THEORY OF COMPUTING, Volume 1 (2005), pp. 47–79
http://theoryofcomputing.org

Quantum Search of Spatial Regions

Scott Aaronson* Andris Ambainis†

Received: June 13, 2004; published: July 13, 2005.

Abstract: Can Grover's 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 $O(\sqrt{n}\log^{5/2} n)$ 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.

ACM Classification: F.1.2, F.1.3

AMS Classification: 81P68, 68Q10

Key words and phrases: Quantum computing, Grover search, amplitude amplification, quantum communication complexity, disjointness, lower bounds

*This work was mostly done while the author was a PhD student at UC Berkeley, supported by an NSF Graduate Fellowship.

†Supported by an IQC University Professorship and by CIAR. This work was mostly done while the author was at the University of Latvia.

DOI: 10.4086/toc.2005.v001a004

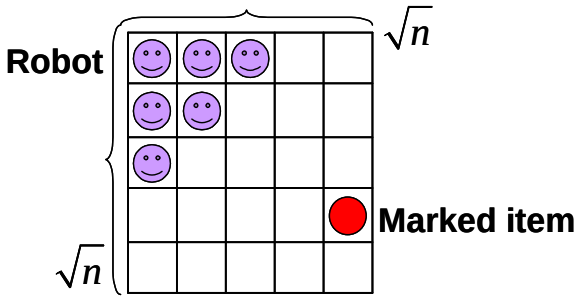

Figure 1: A quantum robot, in a superposition over locations, searching for a marked item on a 2D grid of size $\sqrt{n} \times \sqrt{n}$.

1 Introduction

The goal of Grover's quantum search algorithm [17, 18] 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 [29]. 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 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 [9]. 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 (Figure 1)?

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 for a unique marked vertex in time $O\left(\sqrt{n}\log^{3/2} n\right)$ when $d=2$, or $O(\sqrt{n})$ when $d \geq 3$. 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 $O\left(n^{1/2+\varepsilon}\right)$ follows readily. However, to obtain the tighter bounds is more difficult;

	$d=2$	$d>2$
Hypercube, 1 marked item	$O\left(\sqrt{n}\log^{3/2} n\right)$	$\Theta(\sqrt{n})$
Hypercube, k or more marked items	$O\left(\sqrt{n}\log^{5/2} n\right)$	$\Theta\left(\frac{\sqrt{n}}{k^{1/2-1/d}}\right)$
Arbitrary graph, k or more marked items	$\sqrt{n}2^{O\left(\sqrt{\log n}\right)}$	$\widetilde{\Theta}\left(\frac{\sqrt{n}}{k^{1/2-1/d}}\right)$

Table 1: Upper and lower bounds for quantum search on a d-dimensional graph given in this paper. The symbol $\widetilde{\Theta}$ means that the upper bound includes a polylogarithmic term. Note that, if $d=2$, then $\Omega(\sqrt{n})$ is always a lower bound, for any number of marked items.

for that we use the amplitude-amplification framework of Grover [19] and Brassard et al. [11].

Section 5 presents the main results; Section 5.4 shows further that, when there are k or more marked vertices, the search time becomes $O\left(\sqrt{n}\log^{5/2} n\right)$ when $d=2$, or $\Theta\left(\sqrt{n}/k^{1/2-1/d}\right)$ when $d\geq 3$. Also, Section 6 generalizes our algorithm to arbitrary graphs that have 'hypercube-like' expansion properties. Here the best bounds we can achieve are $\sqrt{n}2^{O\left(\sqrt{\log n}\right)}$ when $d=2$, or $O(\sqrt{n}\,\mathrm{polylog}\, n)$ when $d>2$ (note that d need not be an integer). Table 1 summarizes the results.

Section 7 shows, as an unexpected application of our search algorithm, that the quantum communication complexity of the well-known *disjointness problem* is $O(\sqrt{n})$. This improves an $O\left(\sqrt{n}c^{\log^* n}\right)$ upper bound of Høyer and de Wolf [20], and matches the $\Omega(\sqrt{n})$ lower bound of Razborov [23].

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. [4], 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 [31], of what 'local' means. Section 4 proves 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 proof (using the hybrid argument of Bennett et al. [7]) that this upper bound is tight for certain graphs. We conclude in Section 8 with some open problems.

1.2 Related Work

In a paper on 'Space searches with a quantum robot,' Benioff [6] 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 $\Theta(\sqrt{n})$ 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

	$d=2$	$d=3$	$d=4$	$d\geq 5$
This paper	$O\left(\sqrt{n}\log^{3/2} n\right)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
[16]	$O(n)$	$O\left(n^{5/6}\right)$	$O(\sqrt{n}\log n)$	$O(\sqrt{n})$
[3, 15]	$O(\sqrt{n}\log n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$

Table 2: Time needed to find a unique marked item in a d-dimensional hypercube, using the divide-and-conquer algorithms of this paper, the original quantum walk algorithm of Childs and Goldstone [16], and the improved walk algorithms of Ambainis, Kempe, and Rivosh [3] and Childs and Goldstone [15].

can search a 2-D grid for a single marked vertex in $O\left(\sqrt{n}\log^{3/2} n\right)$ time. 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 [30]—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, 13, 14, 28]. In an earlier version of this paper, we asked whether quantum walks might yield an alternative spatial search algorithm, possibly even one that outperforms our divide-and-conquer algorithm. Motivated by this question, Childs and Goldstone [16] managed to show that in the continuous-time setting, a quantum walk can search a d-dimensional hypercube for a single marked vertex in time $O(\sqrt{n}\log n)$ when $d=4$, or $O(\sqrt{n})$ when $d\geq 5$. Our algorithm was still faster in 3 or fewer dimensions (see Table 2). Subsequently, however, Ambainis, Kempe, and Rivosh [3] gave an algorithm based on a discrete-time quantum walk, which was as fast as ours in 3 or more dimensions, and faster in 2 dimensions. In particular, when $d=2$ their algorithm used only $O(\sqrt{n}\log n)$ time to find a unique marked vertex. Childs and Goldstone [15] then gave a continuous-time quantum walk algorithm with the same performance, and related this algorithm to properties of the Dirac equation. It is still open whether $O(\sqrt{n})$ time is achievable in 2 dimensions.

Currently, the main drawback of the quantum walk approach is that all analyses have relied heavily on symmetries in the underlying graph. If even minor 'defects' are introduced, it is no longer known how to upper-bound the running time. By contrast, the analysis of our divide-and-conquer algorithm is elementary, and does not depend on eigenvalue bounds. We can therefore show that the algorithm works for any graphs with sufficiently good expansion properties.

Childs and Goldstone [16] argued that the quantum walk approach has the advantage of requiring fewer auxiliary qubits than the divide-and-conquer approach. However, the need for many qubits was an artifact of how we implemented the algorithm in a previous version of the paper. The current version uses only *one* qubit.

