Quantum Search of Spatial Regions (Extended Abstract)*

Scott Aaronson†
University of California, Berkeley

Andris Ambainis‡
University of Latvia

Abstract

Can Grover's quantum search algorithm speed up search of a physical region—for example a 2-D grid of size $\sqrt{n} \times \sqrt{n}$? The problem is that $\sqrt{n}$ time seems to be needed for each query, just to move amplitude across the grid. Here we show that this problem can be surmounted, refuting a claim to the contrary by Benioff. In particular, we show how to search a d-dimensional hypercube in time $O(\sqrt{n})$ for $d \geq 3$, or $O(\sqrt{n}\log^3 n)$ for $d = 2$. More generally, we introduce a model of quantum query complexity on graphs, motivated by fundamental physical limits on information storage, particularly the holographic principle from black hole thermodynamics. Our results in this model include almost-tight upper and lower bounds for many search tasks; a generalized algorithm that works for any graph with good expansion properties, not just hypercubes; and relationships among several notions of 'locality' for unitary matrices acting on graphs. As an application of our results, we give an $O(\sqrt{n})$-qubit communication protocol for the disjointness problem, which improves an upper bound of Høyer and de Wolf and matches a lower bound of Razborov.

1 Introduction

The goal of Grover's quantum search algorithm [13] is to search an 'unsorted database' of size n in a number of queries proportional to $\sqrt{n}$. Classically, of course, order n queries are needed. It is sometimes asserted that, although the speedup of Grover's algorithm is only quadratic, this speedup is *provable*, in contrast to the exponential speedup of Shor's factoring algorithm [20]. But is that really true? Grover's algorithm is typically imagined as speeding up combinatorial search—and we do not know whether every problem in NP can be classically solved quadratically faster than in the "obvious" way, any more than we know whether factoring is in BPP.

But could Grover's algorithm speed up search of a *physical region*? Here the basic problem, it seems to us, is the time needed for signals to travel across the region. For if we are interested in the fundamental limits imposed by physics, then we should acknowledge that the speed of light is finite, and that a bounded region of space can store only a finite amount of information, according to the *holographic principle* [7]. We discuss the latter constraint in detail in Section 2; for now, we say only that it suggests a model in which a 'quantum robot' occupies a superposition over finitely many locations, and moving the robot from one location to an adjacent one takes unit time. In such a model, the time needed to search a region could depend critically on its spatial layout. For example, if the n entries are arranged on a line, then even to move the robot from one end to the other takes $n-1$ steps. But what if the entries are arranged on, say, a 2-dimensional square grid?

1.1 Summary of Results

This paper gives the first systematic treatment of quantum search of spatial regions, with 'regions' modeled as connected graphs. Our main result is positive: we show that a quantum robot can search a d-dimensional hypercube with n vertices in time $O(\sqrt{n})$ for $d \geq 3$, or $O(\sqrt{n}\log^3 n)$ for $d = 2$. This matches (or in the case of 2 dimensions, nearly matches) the $\Omega(\sqrt{n})$ lower bound for quantum search, and supports the view that Grover search of a physical region presents no problem of principle. Our basic technique is divide-and-conquer; indeed, once the idea is pointed out, an upper bound of $O(n^{1/2+\varepsilon})$ follows readily. However, to obtain the tighter bounds is more difficult; for that we use

*Due to space limitations many proofs are omitted from this abstract. Please see `www.arxiv.org/abs/quant-ph/0303041` for the full version.

†Email: `aaronson@cs.berkeley.edu`. Supported by an NSF Graduate Fellowship and by the Defense Advanced Research Projects Agency (DARPA) and Air Force Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-0524.

‡Email: `ambainis@ias.edu`.

the amplitude-amplification framework of Brassard et al. [9].

Section 5 presents the main results; Section 5.4 shows further that, if the number of marked items is at least k, then the search time decreases to $O\left(\sqrt{n}k^{-1/2+1/d}\right)$ for $d \geq 3$ (this upper bound is tight). Also, Section 7 generalizes our algorithm to arbitrary graphs that have 'hypercube-like' expansion properties. Here the best bounds we can achieve are $O\left(\sqrt{n}\operatorname{polylog} n\right)$ for $d > 2$, or $\sqrt{n}2^{O\left(\sqrt{\log n}\right)}$ for $d = 2$. Building on these results, Section 7.1 shows that if a d-dimensional hypercube (or graph with similar expansion properties) has h 'possible' marked items, at arbitrary known locations, then the search time is $O\left(\sqrt{h}\left(n/h\right)^{1/d}\operatorname{polylog} h\right)$ for $d > 2$. Intuitively, this says that there is no worse scenario than having the h possible marked items spaced equally throughout the hypercube.

Section 6 shows, as an unexpected application of our search algorithm, that the quantum communication complexity of the well-known *disjointness problem* is $O\left(\sqrt{n}\right)$. This improves an $O\left(\sqrt{n}c^{\log^* n}\right)$ upper bound of Høyer and de Wolf [14], and matches the $\Omega\left(\sqrt{n}\right)$ lower bound of Razborov [17].

The rest of the paper is about the formal model that underlies our results. Section 2 sets the stage for this model, by exploring the ultimate limits on information storage imposed by properties of space and time. This discussion serves only to motivate our results; thus, it can be safely skipped by readers unconcerned with the physical Universe. In Section 3 we define *quantum query algorithms on graphs*, a model similar to quantum query algorithms as defined by Beals et al. [2], but with the added requirement that unitary operations be 'local' with respect to some graph. In Section 3.1 we address the difficult question, which also arises in work on quantum random walks [1] and quantum cellular automata [21], of what 'local' means. In Section 4 we give general facts about our model, including an upper bound of $O\left(\sqrt{n\delta}\right)$ for the time needed to search any graph with diameter δ, and a matching lower bound using the hybrid argument of Bennett et al. [5].

1.2 Relation to Previous Work

In a paper on 'Space searches with a quantum robot,' Benioff [4] asked whether Grover's algorithm can speed up search of a physical region, as opposed to a combinatorial search space. His answer was discouraging: for a 2-D grid of size $\sqrt{n} \times \sqrt{n}$, Grover's algorithm is no faster than classical search. The reason is that, during each of the $\Theta\left(\sqrt{n}\right)$ Grover iterations, the algorithm must use order $\sqrt{n}$ steps just to travel across the grid and return to its starting point for the diffusion step. On the other hand, Benioff noted, Grover's algorithm does yield some speedup for grids of dimension 3 or higher, since those grids have diameter less than $\sqrt{n}$.

