space tradeoffs for quantum circuits for sorting. This follows Klauck [Kla03], who described an upper bound $T^2 S = \mathrm{O}\big((N \log N)^3\big)$ and a lower bound $TS = \Omega\big(N^{3/2}\big)$. In our model, the numbers $a_1, \ldots, a_N$ that we want to sort can be accessed by means of queries, and the number of queries lower bounds the actual time taken by the circuit. This includes the model of branching programs. The circuit has N output gates and in the course of its computation outputs the N numbers in sorted order, say from large to small. Its success probability is at least $2/3$.

Theorem 20 *Every bounded-error quantum circuit for sorting N numbers that uses T queries and S qubits of workspace satisfies $T^2 S = \Omega\big(N^3\big)$.*

Proof. We "slice" the circuit along the time-axis into $L = T/\alpha\sqrt{SN}$ slices, each containing $T/L = \alpha\sqrt{SN}$ queries. Each such slice has a number of output gates. Consider any slice. Suppose it contains output gates $i_1 < \ldots < i_k \leq N/2$, so it is supposed to output the i_1-th up to i_k-th smallest elements of its input. We want to show that $k = \mathrm{O}(S)$. If $k \leq S$ then we are done, so assume $k > S$. We can use the slice as a k-threshold algorithm on $N/2$ bits, as follows. For an $N/2$-bit input x, let the sorting input consist of $i_1 - 1$ copies of the number 2, the $N/2$ bits in x,

and $N/2 - i_1 + 1$ copies of the number 0. Consider the behavior of the sorting circuit on this input. The first part of the circuit has to output the $i_1 - 1$ largest numbers, which are all 2. We condition on the event that the circuit succeeds in this. It then passes on an S-qubit state (possibly mixed) as the starting state of the particular slice we are considering. This slice then outputs the k largest numbers in x with probability at least 2/3. Now, consider an algorithm that runs just this slice, starting with the completely mixed state on S-qubits, and that outputs 1 if it finds k ones, and outputs 0 otherwise. If $|x| < k$ this new algorithm always outputs 0, but if $|x| = k$ then it outputs 1 with probability at least

$$\sigma \geq \frac{2}{3} \cdot 2^{-S},$$

because the completely mixed state has "overlap" 2^{-S} with the "good" S-qubit state that would have been the starting state of the slice in the run of the sorting circuit. On the other hand, the slice has only $\alpha\sqrt{SN} < \alpha\sqrt{kN}$ queries, so by choosing α sufficiently small, Theorem 12 implies

$$\sigma \leq 2^{-\Omega(k)},$$

hence $k = O(S)$. Thus we need $L = \Omega(N/S)$ slices, so $T = L\alpha\sqrt{SN} = \Omega\left(N^{3/2}/\sqrt{S}\right)$. $\qquad\square$

Note that our lower bound already holds for sorting N numbers from $\{0, 1, 2\}$. As mentioned, this tradeoff is achievable up to polylog factors [Kla03]. Interestingly, the near-optimal algorithm uses only a polylogarithmic number of qubits and otherwise just classical memory.

7 Time-Space Tradeoffs for Boolean Matrix Products

First we show a lower bound on the time-space tradeoff for Boolean matrix-vector multiplication on *classical* machines.

Theorem 21 *There is a matrix A such that every classical bounded-error circuit that computes the Boolean matrix-vector product Ab with T queries and space $S = o(N/\log N)$ satisfies $TS = \Omega\left(N^2\right)$.*

Note that we could equally well formulate the above theorem in terms of branching programs, where time corresponds to depth, and space to the logarithm of the size of such programs. The bound is tight if T measures queries to the input or if we consider the branching program model.

Proof. Fix $k = O(S)$ large enough. First we have to find a hard matrix A. We pick A randomly by setting $N/(2k)$ random positions in each row to 1. We want to show that with positive probability for all sets of k rows $A[i_1], \ldots, A[i_k]$ many of the rows $A[i_j]$ contain at least $N/(6k)$ ones that are not ones in any of the $k - 1$ other rows.