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* [9], 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, Bousso [9] shows that an enclosed ball of radiation with radius r can store $n = \Theta\left(r^{3/2}\right)$ bits, even though its energy grows only as r. Our results in Section 6.1 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 will show in Theorem 6.7 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 $O(\sqrt{n}\,\mathrm{polylog}\,n)$, 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* [5], an even stronger entropy constraint than the holographic bound.

[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—as recent evidence suggests that they are [27]—takes time proportional to r^3, which is much too long.

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 [6]), or classical search. Our results show that it is possible.

From a physical point of view, several questions naturally arise: (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 ([32], though see also [24]). 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 seems physically unobjectionable. For a particle (and indeed any object) really does have an indeterminate location, not merely an indeterminate 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, should we not 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 [8] 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 [26] on Λ.[3] 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[4], 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

[3] Also, Lloyd [21] 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 $\left(10^{61}\right)^2 = 10^{122}$. Lloyd's bound, unlike Bousso's, does not depend on Λ being positive. The numerical coincidence between the two bounds reflects the experimental finding [26, 25] 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.

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

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 $O\left(\frac{1}{\sqrt{\Lambda}}\text{polylog}\frac{1}{\sqrt{\Lambda}}\right)$. How anyone could *prepare* a database of size much greater than $1/\sqrt{\Lambda}$ remains unclear, but if such a database existed, it could be searched!

3 The Model

Much of what is known about the power of quantum computing comes from the *black-box* or *query* model [2, 4, 7, 17, 29], 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:

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

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 criteria for whether a unitary matrix U is local. Our algorithms will then be implemented using the most restrictive of these criteria.

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 $U_{i,z\to i^*,z^*}$ be the entry in the $|v_i,z\rangle$ column and $|v_{i^*},z^*\rangle$ row of U.

Definition 3.1. U is Z-local if $U_{i,z\to i^*,z^*}=0$ whenever $i\neq i^*$ and (v_i,v_{i^*}) is not an edge of G.

Definition 3.2. U is C-local if the basis states can be partitioned into subsets $P_1,\ldots,P_q$ such that

(i) $U_{i,z\to i^*,z^*}=0$ 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 3.3. U is H-local if $U=e^{iH}$ for some Hermitian H with eigenvalues of absolute value at most π, such that $H_{i,z\to i^*,z^*}=0$ 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. For the latter implication, note that any unitary U can be written as e^{iH} for some H with eigenvalues of absolute value at most π. So we can write the unitary U_j acting on each P_j as e^{iH_j}; then since the U_j's commute,

$$\prod U_j = e^{i\sum H_j}.$$

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? Let us emphasize that these questions are *not* answered by, for example, the Solovay-Kitaev theorem (see [22]), that an $n\times n$ unitary matrix can be approximated using a number of gates polynomial in n. For recall that the definition of C-locality requires the edgewise operations to commute—indeed, without that requirement, one could produce any unitary matrix at all. So the relevant 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.)

4 General Bounds

Given a Boolean function $f:\{0,1\}^n\to\{0,1\}$, the quantum query complexity $Q(f)$, defined by Beals et al. [4], 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 let $Q(f,G)$ be the minimum T for which there exists a T-query quantum robot on G that evaluates f

with probability $2/3$. Here we require the algorithm steps to be C-local. One might also consider the corresponding measures $Q^Z(f,G)$ and $Q^H(f,G)$ with Z-local and H-local steps respectively. Clearly $Q(f,G) \geq Q^Z(f,G)$ and $Q(f,G) \geq Q^H(f,G)$; we conjecture that all three measures are asymptotically equivalent but were unable to prove this.

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

Proposition 4.1. *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.*

Proof.

(i) Starting from the root, a spanning tree for G can be traversed in $2(n-1)-1$ steps (there is no need to return to the root).

(ii) We can simulate a query in $2\delta_G$ steps, by fanning out from the start vertex v_1 and then returning. Applying a unitary at v_1 takes 1 step.

(iii) Obvious.

(iv) There exists a vertex v_i whose distance to v_1 is at least $\delta_G/2$, and f could depend on x_i.

□

We now show that the model is robust.

Proposition 4.2. *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 $1-\varepsilon$, for a constant $\varepsilon > 0$.

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

Proof. (i) We can transform $|v_1,0\rangle$ to $|\psi\rangle$ (and hence $|\psi\rangle$ to $|v_1,0\rangle$) in $\delta_G = O(Q(f,G))$ steps, by fanning out from v_1 along the edges of a minimum-height spanning tree.

(ii) Assume without loss of generality that z_{OUT} is accessed only once, to write the output. Then after z_{OUT} is accessed, uncompute (that is, run the algorithm backwards) to localize the final state at v_1. The state can then be localized at any v_i in $\delta_G = O(Q(f,G))$ steps. We can succeed with any constant probability by repeating this procedure a constant number of times.

(iii) The oracle step O is its own inverse, so we can implement a sequence $U_1, U_2, \ldots$ of algorithm steps as follows (where I is the identity):

$$U_1 \to O \to I \to O \to U_2 \to \cdots$$

□

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 sometimes omit floor and ceiling signs if they clearly have no effect on the asymptotics.)

Proposition 4.3.

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

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 depth-first search 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 4.3 is tight:

Theorem 4.4. *For all δ, there exists a graph G with diameter $\delta_G = \delta$ such that*

$$Q(\mathrm{OR}, G) = \Omega\left(\sqrt{n\delta}\right).$$

Proof. Let G be a 'starfish' with central vertex v_1 and $M = 2(n-1)/\delta$ legs $L_1, \ldots, L_M$, each of length $\delta/2$ (see Figure 2). We use the hybrid argument of Bennett et al. [7]. Suppose we run the algorithm on the all-zero input X_0. Then define the *query magnitude* $\Gamma_j^{(t)}$ to be the probability of finding the robot in leg L_j immediately after the t^{th} query:

$$\Gamma_j^{(t)} = \sum_{v_i \in L_j} \sum_z \left|\alpha_{i,z}^{(t)}(X_0)\right|^2.$$

Let T be the total number of queries, and let $w = T/(c\delta)$ for some constant $0 < c < 1/2$. Clearly

$$\sum_{q=0}^{w-1} \sum_{j=1}^{M} \Gamma_j^{(T-qc\delta)} \leq \sum_{q=0}^{w-1} 1 = w.$$

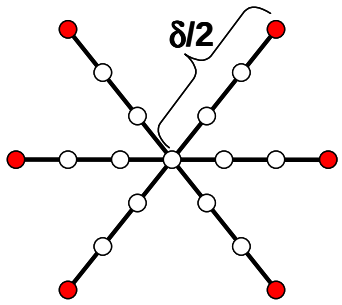

Figure 2: The 'starfish' graph G. The marked item is at one of the tip vertices.