Our results show that Benioff's claim is mistaken: by using Grover's algorithm more carefully, one can search a 2-D grid in $O\left(\sqrt{n}\operatorname{polylog} n\right)$ steps. To us this illustrates why one should not assume an algorithm is optimal on heuristic grounds. Painful experience—for example, the "obviously optimal" $O\left(n^3\right)$ matrix multiplication algorithm—is what taught computer scientists to see the proving of lower bounds as more than a formality.

Our setting is related to that of quantum random walks on graphs [1, 11]. Most work on quantum walks is concerned with preparing a (near-) uniform distribution over vertices, a problem quite different from that of finding a particular marked vertex. However, Shenvi, Kempe, and Whaley [19] showed how to use a quantum walk to find a marked vertex on the Boolean hypercube, with efficiency matching that of Grover's algorithm. In light of that result, and of intriguing numerical evidence supplied by N. Shenvi, we asked in an earlier version of this paper whether quantum walks can yield a speedup for searching fixed-dimensional grid graphs. Recently Childs and Goldstone [11] have studied our question in the continuous-time setting. Using sophisticated eigenvalue analysis, they show that a quantum walk can search a d-dimensional hypercube for a single marked item in $O\left(\sqrt{n}\right)$ steps when $d > 4$, or $O\left(\sqrt{n}\log n\right)$ steps when $d = 4$ (assuming one can use amplitude amplification over multiple runs of the walk). When $d = 3$ the running time increases to $O\left(n^{5/6}\right)$, again allowing amplitude amplification (A. Childs, personal communication), whereas when $d = 2$ no speedup is found. It is still possible that a choice of Hamiltonian different from Childs and Goldstone's would yield better results for $d < 4$.

Currently, the main drawbacks of the quantum walk approach are that (1) the running time is not competitive with ours in low dimensions, and (2) all analyses of quantum walks to date have relied heavily on symmetries of the underlying graph. If even minor 'defects' are introduced, the analysis becomes much harder. By contrast, as we show in Section 7, our algorithm relies only on global properties of the graph being searched.

2 The Physics of Databases

Theoretical computer science generally deals with the *limit* as some resource (such as time or memory) increases to infinity. What is not always appreciated is that, as the resource bound increases, physical constraints may come into play that were negligible at 'sub-asymptotic' scales. We believe theoretical computer scientists ought to know something about such constraints, and to account for them when possible. For if the constraints are ignored on the ground that they "never matter in practice," then the obvious question arises: why use asymptotic analysis in the first place, rather than restricting attention to those instance sizes that occur in practice?

A constraint of particular interest for us is the *holographic principle* [7], which arose from black-hole thermodynamics. The principle states that the information content of any spatial region is upper-bounded by its *surface area* (not volume), at a rate of one bit per Planck area, or about 1.4×10^{69} bits per square meter. Intuitively, if one tried to build a spherical hard disk with mass density υ, one could not keep expanding it forever. For as soon as the radius reached the Schwarzschild bound of $r = \sqrt{3/(8\pi\upsilon)}$ (in Planck units, $c = G = \hbar = k = 1$), the hard disk would collapse to form a black hole, and thus its contents would be irretrievable.

Actually the situation is worse than that: even a *planar* hard disk of constant mass density would collapse to form a black hole once its radius became sufficiently large, $r = \Theta(1/\upsilon)$. (We assume here that the hard disk is disc-shaped. A linear or 1-D hard disk could expand indefinitely without collapse.) It is possible, though, that a hard disk's information content could asymptotically exceed its mass. For example, a black hole's mass is proportional to the radius of its event horizon, but the entropy is proportional to the *square* of the radius (that is, to the surface area). Admittedly, inherent difficulties with storage and retrieval make a black hole horizon less than ideal as a hard disk. However, even a weakly-gravitating system could store information at a rate asymptotically exceeding its mass-energy. For instance, an enclosed ball of radiation with radius r can store $n = \Theta\left(r^{3/2}\right)$ bits [7, 12], even though its energy grows only as r. Our results in Section 7.1 will imply that a quantum robot could (in principle!) search such a 'radiation disk' for a marked item in time $O\left(r^{5/4}\right) = O\left(n^{5/6}\right)$. This is some improvement over the trivial $O(n)$ upper bound for a 1-D hard disk, though it falls short of the desired $O(\sqrt{n})$.

In general, if $n = r^c$ bits are scattered throughout a 3-D ball of radius r (where $c \leq 3$ and the bits' locations are known), we show that the time needed to search for a '1' bit grows as $n^{1/c+1/6} = r^{1+c/6}$ (omitting logarithmic factors). In particular, if $n = \Theta\left(r^2\right)$ (saturating the holographic bound), then the time grows as $n^{2/3}$ or $r^{4/3}$. To achieve a search time of $O(\sqrt{n}\operatorname{polylog} n)$, the bits would need to be concentrated on a 2-D surface.

Because of the holographic principle, we see that it is not only quantum mechanics that yields a $\Omega(\sqrt{n})$ lower bound on the number of steps needed for unordered search. If the items to be searched are laid out spatially, then general relativity in $3+1$ dimensions independently yields the same bound, $\Omega(\sqrt{n})$, up to a constant factor.[1] Interestingly, in $d+1$ dimensions the relativity bound would be $\Omega\left(n^{1/(d-1)}\right)$, which for $d > 3$ is weaker than the quantum mechanics bound. Given that our two fundamental theories yield the same lower bound, it is natural to ask whether that bound is tight. The answer seems to be that it is *not* tight, since (i) the entropy on a black hole horizon is not efficiently accessible,[2] and (ii) weakly-gravitating systems are subject to the *Bekenstein bound* [3], an even stronger entropy constraint than the holographic bound.[3]

Yet it is still of basic interest to know whether n bits in a radius-r ball can be searched in time $o(\min\{n, r\sqrt{n}\})$—that is, whether it is possible to do *anything* better than either brute-force quantum search (with the drawback pointed out by Benioff [4]), or classical search. Our results show that it is possible.

Physicists with whom we discussed these ideas have raised three questions: (1) whether our complexity measure is realistic; (2) how to account for time dilation; and (3) whether given the number of bits we are imagining, cosmological bounds are also relevant. Let us address these questions in turn.