This probability can be bounded as follows. We will treat the rows as subsets of $\{1, \ldots, N\}$. A row $A[j]$ is called *bad* with respect to $k-1$ other rows $A[i_1], \ldots, A[i_{k-1}]$, if $|A[j] - \cup_\ell A[i_\ell]| \leq N/(6k)$. For fixed $i_1, \ldots, i_{k-1}$, the probability that some $A[j]$ is bad with respect to the $k - 1$ other rows is at most $e^{-\Omega(N/k)}$ by the Chernoff bound and the fact that k rows can together contain at most $N/2$ elements. Since $k = o(N/\log N)$ we may assume this probability is at most $1/N^{10}$.

Now fix any set $I = \{i_1, \ldots, i_k\}$. The probability that for $j \in I$ it holds that $A[j]$ is bad with respect to the other rows is at most $1/N^{10}$, and this also holds, if we condition on the event that some other rows are bad, since this condition makes it only less probable that another row is also

bad. So for any fixed $J \subset I$ of size $k/2$ the probability that all rows in J are bad is at most N^{-5k}, and the probability that there exists such J is at most

$$\binom{k}{k/2} N^{-5k}.$$

Furthermore the probability that there is a set I of k rows for which $k/2$ are bad is at most

$$\binom{N}{k} \binom{k}{k/2} N^{-5k} < 1.$$

So there is an A as required and we may fix one.

Now suppose we are given a circuit with space S that computes the Boolean product between the rows of A and b in some order. We again proceed by "slicing" the circuit into $L = T/\alpha N$ slices, each containing $T/L = \alpha N$ queries. Each such slice has a number of output gates. Consider any slice. Suppose it contains output gates $i_1 < \ldots < i_k \le N/2$, so it is supposed to output $\vee_{\ell=1}^{N} (A[i_j, \ell] \wedge b_\ell)$ for all i_j with $1 \le j \le k$.

Such a slice starts on a classical value of the "memory" of the circuit, which is in general a probability distribution on S bits (if the circuit is randomized). We replace this probability distribution by the uniform distribution on the possible values of S bits. If the original circuit succeeds in computing the function correctly with probability at least $1/2$, then so does the circuit slice with its outputs, and replacing the initial value of the memory by a uniformly random one decreases the success probability to no less than $(1/2) \cdot 1/2^S$.

If we now show that any classical circuit with αN queries that produces the outputs $i_1, \ldots, i_k$ can succeed only with exponentially small probability in k, we get that $k = \mathrm{O}(S)$, and hence $(T/\alpha N) \cdot \mathrm{O}(S) \ge N$, which gives the claimed lower bound for the time-space tradeoff.

Each set of k outputs corresponds to k rows of A, which contain $N/(2k)$ ones each. Thanks to the construction of A there are $k/2$ rows among these, such that $N/(6k)$ of the ones in each such row are in position where none of the other contains a one. So we get $k/2$ sets of $N/(6k)$ positions that are unique to each of the $k/2$ rows. The inputs for b will be restricted to contain ones only at these positions, and so the algorithm naturally has to solve $k/2$ independent OR problems on $n = N/(6k)$ bits each. By Theorem 3, this is only possible with αN queries if the success probability is exponentially small in k. $\qquad\square$

An absolutely analogous construction can be done in the quantum case. Using circuit slices of length $\alpha\sqrt{NS}$ we can prove:

Theorem 22 *There is a matrix A such that every quantum bounded-error circuit that computes the Boolean matrix-vector product Ab with T queries and space $S = \mathrm{o}(N/\log N)$ satisfies $T^2 S = \Omega(N^3)$.*

Note that this is tight within a logarithmic factor (needed to improve the success probability of Grover search).