Hence there must exist a leg L_{j^*} such that

$$\sum_{q=0}^{w-1} \Gamma_{j^*}^{(T-qc\delta)} \leq \frac{w}{M} = \frac{w\delta}{2(n-1)}.$$

Let v_{i^*} be the tip vertex of L_{j^*}, and let Y be the input which is 1 at v_{i^*} and 0 elsewhere. Then let X_q be a hybrid input, which is X_0 during queries 1 to $T-qc\delta$, but Y during queries $T-qc\delta+1$ to T. Also, let

$$\left|\psi^{(t)}(X_q)\right\rangle = \sum_{i,z} \alpha_{i,z}^{(t)}(X_q)\,|v_i,z\rangle$$

be the algorithm's state after t queries when run on X_q, and let

$$\begin{aligned} D(q,r) &= \left\| \left|\psi^{(T)}(X_q)\right\rangle - \left|\psi^{(T)}(X_r)\right\rangle \right\|_2^2 \\ &= \sum_{v_i \in G} \sum_z \left| \alpha_{i,z}^{(T)}(X_q) - \alpha_{i,z}^{(T)}(X_r) \right|^2. \end{aligned}$$

Then for all $q \geq 1$, we claim that $D(q-1,q) \leq 4\Gamma_{j^*}^{(T-qc\delta)}$. For by unitarity, the Euclidean distance between $\left|\psi^{(t)}(X_{q-1})\right\rangle$ and $\left|\psi^{(t)}(X_q)\right\rangle$ can only increase as a result of queries $T-qc\delta+1$ through $T-(q-1)c\delta$. But no amplitude from outside L_{j^*} can reach v_{i^*} during that interval, since the distance is $\delta/2$ and there are only $c\delta < \delta/2$ time steps. Therefore, switching from X_{q-1} to X_q can only affect amplitude that is in L_{j^*} immediately after query $T-qc\delta$:

$$\begin{aligned} D(q-1,q) &\leq \sum_{v_i \in L_{j^*}} \sum_z \left| \alpha_{i,z}^{(T-qc\delta)}(X_q) - \left(-\alpha_{i,z}^{(T-qc\delta)}(X_q)\right) \right|^2 \\ &= 4 \sum_{v_i \in L_{j^*}} \sum_z \left| \alpha_{i,z}^{(T-qc\delta)}(X_0) \right|^2 = 4\Gamma_{j^*}^{(T-qc\delta)}. \end{aligned}$$

It follows that

$$\sqrt{D(0,w)} \leq \sum_{q=1}^{w} \sqrt{D(q-1,q)} \leq 2\sum_{q=1}^{w} \sqrt{\Gamma_{j^*}^{(T-qc\delta)}} \leq 2w\sqrt{\frac{\delta}{2(n-1)}} = \frac{T}{c}\sqrt{\frac{2}{\delta(n-1)}}.$$

Here the first inequality uses the triangle inequality, and the third uses the Cauchy-Schwarz inequality. Now assuming the algorithm is correct we need $D(0,w) = \Omega(1)$, which implies that $T = \Omega\left(\sqrt{n\delta}\right)$. □

It is immediate that Theorem 4.4 applies to Z-local unitaries as well as C-local ones: that is, $Q^Z(\mathrm{OR}, G) = \Omega\left(\sqrt{n\delta}\right)$. We believe the theorem can be extended to H-local unitaries as well, but a full discussion of this issue would take us too far afield.

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 partition $\mathcal{L}_2(n)$ into $\sqrt{n}$ subsquares, each a copy of $\mathcal{L}_2(\sqrt{n})$. In $5\sqrt{n}$ steps, the robot can travel from the start vertex to any subsquare C, search C classically for a marked vertex, and then return to the start vertex. Thus, by searching all $\sqrt{n}$ of the C'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 partition $\mathcal{L}_2(n)$ into $n^{1/3}$ subsquares, each a copy of $\mathcal{L}_2\left(n^{2/3}\right)$. 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 Grover [19] and Brassard et al. [11] to improve these bounds, in the case of a single marked vertex, to $O(\sqrt{n})$ for $d \geq 3$ (Section 5.2) and $O\left(\sqrt{n}\log^{3/2} n\right)$ for $d = 2$ (Section 5.3). Section 5.4 generalizes these results to the case of multiple marked vertices.

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 [11, 19], a generalization of Grover search. Let $\mathcal{U}$ 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 generates a new algorithm that calls $\mathcal{U}$ order $1/\sqrt{\varepsilon}$ times, and that produces both a correct answer and a witness with probability $\Omega(1)$. In particular, assume $\mathcal{U}$ starts in basis state $|s\rangle$, and let m be a positive integer. Then the amplification procedure works as follows:

(1) Set $|\psi_0\rangle = \mathcal{U}|s\rangle$.

(2) For $i = 1$ to m set $|\psi_{i+1}\rangle = \mathcal{U}S\mathcal{U}^{-1}W|\psi_i\rangle$, where

- W flips the phase of basis state $|y\rangle$ if and only if $|y\rangle$ contains a description of a correct witness, and
- S flips the phase of basis state $|y\rangle$ if and only if $|y\rangle = |s\rangle$.

We can decompose $|\psi_0\rangle$ as $\sin\alpha\,|\Psi_{\text{succ}}\rangle + \cos\alpha\,|\Psi_{\text{fail}}\rangle$, where $|\Psi_{\text{succ}}\rangle$ is a superposition over basis states containing a correct witness and $|\Psi_{\text{fail}}\rangle$ is a superposition over all other basis states. Brassard et al. [11] showed the following:

Lemma 5.1 ([11]). $|\psi_i\rangle = \sin[(2i+1)\alpha]\,|\Psi_{\text{succ}}\rangle + \cos[(2i+1)\alpha]\,|\Psi_{\text{fail}}\rangle$.

If measuring $|\psi_0\rangle$ gives a correct witness with probability ε, then $|\sin\alpha|^2 = \varepsilon$ and $|\alpha| \geq 1/\sqrt{\varepsilon}$. So taking $m = O(1/\sqrt{\varepsilon})$ yields $\sin[(2m+1)\alpha] \approx 1$. For our algorithms, though, the multiplicative constant under the big-O also matters. To upper-bound this constant, we prove the following lemma.