(1) One could argue that to maintain a 'quantum database' of size n requires n computing elements ([22], though see also [18]). So why not just exploit those elements to search the database in *parallel*? Then it becomes trivial to show that the search time is limited only by the radius of the database, so the algorithms of this paper are unnecessary. Our response is that, while there might be n 'passive' computing elements (capable of storing data), there might be many fewer 'active' elements, which we consequently wish to place in a superposition over locations. This assumption is unobjectionable from a physical point of view. For a particle (and indeed any object) really does have an indeterminate location, not merely an indeterminate

[1] Admittedly, the holographic principle is part of quantum gravity and not general relativity *per se*. All that matters for us, though, is that the principle seems logically independent of quantum-mechanical linearity, which is what produces the "other" $\Omega(\sqrt{n})$ bound.

[2] In the case of a black hole horizon, waiting for the bits to be emitted as Hawking radiation (if indeed they are) takes time proportional to r^3, which is much too long.

[3] Assuming a finite number of particle species in Nature, the entropy upper bound is $O\left(r^{3/2}\right)$ (which is saturated by a 'radiation disk') [12].

internal state (such as spin) *at* some location. We leave as an open problem, however, whether our assumption is valid for specific quantum computer architectures such as ion traps.

(2) So long as we invoke general relativity, shouldn't we also consider the effects of time dilation? Those effects are indeed pronounced near a black hole horizon. Again, though, for our upper bounds we will have in mind systems far from the Schwarzschild limit, for which any time dilation is by at most a constant factor independent of n.

(3) How do cosmological considerations affect our analysis? Bousso [6] argues that, in a spacetime with positive cosmological constant $\Lambda > 0$, the total number of bits accessible to any one experiment is at most $3\pi/(\Lambda \ln 2)$, or roughly 10^{122} given current experimental bounds [16] on Λ.[4] Intuitively, even if the Universe is spatially infinite, most of it recedes too quickly from any one observer to be harnessed as computer memory.

One response to this result is to assume an idealization in which Λ vanishes, although Planck's constant $\hbar$ does not vanish. As justification, one could argue that without the idealization $\Lambda = 0$, *all* asymptotic bounds in computer science are basically fictions. But perhaps a better response is to accept the $3\pi/(\Lambda \ln 2)$ bound, and then ask how close one can come to *saturating* it in different scenarios. Classically, the maximum number of bits that can be searched is, in a crude model[5], actually proportional to $1/\sqrt{\Lambda} \approx 10^{61}$ rather than $1/\Lambda$. The reason is that if a region had much more than $1/\sqrt{\Lambda}$ bits, then after $1/\sqrt{\Lambda}$ Planck times—that is, about 10^{10} years, or roughly the current age of the Universe—most of the region would have receded beyond one's cosmological horizon. What our results suggest is that, using a quantum robot, one could come closer to saturating the cosmological bound—since, for example, a 2-D region of size $1/\Lambda$ can be searched in time $O\left(\frac{1}{\sqrt{\Lambda}} \operatorname{polylog} \frac{1}{\sqrt{\Lambda}}\right)$. How anyone could *prepare* (say) a database of size much greater than $1/\sqrt{\Lambda}$ remains unclear, but if such a database existed, it could be searched!

[4]Also, Lloyd [15] argues that the total number of bits accessible *up till now* is at most the square of the number of Planck times elapsed so far, or about $(10^{61})^2 = 10^{122}$. Lloyd's bound, unlike Bousso's, does not depend on Λ being positive. The numerical coincidence between the two bounds reflects the experimental finding [16] that we live in a transitional era, when both Λ and "dust" contribute significantly to the Universe's net energy balance ($\Omega_\Lambda \approx 0.7$, $\Omega_{\text{dust}} \approx 0.3$). In earlier times dust (and before that radiation) dominated, and Lloyd's bound was tighter. In later times Λ will dominate, and Bousso's bound will be tighter. *Why* we should live in such a transitional era is unknown.

[5]Specifically, neglecting gravity and other forces that could counteract the effect of Λ.

3 The Model

Much of what is known about the power of quantum computing comes from the *black-box* or *query* model [2, 5, 13, 20], in which one counts only the number of queries to an oracle, not the number of computational steps. We will take this model as the starting point for a formal definition of quantum robots. Doing so will focus attention on our main concern: how much harder is it to evaluate a function when its inputs are spatially separated? As it turns out, all of our algorithms *will* be efficient as measured by the number of gates and auxiliary qubits needed to implement them.

For simplicity, we assume that a robot's goal is to evaluate a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, which could be partial or total. A 'region of space' is a connected undirected graph $G = (V, E)$ with vertices $V = \{v_1, \ldots, v_n\}$. Let $X = x_1 \ldots x_n \in \{0,1\}^n$ be an input to f; then each bit x_i is available only at vertex v_i. We assume the robot knows G and the vertex labels in advance, and so is ignorant only of the x_i bits. We thus sidestep a major difficulty for quantum walks [1], which is how to ensure that a process on an unknown graph is unitary.

At any time, the robot's state has the form $\sum \alpha_{i,z} |v_i, z\rangle$. Here $v_i \in V$ is a vertex, representing the robot's location; and z is a bit string (which can be arbitrarily long), representing the robot's internal configuration. The state evolves via an alternating sequence of T algorithm steps and T oracle steps:

$$U^{(1)} \rightarrow O^{(1)} \rightarrow \cdots \rightarrow U^{(T)} \rightarrow O^{(T)}.$$

An oracle step $O^{(t)}$ maps each basis state $|v_i, z\rangle$ to $|v_i, z \oplus x_i\rangle$, where x_i is exclusive-OR'ed into the first bit of z. An algorithm step $U^{(t)}$ can be any unitary matrix that (1) does not depend on X, and (2) acts 'locally' on G. How to make the second condition precise is the subject of Section 3.1.

The initial state of the algorithm is $|v_1, 0\rangle$. Let $\alpha_{i,z}^{(t)}(X)$ be the amplitude of $|v_i, z\rangle$ immediately after the t^{th} oracle step; then the algorithm succeeds with probability $1 - \varepsilon$ if

$$\sum_{|v_i,z\rangle \,:\, z_{OUT} = f(X)} \left|\alpha_{i,z}^{(T)}(X)\right|^2 \geq 1 - \varepsilon$$

for all inputs X, where z_{OUT} is a bit of z representing the output.