Theorem 23 *Every classical bounded-error circuit that computes the Boolean matrix product AB with T queries and space S satisfies $TS = \Omega(N^3)$.*

While this is near-optimal for small S, it is probably not tight for large S, a likely tight trade-off being $T^2 S = \Omega(N^6)$. It is also no improvement compared to Abrahamson's average case bounds [Abr90].

Proof. Suppose that $S = o(N)$, otherwise the bound is trivial, since time N^2 is always needed. We can proceed similar to the proof of Theorem 21. We slice the circuit so that each slice has only αN queries. Suppose a slice makes k outputs. We are going to restrict the inputs to get a direct product problem with k instances of size N/k each, hence a slice with αN queries has exponentially small success probability in k and can thus produce only O(S) outputs. Since the overall number of outputs is N^2 we get the tradeoff $TS = \Omega\left(N^3\right)$.

Suppose a circuit slice makes k outputs, where an output labeled (i,j) needs to produce the vector product of the ith row $A[i]$ of A and the jth column $B[j]$ of B. We may partition the set $\{1,\ldots,N\}$ into k mutually disjoint subsets $U(i,j)$ of size N/k, each associated to an output (i,j).

Assume that there are ℓ outputs $(i,j_1),\ldots,(i,j_\ell)$ involving $A[i]$. Each such output is associated to a subset $U(i,j_t)$, and we set $A[i]$ to zero on all positions that are not in any of these subsets, and to one on all positions that are in one of these. When there are ℓ outputs $(i_1,j),\ldots,(i_\ell,j)$ involving $B[j]$, we set $B[j]$ to zero on all positions that are not in any of the corresponding subsets, and allow the inputs to be arbitrary on the other positions.

If the circuit computes on these restricted inputs, it actually has to compute k instances of OR of size $n = N/k$ in B, for it is true that $A[i]$ and $B[j]$ contain a single block of size N/k in which $A[i]$ contains only ones, and $B[j]$ "free" input bits, if and only if (i,j) is one of the k outputs. Hence the strong direct product theorem is applicable. □

The application to the quantum case is analogous.

Theorem 24 *Every quantum bounded-error circuit that computes the Boolean matrix product AB with T queries and space S satisfies $T^2S = \Omega\left(N^5\right)$.*

If $S = $ O($\log N$), then N^2 applications of Grover can compute AB with $T = $ O$\left(N^{2.5}\log N\right)$. Hence our tradeoff is near-optimal for small S. We do not know whether it is optimal for large S.

8   Quantum Communication-Space Tradeoffs for Matrix Products

In this section use the strong direct product result for quantum communication Theorem 19 to prove communication-space tradeoffs. We later show that these are close to optimal.

Theorem 25 *Every quantum bounded-error protocol in which Alice and Bob have bounded space S and that computes the Boolean matrix-vector product, satisfies $C^2S = \Omega\left(N^3\right)$.*

Proof. In a protocol, Alice receives a matrix A, and Bob a vector b as inputs. Given a circuit that multiplies these with communication C and space S, we again proceed to slice it. This time, however, a slice contains a limited amount of communication. Recall that in communicating quantum circuits the communication corresponds to wires carrying qubits that cross between Alice's and Bob's circuits. Hence we may cut the circuit after $\alpha\sqrt{NS}$ qubits have been communicated and so on. Overall there are $C/\alpha\sqrt{NS}$ circuit slices. Each starts with an initial state that may be replaced by the completely mixed state at the cost of decreasing the success probability to $(1/2)\cdot 1/2^S$. We want to employ the direct product theorem for quantum communication complexity to show that a protocol with the given communication has success probability at most exponentially small in the number of outputs it produces, and so a slice can produce at most O(S) outputs. Combining these bounds with the fact that N outputs have to be produced gives the tradeoff.