Lemma 5.2. *Suppose a quantum algorithm $\mathcal{U}$ outputs a correct answer and witness with probability exactly ε. Then by using $2m+1$ calls to $\mathcal{U}$ or $\mathcal{U}^{-1}$, where*

$$m \leq \frac{\pi}{4\arcsin\sqrt{\varepsilon}} - \frac{1}{2},$$

we can output a correct answer and witness with probability at least

$$\left(1 - \frac{(2m+1)^2}{3}\varepsilon\right)(2m+1)^2\varepsilon.$$

Proof. We perform m steps of amplitude amplification, which requires $2m+1$ calls to $\mathcal{U}$ or $\mathcal{U}^{-1}$. By Lemma 5.1, this yields the final state

$$\sin[(2m+1)\alpha]\,|\Psi_{\text{succ}}\rangle + \cos[(2m+1)\alpha]\,|\Psi_{\text{fail}}\rangle$$

where $\alpha = \arcsin\sqrt{\varepsilon}$. Therefore the success probability is

$$\begin{aligned}
\sin^2\left[(2m+1)\arcsin\sqrt{\varepsilon}\right] &\geq \sin^2\left[(2m+1)\sqrt{\varepsilon}\right] \\
&\geq \left((2m+1)\sqrt{\varepsilon} - \frac{(2m+1)^3}{6}\varepsilon^{3/2}\right)^2 \\
&\geq (2m+1)^2\varepsilon - \frac{(2m+1)^4}{3}\varepsilon^2.
\end{aligned}$$

Here the first line uses the monotonicity of $\sin^2 x$ in the interval $[0,\pi/2]$, and the second line uses the fact that $\sin x \geq x - x^3/6$ for all $x \geq 0$ by Taylor series expansion. □

Note that there is no need to uncompute any garbage left by $\mathcal{U}$, beyond the uncomputation that happens "automatically" within the amplification procedure.

5.2 Dimension At Least 3

Our goal is the following:

Theorem 5.3. *If $d \geq 3$, then $Q(\mathrm{OR}, \mathcal{L}_d) = \Theta(\sqrt{n})$.*

In this section, we prove Theorem 5.3 for the special case of a unique marked vertex; then, in Sections 5.4 and 5.5, we will generalize to multiple marked vertices. Let $\mathrm{OR}^{(k)}$ be the problem of deciding whether there are no marked vertices or exactly k of them, given that one of these is true. Then:

Theorem 5.4. *If $d \geq 3$, then $Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_d\right) = \Theta(\sqrt{n})$.*

Choose constants $\beta \in (2/3, 1)$ and $\mu \in (1/3, 1/2)$ such that $\beta\mu > 1/3$ (for example, $\beta = 4/5$ and $\mu = 5/11$ will work). Let ℓ_0 be a large positive integer; then for all positive integers R, let $\ell_R = \ell_{R-1} \left\lceil \ell_{R-1}^{1/\beta - 1} \right\rceil$. Also let $n_R = \ell_R^d$. Assume for simplicity that $n = n_R$ for some R; in other words, that the hypercube $\mathcal{L}_d(n_R)$ to be searched has sides of length ℓ_R. Later we will remove this assumption.

Consider the following recursive algorithm $\mathcal{A}$. If $n = n_0$, then search $\mathcal{L}_d(n_0)$ classically, returning 1 if a marked vertex is found and 0 otherwise. Otherwise partition $\mathcal{L}_d(n_R)$ into n_R/n_{R-1} subcubes, each one a copy of $\mathcal{L}_d(n_{R-1})$. Take the algorithm that consists of picking a subcube C uniformly at random, and then running $\mathcal{A}$ recursively on C. Amplify this algorithm $(n_R/n_{R-1})^{\mu}$ times.

The intuition behind the exponents is that $n_{R-1} \approx n_R^{\beta}$, so searching $\mathcal{L}_d(n_{R-1})$ should take about $n_R^{\beta/2}$ steps, which dominates the $n_R^{1/d}$ steps needed to travel across the hypercube when $d \geq 3$. Also, at level R we want to amplify a number of times that is less than $(n_R/n_{R-1})^{1/2}$ by some polynomial amount, since full amplification would be inefficient. The reason for the constraint $\beta\mu > 1/3$ will appear in the analysis.

We now provide a more explicit description of $\mathcal{A}$, which shows that it can be implemented using C-local unitaries and only a single bit of workspace. At any time, the quantum robot's state will have the form $\sum_{i,z} \alpha_{i,z} |v_i, z\rangle$, where v_i is a vertex of $\mathcal{L}_d(n_R)$ and z is a single bit that records whether or not a marked vertex has been found. Given a subcube C, let $v(C)$ be the "corner" vertex of C; that is, the vertex that is minimal in all d coordinates. Then the initial state when searching C will be $|v(C), 0\rangle$. Beware, however, that "initial state" in this context just means the state $|s\rangle$ from Section 5.1. Because of the way amplitude amplification works, $\mathcal{A}$ will often be invoked on C with other initial states, and even run in reverse.

For convenience, we will implement $\mathcal{A}$ using a two-stage recursion: given any subcube, the task of $\mathcal{A}$ will be to amplify the result of another procedure called $\mathcal{U}$, which in turn runs $\mathcal{A}$ recursively on smaller subcubes. We will also use the conditional phase flips W and S from Section 5.1. For convenience, we write $\mathcal{A}_R, \mathcal{U}_R, W_R, S_R$ to denote the level of recursion that is currently active. Thus, $\mathcal{A}_R$ calls $\mathcal{U}_R$, which calls $\mathcal{A}_{R-1}$, which calls $\mathcal{U}_{R-1}$, and so on down to $\mathcal{A}_0$.

Algorithm 5.5 ($\mathcal{A}_R$). *Searches a subcube C of size n_R for the marked vertex, and amplifies the result to have larger probability. Default initial state: $|v(C),0\rangle$.*

If** $R=0$ **then:

(1) Use classical C-local operations to visit all n_0 vertices of C in any order. At each $v_i \in C$, use a query transformation to map the state $|v_i,z\rangle$ to $|v_i,z\oplus x_i\rangle$.

(2) Return to $v(C)$.

If** $R \geq 1$ **then:

(1) Let m_R be the smallest integer such that $2m_R+1 \geq (n_R/n_{R-1})^{\mu}$.

(2) Call $\mathcal{U}_R$.

(3) For $i=1$ to m_R, call W_R, then $\mathcal{U}_R^{-1}$, then S_R, then $\mathcal{U}_R$.

Suppose $\mathcal{A}_R$ is run on the initial state $|v(C),0\rangle$, and let $C_1,\ldots,C_{n_R/n_0}$ be the *minimal subcubes* in C—meaning those of size n_0. Then the final state after $\mathcal{A}_R$ terminates should be

$$\frac{1}{\sqrt{n_R/n_0}} \sum_{i=1}^{n_R/n_0} |v(C_i),0\rangle$$

if C does not contain the marked vertex. Otherwise the final state should have non-negligible overlap with $|v(C_{i^*}),1\rangle$, where C_{i^*} is the minimal subcube in C that contains the marked vertex. In particular, if $R=0$, then the final state should be $|v(C),1\rangle$ if C contains the marked vertex, and $|v(C),0\rangle$ otherwise.

The two phase-flip subroutines, W_R and S_R, are both trivial to implement. To apply W_R, map each basis state $|v_i,z\rangle$ to $(-1)^z|v_i,z\rangle$. To apply S_R, map each $|v_i,z\rangle$ to $-|v_i,z\rangle$ if $z=0$ and $v_i=v(C)$ for some subcube C of size n_R, and to $|v_i,z\rangle$ otherwise. Below we give pseudocode for $\mathcal{U}_R$.