3.1 Locality Criteria

Classically, it is easy to decide whether a stochastic matrix acts *locally* with respect to a graph G: it does if it moves probability only along the edges of G. In the quantum case, however, interference makes the question much more subtle. In this section we propose three alternative criteria for whether a unitary matrix U is local. Our algorithms can be implemented using the most restrictive of these criteria, whereas our lower bounds apply to all three of them.

The first criterion we call *Z-locality* (for zero): U is Z-local if, given any pair of non-neighboring vertices v_1, v_2 in G, U "sends no amplitude" from v_1 to v_2; that is, the corresponding entries in U are all 0. The second criterion, *C-locality* (for composability), says that this is not enough: not only must U send amplitude only between neighboring vertices, but it must be composed of a product of commuting unitaries, each of which acts on a single edge. The third criterion is perhaps the most natural one to a physicist: U is *H-local* (for Hamiltonian) if it can be obtained by applying a locally-acting, low-energy Hamiltonian for some fixed amount of time. More formally, let $U_{i,z\rightarrow i^*,z^*}$ be the entry in the $|v_i, z\rangle$ column and $|v_{i^*}, z^*\rangle$ row of U.

Definition 1 *U is Z-local if $U_{i,z\rightarrow i^*,z^*} = 0$ whenever $i \neq i^*$ and (v_i, v_{i^*}) is not an edge of G.*

Definition 2 *U is C-local if the basis states can be partitioned into subsets $P_1, \ldots, P_q$ such that*

(i) $U_{i,z\rightarrow i^*,z^*} = 0$ whenever $|v_i, z\rangle$ and $|v_{i^*}, z^*\rangle$ belong to distinct P_j's, and

(ii) for each j, all basis states in P_j are either from the same vertex or from two adjacent vertices.

Definition 3 *U is H-local if $U = e^{iH}$ for some Hermitian H with eigenvalues of absolute value at most π, such that $H_{i,z\rightarrow i^*,z^*} = 0$ whenever $i \neq i^*$ and (v_i, v_{i^*}) is not an edge in E.*

If a unitary matrix is C-local, then it is also Z-local and H-local. Beyond that, though, how are the locality criteria related? Are they approximately equivalent? If not, then does a problem's complexity in our model ever depend on which criterion is chosen? A key question, which we leave open, is whether any Z-local or H-local unitary can be approximated by a product of, say, $O(\log n)$ C-local unitaries. (A product of $O(n)$ such unitaries trivially suffices, but that is far too many.) In the full version of the paper, we prove many weaker relations among the locality criteria (as well as some weak separations), and also show that the criteria are robust under various changes to the definitions.

4 General Bounds

Given a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, the quantum query complexity $Q(f)$, defined by Beals et al. [2], is the minimum T for which there exists a T-query quantum algorithm that evaluates f with probability at least 2/3 on all inputs. (We will always be interested in the *two-sided, bounded-error* complexity, sometimes denoted $Q_2(f)$.) Similarly, given a graph G with n vertices labeled $1, \ldots, n$, we define $Q(f, G)$ to be the minimum T for which there exists a T-query quantum robot on G that evaluates f with probability 2/3. Here the algorithm steps must be C-local.

Let δ_G be the diameter of G, and call f *nondegenerate* if it depends on all n input bits.

Proposition 4 *For all f, G,*

(i) $Q(f, G) \leq 2n - 3$.

(ii) $Q(f, G) \leq (2\delta_G + 1) Q(f)$.

(iii) $Q(f, G) \geq Q(f)$.

(iv) $Q(f, G) \geq \delta_G/2$ if f is nondegenerate.

Our model is robust in the following senses (the proof is omitted from this abstract).

Proposition 5 *For nondegenerate f, the following change $Q(f, G)$ by at most a constant factor.*

(i) Replacing the initial state $|v_1, 0\rangle$ by an arbitrary (known) $|\psi\rangle$.

(ii) Requiring the final state to be localized at some vertex v_i with probability at least $1 - \varepsilon$.

(iii) Allowing multiple algorithm steps between each oracle step (and measuring the complexity by the number of algorithm steps).

A function of particular interest is $f = \mathrm{OR}(x_1, \ldots, x_n)$, which outputs 1 if and only if $x_i = 1$ for some i. We first give a general upper bound on $Q(\mathrm{OR}, G)$ in terms of the diameter of G. (Throughout the paper, we omit floor and ceiling signs when they have no effect on the asymptotics.)

Proposition 6 $Q(\mathrm{OR}, G) = O\left(\sqrt{n\delta_G}\right)$.

Proof. Let τ be a minimum-height spanning tree for G, rooted at v_1. A depth-first search on τ uses $2n - 2$ steps. Let S_1 be the set of vertices visited by DFS in steps 1 to δ_G, S_2 be those visited in steps $\delta_G + 1$ to $2\delta_G$, and so on. Then $S_1 \cup \cdots \cup S_{2n/\delta_G} = V$. Furthermore, for each S_j there is a classical algorithm

A_j, using at most $3\delta_G$ steps, that starts at v_1, ends at v_1, and outputs '1' if and only if $x_i = 1$ for some $v_i \in S_j$. Then we simply perform Grover search at v_1 over all A_j; since each iteration takes $O(\delta_G)$ steps and there are $O\left(\sqrt{2n/\delta_G}\right)$ iterations, the number of steps is $O\left(\sqrt{n\delta_G}\right)$. ■

The bound of Proposition 6 is tight:

Theorem 7 *For all δ, there exists a G for which $\delta_G = \delta$ and $Q(\mathrm{OR}, G) = \Omega\left(\sqrt{n\delta}\right)$.*

Proof Sketch. We let G be a 'starfish' with a central vertex v and $2(n-1)/\delta$ legs radiating outward from v, each of length $\delta/2$. A marked item is placed at the tip vertex of one of the legs. Intuitively, this turns the problem into a standard Grover search of a list of size $2(n-1)/\delta$, but where each query takes δ steps—$\delta/2$ to move the robot to a tip vertex, and $\delta/2$ to move it back to v. Thus the total number of steps needed is $\Omega\left(\delta\sqrt{n/\delta}\right) = \Omega\left(\sqrt{n\delta}\right)$. This intuition can be formalized by a relatively standard hybrid argument along the lines of Bennett et al. [5], which we omit from this abstract. ■

5 Search on Grids

Let $\mathcal{L}_d(n)$ be a d-dimensional grid graph of size $n^{1/d} \times \cdots \times n^{1/d}$. That is, each vertex is specified by d coordinates $i_1, \ldots, i_d \in \{1, \ldots, n^{1/d}\}$, and is connected to the at most $2d$ vertices obtainable by adding or subtracting 1 from a single coordinate (boundary vertices have fewer than $2d$ neighbors). We write simply $\mathcal{L}_d$ when n is clear from context. In this section we present our main positive results: that $Q(\mathrm{OR}, \mathcal{L}_d) = \Theta(\sqrt{n})$ for $d \geq 3$, and $Q(\mathrm{OR}, \mathcal{L}_2) = O(\sqrt{n}\,\mathrm{polylog}\, n)$ for $d = 2$.