To use the direct product theorem we restrict the inputs in the following way: Suppose a protocol makes k outputs. We partition the vector b into k blocks of size N/k, and each block

is assigned to one of the k rows of A for which an output is made. This row is made to contain zeroes outside of the positions belonging to its block, and hence we arrive at a problem where Disjointness has to be computed on k instances of size N/k. With communication $\alpha\sqrt{kN}$, the success probability must be exponentially small in k due to Theorem 19. Hence $k = \mathrm{O}(S)$ is an upper bound on the number of outputs produced. $\qquad\square$

Theorem 26 *Every quantum bounded-error protocol in which Alice and Bob have bounded space S and that computes the Boolean matrix product, satisfies $C^2 S = \Omega\big(N^5\big)$.*

Proof. The proof uses the same slicing approach as in the other tradeoff results. Note that we can assume that $S = \mathrm{o}(N)$, since otherwise the bound is trivial. Each slice contains communication $\alpha\sqrt{NS}$, and as before a direct product result showing that k outputs can be computed only with success probability exponentially small in k leads to the conclusion that a slice can only compute $\mathrm{O}(S)$ outputs. Therefore $(C/\alpha\sqrt{NS}) \cdot \mathrm{O}(S) \geq N^2$, and we are done.

Consider a protocol with $\alpha\sqrt{NS}$ qubits of communication. We partition the universe $\{1, \ldots, N\}$ of the Disjointness problems to be computed into k mutually disjoint subsets $U(i,j)$ of size N/k, each associated to an output (i,j), which in turn corresponds to a row/column pair $A[i]$, $B[j]$ in the input matrices A and B. Assume that there are ℓ outputs $(i,j_1), \ldots, (i,j_\ell)$ involving $A[i]$. Each output is associated to a subset of the universe $U(i,j_t)$, and we set $A[i]$ to zero on all positions that are not in one of these subsets. Then we proceed analogously with the columns of B.

If the protocol computes on these restricted inputs, it has to solve k instances of Disjointness of size $n = N/k$ each, since $A[i]$ and $B[j]$ contain a single block of size N/k in which both are not set to 0 if and only if (i,j) is one of the k outputs. Hence Theorem 19 is applicable. $\qquad\square$

We now want to show that these tradeoffs are not too far from optimal.

Theorem 27 *There is a quantum bounded-error protocol with space S that computes the Boolean product between a matrix and a vector within communication $C = \mathrm{O}((N^{3/2}\log^2 N)/\sqrt{S})$.*

There is a quantum bounded-error protocol with space S that computes the Boolean product between two matrices within communication $C = \mathrm{O}((N^{5/2}\log^2 N)/\sqrt{S})$.

Proof. We begin by showing a protocol for the following scenario: Alice gets S N-bit vectors $x_1, \ldots, x_S$, Bob gets an N-vector vector y, and they want to compute the S Boolean inner products between these vectors. The protocol uses space $\mathrm{O}(S)$.

In the following, we interpret Boolean vectors as sets. The main idea is that Alice can use the union z of the x_i and then Alice and Bob can find an element in the intersection of z and y using the protocol for the Disjointness problem described in [BCW98]. Alice then marks all x_i that contain this element and removes them from z.

A problem with this approach is that Alice cannot store z explicitly, since it might contain much more than S elements. Alice may, however, store the indices of those sets x_i for which an element in the intersection of x_i and y has already been found, in an array of length S. This array and the input given as an oracle work as an implicit representation of z.

Now suppose at some point during the protocol the intersection of z and y has size k. Then Alice and Bob can find one element in this intersection within $\mathrm{O}(\sqrt{N/k})$ rounds of communication in which $\mathrm{O}(\log N)$ qubits are exchanged each. Furthermore in $\mathrm{O}(\sqrt{Nk})$ rounds all elements in the intersection can be found. So if $k \leq S$, then all elements are found within communication $\mathrm{O}(\sqrt{NS}\log N)$ and the problem can be solved completely. On the other hand, if $k \geq S$, finding

one element costs $O(\sqrt{N/S}\log N)$, but finding such an element removes at least one x_i from z, and hence this has to be done at most S times, giving the same overall communication bound.