Algorithm 5.6 ($\mathcal{U}_R$). *Searches a subcube C of size n_R for the marked vertex. Default initial state: $|v(C),0\rangle$.*

(1) Partition C into n_R/n_{R-1} smaller subcubes $C_1,\ldots,C_{n_R/n_{R-1}}$, each of size n_{R-1}.

(2) For all $j \in \{1,\ldots,d\}$, let V_j be the set of corner vertices $v(C_i)$ that differ from $v(C)$ only in the first j coordinates. Thus $V_0=\{v(C)\}$, and in general $|V_j| = (\ell_R/\ell_{R-1})^j$. For $j=1$ to d, let $|V_j\rangle$ be the state

$$|V_j\rangle = \frac{1}{\ell_R^{j/2}} \sum_{v(C_i)\in V_j} |v(C_i),0\rangle$$

Apply a sequence of transformations Z_1, Z_2, …, Z_d where Z_j is a unitary that maps $|V_{j-1}\rangle$ to $|V_j\rangle$ by applying C-local unitaries that move amplitude only along the j^{th} coordinate.

(3) Call $\mathcal{A}_{R-1}$ recursively. (Note that this searches $C_1,\ldots,C_{n_R/n_{R-1}}$ in superposition. Also, the required amplification is performed for each of these subcubes automatically by step (3) of $\mathcal{A}_{R-1}$.)

If $\mathcal{U}_R$ is run on the initial state $|v(C),0\rangle$, then the final state should be

$$\frac{1}{\sqrt{n_R/n_{R-1}}}\sum_{i=1}^{n_R/n_0}|\phi_i\rangle,$$

where $|\phi_i\rangle$ is the correct final state when $\mathcal{A}_{R-1}$ is run on subcube C_i with initial state $|v(C_i),0\rangle$. A key point is that there is no need for $\mathcal{U}_R$ to call $\mathcal{A}_{R-1}$ twice, once to compute and once to uncompute—for the uncomputation is already built into $\mathcal{A}_R$. This is what will enable us to prove an upper bound of $O(\sqrt{n})$ instead of $O\left(\sqrt{n}2^R\right)=O(\sqrt{n}\,\mathrm{polylog}\,n)$.

We now analyze the running time of $\mathcal{A}_R$.

Lemma 5.7. *$\mathcal{A}_R$ uses $O\left(n_R^\mu\right)$ steps.*

Proof. Let $T_{\mathcal{A}}(R)$ and $T_{\mathcal{U}}(R)$ be the total numbers of steps used by $\mathcal{A}_R$ and $\mathcal{U}_R$ respectively in searching $\mathcal{L}_d(n_R)$. Then we have $T_{\mathcal{A}}(0)=O(1)$, and

$$\begin{aligned}T_{\mathcal{A}}(R)&\le(2m_R+1)\,T_{\mathcal{U}}(R)+2m_R\\ T_{\mathcal{U}}(R)&\le dn_R^{1/d}+T_{\mathcal{A}}(R-1)\end{aligned}$$

for all $R\ge 1$. For W_R and S_R can both be implemented in a single step, while $\mathcal{U}_R$ uses $d\ell_R=dn_R^{1/d}$ steps to move the robot across the hypercube. Combining,

$$\begin{aligned}T_{\mathcal{A}}(R)&\le(2m_R+1)\left(dn_R^{1/d}+T_{\mathcal{A}}(R-1)\right)+2m_R\\ &\le\left((n_R/n_{R-1})^\mu+2\right)\left(dn_R^{1/d}+T_{\mathcal{A}}(R-1)\right)+(n_R/n_{R-1})^\mu+1\\ &=O\left((n_R/n_{R-1})^\mu\,n_R^{1/d}\right)+\left((n_R/n_{R-1})^\mu+2\right)T_{\mathcal{A}}(R-1)\\ &=O\left((n_R/n_{R-1})^\mu\,n_R^{1/d}\right)+(n_R/n_{R-1})^\mu\,T_{\mathcal{A}}(R-1)\\ &=O\left((n_R/n_{R-1})^\mu\,n_R^{1/d}+(n_R/n_{R-2})^\mu\,n_{R-1}^{1/d}+\cdots+(n_R/n_0)^\mu\,n_1^{1/d}\right)\\ &=n_R^\mu\cdot O\left(\frac{n_R^{1/d}}{n_{R-1}^\mu}+\frac{n_{R-1}^{1/d}}{n_{R-2}^\mu}+\cdots+\frac{n_1^{1/d}}{n_0^\mu}\right)\\ &=n_R^\mu\cdot O\left(n_R^{1/d-\beta\mu}+\cdots+n_2^{1/d-\beta\mu}+n_1^{1/d-\beta\mu}\right)\\ &=n_R^\mu\cdot O\left(n_R^{1/d-\beta\mu}+\left(n_R^{1/d-\beta\mu}\right)^{1/\beta}+\cdots+\left(n_R^{1/d-\beta\mu}\right)^{1/\beta^{R-1}}\right)\\ &=O\left(n_R^\mu\right).\end{aligned}$$

Here the second line follows because $2m_R+1\le(n_R/n_{R-1})^\mu+2$, the fourth because the $(n_R/n_{R-1})^\mu$ terms increase doubly exponentially, so adding 2 to each will not affect the asymptotics; the seventh because $n_i^\mu=\Omega\left(\left(n_{i+1}^\mu\right)^\beta\right)$, the eighth because $n_{R-1}\le n_R^\beta$; and the last because $\beta\mu>1/3\ge 1/d$, hence $n_1^{1/d-\beta\mu}<1$. □

Next we need to lower-bound the success probability. Say that $\mathcal{A}_R$ or $\mathcal{U}_R$ "succeeds" if a measurement in the standard basis yields the result $|v(C_{i^*}),1\rangle$, where C_{i^*} is the minimal subcube that contains the marked vertex. Of course, the marked vertex itself can then be found in $n_0 = O(1)$ steps.