Before proving these claims, let us develop some intuition by showing weaker bounds, taking the case $d = 2$ for illustration. Clearly $Q(\mathrm{OR}, \mathcal{L}_2) = O\left(n^{3/4}\right)$: we simply divide $\mathcal{L}_2(n)$ into $\sqrt{n}$ subsquares $S(1), \ldots, S(\sqrt{n})$, each isomorphic to $\mathcal{L}_2(\sqrt{n})$. In $5\sqrt{n}$ steps, the robot can travel from the start vertex to any $S(i)$, search $S(i)$ classically for a marked item, and then return to the start vertex. Thus, by searching all $\sqrt{n}$ of the $S(i)$'s in superposition and applying Grover's algorithm, the robot can search the grid in time $O\left(n^{1/4}\right) \times 5\sqrt{n} = O\left(n^{3/4}\right)$.

Once we know that, we might as well divide $\mathcal{L}_2(n)$ into $n^{1/3}$ subsquares, each isomorphic to $\mathcal{L}_2\left(n^{2/3}\right)$. Searching any one of these subsquares by the previous algorithm takes time $O\left(\left(n^{2/3}\right)^{3/4}\right) = O(\sqrt{n})$, an amount of time that also suffices to travel to the subsquare and back from the start vertex. So using Grover's algorithm, the robot can search $\mathcal{L}_2(n)$ in time $O\left(\sqrt{n^{1/3}} \cdot \sqrt{n}\right) = O\left(n^{2/3}\right)$. We can continue recursively in this manner to make the running time approach $O(\sqrt{n})$. The trouble is that, with each additional layer of recursion, the robot needs to repeat the search more often to upper-bound the error probability. Using this approach, the best bounds we could obtain are roughly $O(\sqrt{n}\,\mathrm{polylog}\, n)$ for $d \geq 3$, or $\sqrt{n}2^{O\left(\sqrt{\log n}\right)}$ for $d = 2$. In what follows, we use the amplitude amplification approach of Brassard et al. [9] to improve these bounds, in the case of a single marked item, to $O(\sqrt{n})$ for $d \geq 3$ (Section 5.2) and $O\left(\sqrt{n}\log^2 n\right)$ for $d = 2$ (Section 5.3). Section 5.4 generalizes these results to the case of multiple marked items.

Intuitively, the reason the case $d = 2$ is special is that there, the diameter of the grid is $\Theta(\sqrt{n})$, which matches exactly the time needed for Grover search. For $d \geq 3$, by contrast, the robot can travel across the grid in much less time than is needed to search it.

5.1 Amplitude Amplification

We start by describing amplitude amplification [9], a generalization of Grover search. Let A be a quantum algorithm that, with probability ϵ, outputs a correct answer together with a witness that proves the answer correct. (For example, in the case of search, the algorithm outputs a vertex label i such that $x_i = 1$.) Amplification is a method that generates a new algorithm A' that calls A order $1/\sqrt{\epsilon}$ times, and produces both a correct answer and a witness with probability $\Omega(1)$. In the full version, we prove

Lemma 8 *If we are given a quantum algorithm with success probability ϵ, then by executing it $2m + 1$ times, where $m \leq \pi/\left(4\arcsin\sqrt{\epsilon}\right) - 1/2$, we can achieve success probability at least $\left(1 - (2m+1)^2\epsilon/3\right)(2m+1)^2\epsilon$.*

Intuitively, this lemma says that amplification to small probabilities is more efficient than amplification to larger probabilities. If $(2m+1)^2\epsilon$ is small, then $2m+1$ repetitions increase the probability by a factor of almost $(2m+1)^2$. Otherwise, the increase in probability is just by a factor of $\Omega\left((2m+1)^2\right)$.

5.2 Dimension At Least 3

Our goal is the following:

Theorem 9 *If $d \geq 3$, then $Q(\mathrm{OR}, \mathcal{L}_d) = \Theta(\sqrt{n})$.*

In this section, we prove Theorem 9 for the special case of a unique marked item; then, in Section 5.4, we will generalize to multiple marked items. Let $\mathrm{OR}^{(k)}$ be the problem of deciding whether there are no marked items or exactly k of them, given that one of these is true. Then:

Theorem 10 *If $d \geq 3$, then $Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_d\right) = \Theta\left(\sqrt{n}\right)$.*

Consider the following recursive algorithm $\mathcal{A}$. If the number of vertices n is at most some constant n_0, then search $\mathcal{L}_d\left(n\right)$ classically, returning 1 if a marked item is found and 0 otherwise. Otherwise subdivide $\mathcal{L}_d\left(n\right)$ into $n^{1/5}$ subcubes $S\left(1\right), \ldots, S\left(n^{1/5}\right)$, each isomorphic to $\mathcal{L}_d\left(n^{4/5}\right)$. Take the algorithm that consists of picking an $S\left(i\right)$ uniformly at random, and then running $\mathcal{A}$ recursively on $S\left(i\right)$. Amplify this algorithm m times for the smallest m such that $2m+1 \geq n^{1/11}$.

The above was a 'high-level description' of $\mathcal{A}$; in the full version, we give explicit pseudocode to show that $\mathcal{A}$ can be implemented using C-local unitaries. Here we content ourselves to analyze $\mathcal{A}$'s running time.