It is not hard to see that this process can be implemented with space $O(S)$. The protocol from [BCW98] is a distributed Grover search that itself uses only space $O(\log N)$. Bob can work as in this protocol. For each search, Alice has to start with a superposition over all indices in z. This superposition can be computed from her oracle and her array. In each step she has to apply the Grover iteration. This can also be implemented from these two resources.

To get a protocol for matrix-vector product, the above procedure is repeated N/S times, hence the communication is $O((N/S) \cdot \sqrt{NS}\log^2 N)$, where one logarithmic factor stems from improving success probability to $1/\mathrm{poly}(N)$.

For the product of two matrices, the matrix-vector protocol may be repeated N times. $\qquad\square$

These near-optimal protocols use only $O(\log N)$ "real" qubits, and otherwise just classical memory.

9 Open Problems

We mention some open problems. The first is to determine tight time-space tradeoffs for Boolean matrix product on both classical and quantum computers. Second, regarding communication-space tradeoffs for Boolean matrix-vector and matrix product, we did not prove any classical bounds that were better than our quantum bounds. Klauck [Kla04] recently proved classical tradeoffs $CS^2 = \Omega(N^3)$ and $CS^2 = \Omega(N^2)$ for Boolean matrix product and matrix-vector product, respectively, by means of a *weak* direct product theorem for Disjointness. A classical *strong* direct product theorem for Disjointness would imply optimal tradeoffs, but we do not know how to prove this at the moment. Finally, it would be interesting to get any lower bounds on time-space or communication-space tradeoffs for decision problems in the quantum case, for example for Element Distinctness [BDH$^+$01, Amb03] or the verification of matrix multiplication [BŠ04].

Acknowledgments

References

[AA03]   S. Aaronson and A. Ambainis. Quantum search of spatial regions. In *Proceedings of 44th IEEE FOCS*, pages 200–209, 2003. quant-ph/0303041.

[Aar04]   S. Aaronson. Limitations of quantum advice and one-way communication. In *Proceedings of 19th IEEE Conference on Computational Complexity*, 2004. To appear. quant-ph/0402095.

[Abr90]   K. Abrahamson. A time-space tradeoff for Boolean matrix multiplication. In *Proceedings of 31st IEEE FOCS*, pages 412–419, 1990.

[Amb03]   A. Ambainis. Quantum walk algorithm for element distinctness. quant-ph/0311001, 1 Nov 2003.

[BBC$^+$01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. quant-ph/9802049.

[BBHT98] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. Earlier version in Physcomp'96. quant-ph/9605034.

[BCW98] H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proceedings of 30th ACM STOC*, pages 63–68, 1998. quant-ph/9802040.

[BCWZ99] H. Buhrman, R. Cleve, R. de Wolf, and Ch. Zalka. Bounds for small-error and zero-error quantum algorithms. In *Proceedings of 40th IEEE FOCS*, pages 358–368, 1999. cs.CC/9904019.

[BDH$^+$01] H. Buhrman, Ch. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. In *Proceedings of 16th IEEE Conference on Computational Complexity*, pages 131–137, 2001. quant-ph/0007016.

[Bea91] P. Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM Journal on Computing*, 20(2):270–277, 1991. Earlier version in STOC'89.

[BFS86] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proceedings of 27th IEEE FOCS*, pages 337–347, 1986.

[BHMT02] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74. 2002. quant-ph/0005055.

[BNRW03] H. Buhrman, I. Newman, H. Röhrig, and R. de Wolf. Robust quantum algorithms and polynomials. quant-ph/0309220, 30 Sep 2003.

[BŠ04] H. Buhrman and R. Špalek. Quantum verification of matrix products. Unpublished manuscript, 2004.

[BTY94] P. Beame, M. Tompa, and P. Yan. Communication-space tradeoffs for unrestricted protocols. *SIAM Journal on Computing*, 23(3):652–661, 1994. Earlier version in FOCS'90.