Lemma 5.8. *Assuming there is a unique marked vertex, $\mathcal{A}_R$ succeeds with probability $\Omega\left(1/n_R^{1-2\mu}\right)$.*

Proof. Let $P_{\mathcal{A}}(R)$ and $P_{\mathcal{U}}(R)$ be the success probabilities of $\mathcal{A}_R$ and $\mathcal{U}_R$ respectively when searching $\mathcal{L}_d(n_R)$. Then clearly $P_{\mathcal{A}}(0) = 1$, and $P_{\mathcal{U}}(R) = (n_{R-1}/n_R) P_{\mathcal{A}}(R-1)$ for all $R \geq 1$. So by Lemma 5.2,

$$\begin{aligned}
P_{\mathcal{A}}(R) &\geq \left(1 - \frac{1}{3}(2m_R+1)^2 P_{\mathcal{U}}(R)\right)(2m_R+1)^2 P_{\mathcal{U}}(R) \\
&= \left(1 - \frac{1}{3}(2m_R+1)^2 \frac{n_{R-1}}{n_R} P_{\mathcal{A}}(R-1)\right)(2m_R+1)^2 \frac{n_{R-1}}{n_R} P_{\mathcal{A}}(R-1) \\
&\geq \left(1 - \frac{1}{3}(n_R/n_{R-1})^{2\mu} \frac{n_{R-1}}{n_R} P_{\mathcal{A}}(R-1)\right)(n_R/n_{R-1})^{2\mu} \frac{n_{R-1}}{n_R} P_{\mathcal{A}}(R-1) \\
&\geq \left(1 - \frac{1}{3}(n_{R-1}/n_R)^{1-2\mu}\right)(n_{R-1}/n_R)^{1-2\mu} P_{\mathcal{A}}(R-1) \\
&\geq (n_0/n_R)^{1-2\mu} \prod_{r=1}^{R} \left(1 - \frac{1}{3}(n_{R-1}/n_R)^{1-2\mu}\right) \\
&\geq (n_0/n_R)^{1-2\mu} \prod_{r=1}^{R} \left(1 - \frac{1}{3n_R^{(1-\beta)(1-2\mu)}}\right) \\
&\geq (n_0/n_R)^{1-2\mu} \left(1 - \sum_{r=1}^{R} \frac{1}{3n_R^{(1-\beta)(1-2\mu)}}\right) \\
&= \Omega\left(1/n_R^{1-2\mu}\right).
\end{aligned}$$

Here the third line follows because $2m_R + 1 \geq (n_{R-1}/n_R)^{\mu}$ and the function $x - \frac{1}{3}x^2$ is nondecreasing in the interval $[0,1]$; the fourth because $P_{\mathcal{A}}(R-1) \leq 1$; the sixth because $n_{R-1} \leq n_R^{\beta}$; and the last because $\beta < 1$ and $\mu < 1/2$, the n_R's increase doubly exponentially, and n_0 is sufficiently large. □

Finally, take $\mathcal{A}_R$ itself and amplify it to success probability $\Omega(1)$ by running it $O(n_R^{1/2-\mu})$ times. This yields an algorithm for searching $\mathcal{L}_d(n_R)$ with overall running time $O\left(n_R^{1/2}\right)$, which implies that $Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_d(n_R)\right) = O\left(n_R^{1/2}\right)$.

All that remains is to handle values of n that do not equal n_R for any R. The solution is simple: first find the largest R such that $n_R < n$. Then set $n' = n_R \left\lceil n^{1/d}/\ell_R \right\rceil^d$, and embed $\mathcal{L}_d(n)$ into the larger hypercube $\mathcal{L}_d(n')$. Clearly $Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_d(n)\right) \leq Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_d(n')\right)$. Also notice that $n' = O(n)$ and that $n' = O\left(n_R^{1/\beta}\right) = O\left(n_R^{3/2}\right)$. Next partition $\mathcal{L}_d(n')$ into n'/n_R subcubes, each a copy of $\mathcal{L}_d(n_R)$. The algorithm will now have one additional level of recursion, which chooses a subcube of $\mathcal{L}_d(n')$ uniformly

at random, runs $\mathcal{A}_R$ on that subcube, and then amplifies the resulting procedure $\Theta\left(\sqrt{n'/n_R}\right)$ times. The total time is now

$$O\left(\sqrt{\frac{n'}{n_R}}\left(\left(n'\right)^{1/d}+n_R^{1/2}\right)\right)=O\left(\sqrt{\frac{n'}{n_R}}n_R^{1/2}\right)=O\left(\sqrt{n}\right),$$

while the success probability is $\Omega(1)$. This completes Theorem 5.4.

5.3 Dimension 2

In the $d=2$ case, the best we can achieve is the following:

Theorem 5.9. $Q(\mathrm{OR},\mathcal{L}_2)=O\left(\sqrt{n}\log^{5/2}n\right)$.

Again, we start with the single marked vertex case and postpone the general case to Sections 5.4 and 5.5.

Theorem 5.10. $Q\left(\mathrm{OR}^{(1)},\mathcal{L}_2\right)=O\left(\sqrt{n}\log^{3/2}n\right)$.

For $d\geq 3$, we performed amplification on large (greater than $O\left(1/n^{1-2\mu}\right)$) 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 $\Omega(\sqrt{n})$, simply because that is the diameter of the grid. If we want to keep the running time $O(\sqrt{n})$, then we can only perform $O(1)$ 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.

The procedures $\mathcal{A}_R$, $\mathcal{U}_R$, W_R, and S_R are identical to those in Section 5.2; all that changes are the parameter settings. For all integers $R\geq 0$, we now let $n_R=\ell_0^{2R}$, for some odd integer $\ell_0\geq 3$ to be set later. Thus, $\mathcal{A}_R$ and $\mathcal{U}_R$ search the square grid $\mathcal{L}_2(n_R)$ of size $\ell_0^R\times\ell_0^R$. Also, let $m=(\ell_0-1)/2$; then $\mathcal{A}_R$ applies m steps of amplitude amplification to $\mathcal{U}_R$.

We now prove the counterparts of Lemmas 5.7 and 5.8 for the two-dimensional case.

Lemma 5.11. *$\mathcal{A}_R$ uses $O\left(R\ell_0^{R+1}\right)$ steps.*

Proof. Let $T_{\mathcal{A}}(R)$ and $T_{\mathcal{U}}(R)$ be the time used by $\mathcal{A}_R$ and $\mathcal{U}_R$ respectively in searching $\mathcal{L}_2(n_R)$. Then $T_{\mathcal{A}}(0)=1$, and for all $R\geq 1$,

$$\begin{aligned}T_{\mathcal{A}}(R)&\leq(2m+1)\,T_{\mathcal{U}}(R)+2m,\\ T_{\mathcal{U}}(R)&\leq 2n_R^{1/2}+T_{\mathcal{A}}(R-1).\end{aligned}$$

Combining,

$$\begin{aligned}T_{\mathcal{A}}(R)&\leq(2m+1)\left(2n_R^{1/2}+T_{\mathcal{A}}(R-1)\right)+2m\\ &=\ell_0\left(2\ell_0^R+T_{\mathcal{A}}(R-1)\right)+\ell_0-1\\ &=O\left(\ell_0^{R+1}+\ell_0 T_{\mathcal{A}}(R-1)\right)\\ &=O\left(R\ell_0^{R+1}\right).\end{aligned}$$