Lemma 11 *$\mathcal{A}$ finds the unique marked item (if it exists) with probability $\Omega\left(n^{-1/11}\right)$ in time $O\left(n^{5/11}\right)$.*

Proof. Let $T\left(n\right)$ be the number of steps used by $\mathcal{A}$ to search $\mathcal{L}_d\left(n\right)$. Then we obtain the recurrence

$$T\left(n\right) \leq n^{1/11}\left(T\left(n^{4/5}\right) + cn^{1/d}\right)$$

for some constant c. Since $cn^{1/d} < n^{2/5}$ for sufficiently large n, this resolves to

$$T(n) = O\left(n^{(1/11)\left[1+(4/5)+(4/5)^2+\cdots\right]}\right) = O\left(n^{5/11}\right).$$

Now let $P\left(n\right)$ be the probability that $\mathcal{A}$ finds the unique marked item in $\mathcal{L}_d\left(n\right)$. For the unamplified version, we obtain the recurrence

$$P\left(n\right) \geq n^{-1/5} P\left(n^{4/5}\right).$$

We now apply Lemma 8 to find the success probability of the amplified version. Since

$$\left(2m+1\right)^2 \epsilon \geq n^{2/11} n^{-1/5} P\left(n^{4/5}\right) = n^{-1/55} P\left(n^{4/5}\right),$$

we obtain the recurrence

$$P\left(n\right) \geq \left(1 - \frac{1}{3} n^{-1/55} P\left(n^{4/5}\right)\right) n^{-1/55} P(n^{4/5}).$$

Notice that if we consider the simpler recurrence $P'(n) \geq n^{-1/55} P'(n^{4/5})$, then

$$P'(n) \geq n^{(-1/55)\left[1+(4/5)+(4/5)^2+\cdots\right]} = \Omega\left(n^{-1/11}\right).$$

We claim that the multiplier of $1 - \frac{1}{3} n^{-1/55} P\left(n^{4/5}\right)$ does not affect the result substantially. Since $P(n^{4/5}) \leq 1$,

$$1 - \frac{1}{3} n^{-1/55} P(n^{4/5}) \geq 1 - \frac{1}{3} n^{-1/55}.$$

If we take the product of this expression over all n that appear in the algorithm (n, $n^{4/5}$, and so on down to n_0), the product is at least a constant. ■

Finally, we take $\mathcal{A}$ and amplify it to success probability $\Omega\left(1\right)$ by running it $O(n^{1/22})$ times. This gives an algorithm with overall running time $O\left(n^{5/11+1/22}\right) = O\left(n^{1/2}\right)$, establishing Theorem 10.

5.3 Dimension 2

The $d = 2$ case is trickier; here the best we can achieve is the following:

Theorem 12 $Q\left(\mathrm{OR}, \mathcal{L}_2\right) = O\left(\sqrt{n} \log^3 n\right)$.

Again, we start with the single marked item case and postpone the general case to Section 5.4.

Theorem 13 $Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_2\right) = O\left(\sqrt{n} \log^2 n\right)$.

For $d \geq 3$, we performed amplification on large (greater than $O\left(n^{-1/11}\right)$) probabilities only once, at the end. For $d = 2$, on the other hand, any algorithm that we construct with any nonzero success probability will have running time $\Omega\left(\sqrt{n}\right)$, simply because that is the diameter of the grid. If we want to keep the running time $O(n^{1/2+\varepsilon})$, then we can perform only $O\left(n^{\varepsilon}\right)$ amplification steps at the end. Therefore we need to keep the success probability relatively high throughout the recursion, meaning that we suffer an increase in the running time, since amplification to high probabilities is less efficient.

Our approach is as follows. We consider a sequence of algorithms, each of which has success probability $\Omega\left(1/\log n\right)$. The R^{th} algorithm $\mathcal{A}_R$ searches a square grid $\mathcal{L}_2\left(\log^R n\right)$, of size $\log^{R/2} n \times \log^{R/2} n$. If $R = 1$, then $\mathcal{A}_1$ just chooses one of the $\log n$ vertices uniformly at random, visits that vertex, and accepts if and only if it contains a marked item. If $R > 1$, then $\mathcal{A}_R$ subdivides the grid into $\log n$ subsquares, each isomorphic to $\mathcal{L}_2\left(\log^{R-1} n\right)$. It then chooses a subsquare uniformly at random and runs $\mathcal{A}_{R-1}$ on it. Finally (in the $R > 1$ case only) it applies m iterations of amplitude amplification to this process, m being the smallest integer such that $2m+1 \geq \sqrt{\log n}$. In the full version we show that if $R < \log n$, then $\mathcal{A}_R$ finds the unique marked item with probability $\Omega(1/\log n)$

in time $O\left(R\log^{(R+1)/2} n\right)$. The proof is similar to that of Lemma 11. Increasing the size to $\sqrt{n}\times\sqrt{n}$ requires $R=\log n/\log\log n$ levels of recursion. The running time of the resulting algorithm is $O\left(\sqrt{n}\log^{3/2} n\right)$. To amplify the success probability to $\Omega(1)$, we need $O\left(\sqrt{\log n}\right)$ repetitions of the whole algorithm, hence the bound of Theorem 13.

5.4 Multiple Marked Items

What about the case in which there are multiple i's with $x_i=1$? If there are k marked items (where k need not be known in advance), then Grover's algorithm can find a marked item with high probability in $O\left(\sqrt{n/k}\right)$ queries, as shown by Boyer et al. [8]. In our setting, however, this is too much to hope for—since even if there are many marked items, they might all be in a faraway part of the hypercube. Then $\Omega\left(n^{1/d}\right)$ steps are needed, even if $\sqrt{n/k}<n^{1/d}$. Indeed, in the full version we prove a stronger lower bound. Recall that $\mathrm{OR}^{(k)}$ is the problem of deciding whether there are no marked items or exactly k of them. Then we show that for $d\geq 2$, $Q\left(\mathrm{OR}^{(k)},\mathcal{L}_d\right)=\Omega\left(\sqrt{n}/k^{1/2-1/d}\right)$.

Notice that if $k\approx n$, then this bound becomes $\Omega\left(n^{1/d}\right)$ which is just the diameter of $\mathcal{L}_d$. Also, if $d=2$, then $1/2-1/d=0$ and the bound is simply $\Omega\left(\sqrt{n}\right)$ independent of k. The bound can be achieved for $d\geq 3$, and nearly achieved for $d=2$. The main idea is to reduce OR to $\mathrm{OR}^{(1)}$. In the case where k is known, the algorithm is as follows. Choose γ so that $\gamma/3<k<2\gamma/3$. Subdivide $\mathcal{L}_d(n)$ into n/γ subcubes. In each subcube, choose one vertex uniformly at random. Then run the algorithm for the unique-solution case (Theorem 10 or 13) on the chosen locations only, as if they were vertices of $\mathcal{L}_d(n/\gamma)$. It can be shown that, with probability at least $2/9$, there is exactly one marked vertex among the chosen vertices. Then the unique-solution algorithm finds this vertex. The running time works out to be $O\left(\sqrt{n}/k^{1/2-1/d}\right)$ for $d\geq 3$, or $O\left(\sqrt{n}\log^2 n\right)$ for $d=2$.