[Buh00] H. Buhrman. Quantum computing and communication complexity. *EATCS Bulletin*, 70:131–141, February 2000.

[BW02] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[CDNT98] R. Cleve, W. van Dam, M. Nielsen, and A. Tapp. Quantum entanglement and the communication complexity of the inner product function. In *Proceedings of 1st NASA QCQC conference*, volume 1509 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 1998. quant-ph/9708019.

[CR92] D. Coppersmith and T. J. Rivlin. The growth of polynomials bounded at equally spaced points. *SIAM Journal on Mathematical Analysis*, 23(4):970–983, 1992.

[GNW95]   O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR lemma. Technical report, ECCC TR–95–050, 1995. Available at http://www.eccc.uni-trier.de/eccc/.

[Gro96]   L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996. quant-ph/9605043.

[HW02]   P. Høyer and R. de Wolf. Improved quantum communication complexity bounds for disjointness and equality. In *Proceedings of 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2002)*, volume 2285 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2002. quant-ph/0109068.

[Kla00]   H. Klauck. Quantum communication complexity. In *Proceedings of Workshop on Boolean Functions and Applications at 27th ICALP*, pages 241–252, 2000. quant-ph/0005032.

[Kla03]   H. Klauck. Quantum time-space tradeoffs for sorting. In *Proceedings of 35th ACM STOC*, pages 69–76, 2003. quant-ph/0211174.

[Kla04]   H. Klauck. Quantum and classical communication-space tradeoffs from rectangle bounds. Unpublished manuscript, 2004.

[KN97]   E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[Knu03]   D. E. Knuth. Combinatorial matrices. In *Selected Papers on Discrete Mathematics*, volume 106 of *CSLI Lecture Notes*. Stanford University, 2003.

[Kre95]   I. Kremer. Quantum communication. Master's thesis, Hebrew University, Computer Science Department, 1995.

[KS92]   B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992. Earlier version in Structures'87.

[LTT92]   T.W. Lam, P. Tiwari, and M. Tompa. Trade-offs between communication and space. *Journal of Computer and Systems Sciences*, 45(3):296–315, 1992. Earlier version in STOC'89.

[NC00]   M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[Nis91]   N. Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. Earlier version in STOC'89.

[NRS94]   N. Nisan, S. Rudich, and M. Saks. Products and help bits in decision trees. In *Proceedings of 35th FOCS*, pages 318–329, 1994.

[NS94]   N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994. Earlier version in STOC'92.

[Pat92]   R. Paturi. On the degree of polynomials that approximate symmetric Boolean functions. In *Proceedings of 24th ACM STOC*, pages 468–474, 1992.

[PRW97]   I. Parnafes, R. Raz, and A. Wigderson. Direct product results and the GCD problem, in old and new communication models. In *Proceedings of 29th ACM STOC*, pages 363–372, 1997.

[Raz92]   A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

[Raz03]   A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya of the Russian Academy of Science, mathematics*, 67(1):159–176, 2003. quant-ph/0204025.

[Riv90]   T. J. Rivlin. *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*. Wiley-Interscience, second edition, 1990.

[Sha01]   R. Shaltiel. Towards proving strong direct product theorems. In *Proceedings of 16th IEEE Conference on Computational Complexity*, pages 107–119, 2001.

[Wol02]   R. de Wolf. Quantum communication and complexity. *Theoretical Computer Science*, 287(1):337–353, 2002.

[Yao77]   A. C-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of 18th IEEE FOCS*, pages 222–227, 1977.

[Yao82]   A. C-C. Yao. Theory and applications of trapdoor functions. In *Proceedings of 23rd IEEE FOCS*, pages 80–91, 1982.

[Yao93]   A. C-C. Yao. Quantum circuit complexity. In *Proceedings of 34th IEEE FOCS*, pages 352–360, 1993.