□

Lemma 5.12. *$\mathcal{A}_R$ succeeds with probability $\Omega(1/R)$.*

Proof. Let $P_{\mathcal{A}}(R)$ and $P_{\mathcal{U}}(R)$ be the success probabilities of $\mathcal{A}_R$ and $\mathcal{U}_R$ respectively when searching $\mathcal{L}_2(n_R)$. Then $P_{\mathcal{U}}(R) = P_{\mathcal{A}}(R-1)/\ell_0^2$ for all $R \geq 1$. So by Lemma 5.2, and using the fact that $2m+1 = \ell_0$,

$$\begin{aligned} P_{\mathcal{A}}(R) &\geq \left(1 - \frac{(2m+1)^2}{3} P_{\mathcal{U}}(R)\right)(2m+1)^2 P_{\mathcal{U}}(R) \\ &= \left(1 - \frac{\ell_0^2}{3}\frac{P_{\mathcal{A}}(R-1)}{\ell_0^2}\right)\ell_0^2 \frac{P_{\mathcal{A}}(R-1)}{\ell_0^2} \\ &= P_{\mathcal{A}}(R-1) - \frac{1}{3}P_{\mathcal{A}}^2(R-1) \\ &= \Omega(1/R). \end{aligned}$$

This is because $\Omega(R)$ iterations of the map $x_R := x_{R-1} - \frac{1}{3}x_{R-1}^2$ are needed to drop from (say) $2/R$ to $1/R$, and $x_0 = P_{\mathcal{A}}(0) = 1$ is greater than $2/R$. □

We can amplify $\mathcal{A}_R$ to success probability $\Omega(1)$ by repeating it $O\left(\sqrt{R}\right)$ times. This yields an algorithm for searching $\mathcal{L}_2(n_R)$ that uses $O\left(R^{3/2}\ell_0^{R+1}\right) = O\left(\sqrt{n_R}R^{3/2}\ell_0\right)$ steps in total. We can minimize this expression subject to $\ell_0^{2R} = n_R$ by taking ℓ_0 to be constant and R to be $\Theta(\log n_R)$, which yields $Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_2(n_R)\right) = O\left(\sqrt{n_R}\log n_R^{3/2}\right)$. If n is not of the form ℓ_0^{2R}, then we simply find the smallest integer R such that $n < \ell_0^{2R}$, and embed $\mathcal{L}_2(n)$ in the larger grid $\mathcal{L}_2\left(\ell_0^{2R}\right)$. Since ℓ_0 is a constant, this increases the running time by at most a constant factor. We have now proved Theorem 5.10.

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. [10]. In our setting, however, this is too much to hope for—since even if there are many marked vertices, 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, we can show a stronger lower bound. Recall that $\mathrm{OR}^{(k)}$ is the problem of deciding whether there are no marked vertices or exactly k of them.

Theorem 5.13. *For all dimensions $d \geq 2$,*

$$Q\left(\mathrm{OR}^{(k)}, \mathcal{L}_d\right) = \Omega\left(\frac{\sqrt{n}}{k^{1/2-1/d}}\right).$$

Here, for simplicity, we ignore constant factors depending on d.

Proof. For simplicity, we assume that both $k^{1/d}$ and $\left(n/3^d k\right)^{1/d}$ are integers. (In the general case, we can just replace k by $\left\lceil k^{1/d}\right\rceil^d$ and n by the largest integer of the form $(3m)^d k$ which is less than n. This only changes the lower bound by a constant factor depending on d.)

We use a hybrid argument almost identical to that of Theorem 4.4. Divide $\mathcal{L}_d$ into n/k subcubes, each having k vertices and side length $k^{1/d}$. Let S be a regularly-spaced set of $M = n/\left(3^d k\right)$ of these subcubes, so that any two subcubes in S have distance at least $2k^{1/d}$ from one another. Then choose a subcube $C_j \in S$ uniformly at random and mark all k vertices in C_j. This enables us to consider each $C_j \in S$ itself as a *single* vertex (out of M in total), having distance at least $2k^{1/d}$ to every other vertex.

More formally, given a subcube $C_j \in S$, let $\widetilde{C}_j$ be the set of vertices consisting of C_j and the $3^d - 1$ subcubes surrounding it. (Thus, $\widetilde{C}_j$ is a subcube of side length $3k^{1/d}$.) Then the query magnitude of $\widetilde{C}_j$ after the t^{th} query is

$$\Gamma_j^{(t)} = \sum_{v_i \in \widetilde{C}_j} \sum_z \left|\alpha_{i,z}^{(t)}(X_0)\right|^2,$$

where X_0 is the all-zero input. Let T be the number of queries, and let $w = T/\left(ck^{1/d}\right)$ for some constant $c > 0$. Then as in Theorem 4.4, there must exist a subcube $\widetilde{C}_{j^*}$ such that

$$\sum_{q=0}^{w-1} \Gamma_{j^*}^{\left(T-qck^{1/d}\right)} \leq \frac{w}{M} = \frac{3^d k w}{n}.$$

Let Y be the input which is 1 in C_{j^*} and 0 elsewhere; then let X_q be a hybrid input which is X_0 during queries 1 to $T - qck^{1/d}$, but Y during queries $T - qck^{1/d} + 1$ to T. Next let

$$D(q,r) = \sum_{v_i \in G} \sum_z \left|\alpha_{i,z}^{(T)}(X_q) - \alpha_{i,z}^{(T)}(X_r)\right|^2.$$

Then as in Theorem 4.4, for all $c < 1$ we have $D(q-1,q) \leq 4\Gamma_{j^*}^{\left(T-qck^{1/d}\right)}$. For in the $ck^{1/d}$ queries from $T - qck^{1/d} + 1$ through $T - (q-1)ck^{1/d}$, no amplitude originating outside $\widetilde{C}_{j^*}$ can travel a distance $k^{1/d}$ and thereby reach C_{j^*}. Therefore switching from X_{q-1} to X_q can only affect amplitude that is in $\widetilde{C}_{j^*}$ immediately after query $T - qck^{1/d}$. It follows that

$$\sqrt{D(0,w)} \leq \sum_{q=1}^{w} \sqrt{D(q-1,q)} \leq 2\sum_{q=1}^{w} \sqrt{\Gamma_{j^*}^{\left(T-qck^{1/d}\right)}} \leq 2w\sqrt{\frac{3^d k}{n}} = \frac{2\sqrt{3^d}k^{1/2-1/d}T}{c\sqrt{n}}.$$

Hence $T = \Omega\left(\sqrt{n}/k^{1/2-1/d}\right)$ for constant d, since assuming the algorithm is correct we need $D(0,w) = \Omega(1)$. □

Notice that if $k \approx n$, then the bound of Theorem 5.13 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 $\Omega(\sqrt{n})$ independent of k. The bound of Theorem 5.13 can be achieved (up to a constant factor that depends on d) for $d \geq 3$, and nearly achieved for $d = 2$. We first construct an algorithm for the case when k is known.

Theorem 5.14.