What if k is unknown? Let $\mathrm{OR}^{(\geq k)}$ be the problem of deciding whether there are no marked items or *at least* k of them, given that one of these is true. Then the straightforward 'doubling' approach of Boyer et al. [8] yields $Q\left(\mathrm{OR}^{(\geq k)},\mathcal{L}_d\right)=O\left(\sqrt{n}/k^{1/2-1/d}\right)$ and $Q\left(\mathrm{OR}^{(\geq k)},\mathcal{L}_2\right)=O\left(\sqrt{n}\log^3 n\right)$, completing Theorems 9 and 12.

6 Application to Disjointness

In this section we show how our results can be used to strengthen a seemingly unrelated result in quantum computing. Suppose Alice has a string $X=x_1\ldots x_n\in\{0,1\}^n$, and Bob has a string $Y=y_1\ldots y_n\in\{0,1\}^n$. In the *disjointness problem*, Alice and Bob must decide with high probability whether there exists an i such that $x_i=y_i=1$, using as few bits of communication as possible. Buhrman, Cleve, and Wigderson [10] observed that in the quantum setting, Alice and Bob can solve this problem using only $O\left(\sqrt{n}\log n\right)$ qubits of communication. This was subsequently improved by Høyer and de Wolf [14] to $O\left(\sqrt{n}c^{\log^* n}\right)$, where c is a constant and $\log^* n$ is the iterated logarithm function. Using the search algorithm of Theorem 9,[6] we can improve this to $O\left(\sqrt{n}\right)$, which matches the celebrated $\Omega\left(\sqrt{n}\right)$ lower bound of Razborov [17].

Theorem 14 *The quantum communication complexity of the disjointness problem is* $O\left(\sqrt{n}\right)$.

Proof. The protocol is as follows. Alice and Bob both store their inputs in a 3-D cube $\mathcal{L}_3(n)$; that is, they let $x_{jkl}=x_i$ and $y_{jkl}=y_i$, where $i=n^{2/3}j+n^{1/3}k+l+1$ and $j,k,l\in\left\{0,\ldots,n^{1/3}-1\right\}$. Throughout, they maintain a joint state of the form

$$\sum \alpha_{j,k,l,z_A,z_B}\left|v_{jkl},z_A\right\rangle\otimes\left|v_{jkl},z_B\right\rangle,$$

so if Alice's state is measured to be at location (j,k,l) of her cube, then Bob's state is at location (j,k,l) of his cube. To decide whether there exists an (j,k,l) with $x_{jkl}=y_{jkl}=1$, Alice simply runs the C-local implementation of our search algorithm, but with two changes. First, at each oracle step, Alice inverts her phase if and only if $x_{jkl}=y_{jkl}=1$; this requires 2 qubits of communication from Bob, to send y_{jkl} to Alice and then to erase it. Second, at each level of the recursion, after Alice prepares a uniform superposition over 'destination vertices,' she sends her destination vertex to Bob, requiring $\log\left(n^{1/5}\right)$ qubits. That way, for each basis state of Alice's that moves to subcube S of Alice's cube, a corresponding basis state of Bob's can move to subcube S of Bob's cube. Let $C(n)$ be the total number of qubits communicated. Then we obtain a recurrence analogous to that for the running time $T(n)$ in Lemma 11:

$$C(n)\leq n^{1/11}\left(C\left(n^{4/5}\right)+O\left(\log\left(n^{1/5}\right)\right)\right).$$

[6] Actually, the communication protocol depends only on the recursive structure of our algorithm, not on its being local with respect to a cube. But without the requirement of locality, we wouldn't have come up with the algorithm!

This resolves to $C(n) = O\left(n^{5/11}\right)$, for a protocol that succeeds with probability $\Omega\left(n^{-1/11}\right)$. Amplifying $O\left(n^{1/22}\right)$ times boosts the success probability to $\Omega(1)$ at the cost of increasing $C(n)$ to $O(\sqrt{n})$. ■

7 Search on Irregular Graphs

In Section 1.2, we claimed that our divide-and-conquer approach has the advantage of being *robust*: it works not only for highly symmetric graphs such as hypercubes, but for any graphs having comparable expansion properties. Let us now substantiate this claim.

Say a family of connected graphs $\{G_n = (V_n, E_n)\}$ is *d-dimensional* if there exists a $\beta > 0$ such that for all n, r and $v \in V_n$, $|B(v,r)| \geq \min\left(\beta r^d, n\right)$, where $B(v,r)$ is the induced subgraph of vertices having distance at most r from v in G_n. Intuitively, G_n is d-dimensional (for $d \geq 2$ an integer) if its expansion properties are *at least* as good as those of the hypercube $\mathcal{L}_d(n)$.[7] It is immediate that the diameter of G_n is at most $(n/\beta)^{1/d}$. Note, though, that G_n might not be an 'expander graph' in the usual sense, since we have not required that every sufficiently small *set* of vertices has many neighbors.

In the full version of the paper we show that if G is d-dimensional, then $Q(\mathrm{OR}, G) = O(\sqrt{n}\operatorname{polylog} n)$ for a constant $d > 2$, and $Q(\mathrm{OR}, G) = \sqrt{n}2^{O\left(\sqrt{\log n}\right)}$ for $d = 2$. To show this we describe an algorithm $\mathcal{A}^*$, which generalizes the algorithm $\mathcal{A}$ of Section 5.2. The intuition is simple: we want to decompose G recursively into subgraphs (called *clusters*), which will serve the same role as subcubes did in $\mathcal{A}$. However, the most obvious ways of implementing this idea fail. The reason is that our definition of d-dimensionality is quite permissive; it allows (for example) graphs that have wildly different expansion properties in different regions. Thus, if we take the clusters to be (possibly overlapping) Hamming balls of a fixed radius, then some clusters could have many more vertices than we want.

Our solution is to create the clusters bottom-up rather than top-down. We first choose vertices uniformly at random to be designated as 1-*pegs*. We then form 1-*clusters* by assigning each vertex to its closest 1-peg, as in a Voronoi diagram. For each 1-cluster C that has too many vertices, we split C up arbitrarily into several 1-clusters, each having the same 1-peg. In the next iteration, we choose random 2-pegs and use them to aggregate the 1-clusters into 2-clusters. We then form 3-clusters, 4-clusters, and so on.

[7] In general, it makes sense to consider non-integer d as well.

When this process is finished we have a tree of subgraphs, which a robot can search recursively exactly as in the hypercube case. Because of the splitting-up step, there may be more than one cluster with the same peg. This does not cause problems with unitarity, however, since we can just label each basis state with its corresponding cluster.

7.1 Bits Scattered on a Graph

In Section 2, we discussed several ways to pack a given amount of entropy into a spatial region of given dimensions. However, we said nothing about how the entropy is *distributed* within the region. It might be uniform, or concentrated on the boundary, or distributed in some other way. So we need to answer the following: suppose that in some graph, h out of the n vertices *might* contain a marked item, and we know which h those are. Then how many queries are needed to determine whether any of the h *does* contain a marked item? If the graph is the hypercube $\mathcal{L}_d$ for $d \geq 2$ or is d-dimensional for $d > 2$, then the results of the previous sections imply that $O(\sqrt{n}\operatorname{polylog} n)$ queries suffice. However, we wish to use fewer queries, taking advantage of the fact that h might be much smaller than n. Formally, given a graph G where (say) exactly h of the vertices have self-loops, let $\mathrm{OR}^{[h]}$ be problem of deciding whether any of the vertices with self-loops contains a marked item.

In the full version we show that, for all integer constants $d \geq 2$, there exists a d-dimensional graph G such that $Q\left(\mathrm{OR}^{[h]}, G\right) = \Omega\left(\sqrt{h}(n/h)^{1/d}\right)$. In particular, if $d = 2$ then $\Omega(\sqrt{n})$ time is always needed, since the vertices with self-loops might be far from the start vertex. Somewhat surprisingly, this lower bound can be achieved up to a polylogarithmic factor. We show that if G has h self-loops, and is d-dimensional for a constant $d > 2$, then $Q\left(\mathrm{OR}^{[h]}, G\right) = O\left(\sqrt{h}(n/h)^{1/d}\operatorname{polylog} h\right)$. Intuitively, this says that the worst case for search occurs when the h potential marked items are scattered evenly throughout the graph.

8 Open Problems

As discussed in Section 3.1, a salient open problem raised by this work is to prove relationships among Z-local, C-local, and H-local unitary matrices.

A second problem is to obtain interesting lower bounds in our model. For example, let G be a $\sqrt{n} \times \sqrt{n}$ grid, and suppose $f(X) = 1$ if and only if every row of G contains a vertex v_i with $x_i = 1$. Clearly $Q(f,G) = O(n^{3/4})$, and we conjecture that this is optimal. However, we were unable to show any lower bound better than $\Omega(\sqrt{n})$.

Finally, can the $O(\sqrt{n} \operatorname{polylog} n)$ bound for search on a 2-D grid be improved, perhaps even to $O(\sqrt{n})$?

9 Acknowledgments

We thank Jakob Bekenstein, Raphael Bousso, Andrew Childs, Wim van Dam, Julia Kempe, Greg Kuperberg, Seth Lloyd, John Preskill, Neil Shenvi, Mario Szegedy, Umesh Vazirani, John Watrous, and Ronald de Wolf for helpful discussions.

References

[1] D. Aharonov, A. Ambainis. J. Kempe, and U. Vazirani. Quantum walks on graphs, STOC'2001, pp. 50–59, 2001. quant-ph/0012090.

[2] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials, FOCS'98, pp. 352–361, 1998. quant-ph/9802049.

[3] J. D. Bekenstein. A universal upper bound on the entropy to energy ratio for bounded systems, Phys. Rev. D23:287, 1981.

[4] P. Benioff. Space searches with a quantum robot, Quantum Computation & Information (S. J. Lomonaco and H. E. Brandt, eds.), 2002. quant-ph/0003006.

[5] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing, SIAM J. Comput. 26(5):1510–1523, 1997. quant-ph/9701001.

[6] R. Bousso. Positive vacuum energy and the N-bound, J. High Energy Phys. 0011:038, 2000. hep-th/0010252.

[7] R. Bousso. The holographic principle, Rev. Mod. Phys. 74(3), 2002. hep-th/0203101.

[8] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching, Fortschritte Der Physik 46(4-5):493–505, 1998. quant-ph/9605034.

[9] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation, Quantum Computation & Information (S. J. Lomonaco and H. E. Brandt, eds.), 2002. quant-ph/0005055.

[10] H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation, STOC'98, pp. 63–68, 1998. quant-ph/9702040.

[11] A. M. Childs and J. Goldstone. Spatial search by quantum walk, submitted, 2003. quant-ph/0306054.

[12] G. Gour. Extensive entropy bounds, Phys. Rev. D67, 2003. gr-qc/0212087.

[13] L. K. Grover. A fast quantum mechanical algorithm for database search, STOC'96, pp. 212–219, 1996. quant-ph/9605043.

[14] P. Høyer and R. de Wolf. Improved quantum communication complexity bounds for disjointness and equality, STACS'2002, LNCS 2285, pp. 299–310, 2002. quant-ph/0109068.

[15] S. Lloyd. Computational capacity of the universe, Phys. Rev. Lett. 88 (237901), 2002. quant-ph/0110141.

[16] S. Perlmutter and 32 others (Supernova Cosmology Project), Measurements of Ω and Λ from 42 high-redshift supernovae, Astrophysical Journal 517(2):565–586, 1999. astro-ph/9812133.

[17] A. A. Razborov. Quantum communication complexity of symmetric predicates (Russian), Izvestiya Math. 6, 2002. English version at quant-ph/0204025.

[18] T. Rudolph and L. Grover. Quantum searching a classical database (or how we learned to stop worrying and love the bomb), 2002. quant-ph/0206066.

[19] N. Shenvi, J. Kempe, and K. B. Whaley. A quantum random walk search algorithm, Phys. Rev. A67 (052307), 2003. quant-ph/0210064.

[20] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. 26(5):1484–1509, 1997. quant-ph/9508027.

[21] J. Watrous. On one-dimensional quantum cellular automata, FOCS'95, pp. 528–537, 1995.

[22] Ch. Zalka. Could Grover's algorithm help in searching an actual database?, 1999. quant-ph/9901068.