(i) For $d \geq 3$,

$$Q\left(\mathrm{OR}^{(k)}, \mathcal{L}_d\right) = O\left(\frac{\sqrt{n}}{k^{1/2-1/d}}\right).$$

(ii) For $d = 2$,

$$Q\left(\mathrm{OR}^{(k)}, \mathcal{L}_2\right) = O\left(\sqrt{n}\log^{3/2} n\right).$$

To prove Theorem 5.14, we first divide $\mathcal{L}_d(n)$ into n/γ subcubes, each of size $\gamma^{1/d} \times \cdots \times \gamma^{1/d}$ (where γ will be fixed later). Then in each subcube, we choose one vertex uniformly at random.

Lemma 5.15. *If $\gamma \geq k$, then the probability that exactly one marked vertex is chosen is at least $k/\gamma - (k/\gamma)^2$.*

Proof. Let x be a marked vertex. The probability that x is chosen is $1/\gamma$. Given that x is chosen, the probability that one of the other marked vertices, y, is chosen is 0 if x and y belong to the same subcube, or $1/\gamma$ if they belong to different subcubes. Therefore, the probability that x alone is chosen is at least

$$\frac{1}{\gamma}\left(1 - \frac{k-1}{\gamma}\right) \geq \frac{1}{\gamma}\left(1 - \frac{k}{\gamma}\right).$$

Since the events "x alone is chosen" are mutually disjoint, we conclude that the probability that exactly one marked vertex is chosen is at least $k/\gamma - (k/\gamma)^2$. □

In particular, fix γ so that $\gamma/3 < k < 2\gamma/3$; then Lemma 5.15 implies that the probability of choosing exactly one marked vertex is at least $2/9$. The algorithm is now as follows. As in the lemma, subdivide $\mathcal{L}_d(n)$ into n/γ subcubes and choose one location at random from each. Then run the algorithm for the unique-solution case (Theorem 5.4 or 5.10) on the chosen locations only, as if they were vertices of $\mathcal{L}_d(n/\gamma)$.

The running time in the unique case was $O\left(\sqrt{n/\gamma}\right)$ for $d \geq 3$ or

$$O\left(\sqrt{\frac{n}{\gamma}}\log^{3/2}(n/\gamma)\right) = O\left(\sqrt{\frac{n}{\gamma}}\log^{3/2} n\right)$$

for $d = 2$. However, each local unitary in the original algorithm now becomes a unitary affecting two vertices v and w in neighboring subcubes C_v and C_w. When placed side by side, C_v and C_w form a rectangular box of size $2\gamma^{1/d} \times \gamma^{1/d} \times \cdots \times \gamma^{1/d}$. Therefore the distance between v and w is at most $(d+1)\gamma^{1/d}$. It follows that each local unitary in the original algorithm takes $O\left(d\gamma^{1/d}\right)$ time in the new algorithm. For $d \geq 3$, this results in an overall running time of

$$O\left(\sqrt{\frac{n}{\gamma}}d\gamma^{1/d}\right) = O\left(d\frac{\sqrt{n}}{\gamma^{1/2-1/d}}\right) = O\left(\frac{\sqrt{n}}{k^{1/2-1/d}}\right).$$

For $d = 2$ we obtain

$$O\left(\sqrt{\frac{n}{\gamma}}\gamma^{1/2}\log^{3/2} n\right) = O\left(\sqrt{n}\log^{3/2} n\right).$$

5.5 Unknown Number of Marked Items

We now show how to deal with an unknown k. Let $\mathrm{OR}^{(\geq k)}$ be the problem of deciding whether there are no marked vertices or *at least* k of them, given that one of these is true.

Theorem 5.16.

(i) For $d \geq 3$,

$$Q\left(\mathrm{OR}^{(\geq k)}, \mathcal{L}_d\right) = O\left(\frac{\sqrt{n}}{k^{1/2-1/d}}\right).$$

(ii) For $d = 2$,

$$Q\left(\mathrm{OR}^{(\geq k)}, \mathcal{L}_2\right) = O\left(\sqrt{n}\log^{5/2} n\right).$$

Proof. We use the straightforward 'doubling' approach of Boyer et al. [10]:

(1) For $j = 0$ to $\log_2(n/k)$

- Run the algorithm of Theorem 5.14 with subcubes of size $\gamma_j = 2^j k$.
- If a marked vertex is found, then output 1 and halt.

(2) Query a random vertex v, and output 1 if v is a marked vertex and 0 otherwise.

Let $k^* \geq k$ be the number of marked vertices. If $k^* \leq n/3$, then there exists a $j \leq \log_2(n/k)$ such that $\gamma_j/3 \leq k^* \leq 2\gamma_j/3$. So Lemma 5.15 implies that the j^{th} iteration of step (1) finds a marked vertex with probability at least $2/9$. On the other hand, if $k^* \geq n/3$, then step (2) finds a marked vertex with probability at least $1/3$. For $d \geq 3$, the time used in step (1) is at most

$$\sum_{j=0}^{\log_2(n/k)} \frac{\sqrt{n}}{\gamma_j^{1/2-1/d}} = \frac{\sqrt{n}}{k^{1/2-1/d}} \left[\sum_{j=0}^{\log_2(n/k)} \frac{1}{2^{j(1/2-1/d)}}\right] = O\left(\frac{\sqrt{n}}{k^{1/2-1/d}}\right),$$

the sum in brackets being a decreasing geometric series. For $d = 2$, the time is $O\left(\sqrt{n}\log^{5/2} n\right)$, since each iteration takes $O\left(\sqrt{n}\log^{3/2} n\right)$ time and there are at most $\log n$ iterations. In neither case does step (2) affect the bound, since $k \leq n$ implies that $n^{1/d} \leq \sqrt{n}/k^{1/2-1/d}$. □

Taking $k = 1$ gives algorithms for unconstrained OR with running times $O(\sqrt{n})$ for $d \geq 3$ and $O(\sqrt{n}\log^{5/2} n)$ for $d = 2$, thereby establishing Theorems 5.3 and 5.9.

6 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 $\kappa > 0$ such that for all n, ℓ and $v \in V_n$,

$$|B(v,\ell)| \geq \min\left(\kappa \ell^d, n\right),$$

where $B(v,\ell)$ is the set of vertices having distance at most ℓ 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 $\mathcal{L}_d(n)$.[5] It is immediate that the diameter of G_n is at most $(n/\kappa)^{1/d}$. 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.

Our goal is to show the following.

Theorem 6.1. *If G is d-dimensional, then*

(i) For a constant $d > 2$,

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

(ii) For $d = 2$,

$$Q(\mathrm{OR}, G) = \sqrt{n} 2^{O\left(\sqrt{\log n}\right)}.$$

In proving part (i), the intuition is simple: we want to decompose G recursively into subgraphs (called *clusters*), which will serve the same role as subcubes did in the hypercube case. The procedure is as follows. For some constant $n_1 > 1$, first choose $\lceil n/n_1 \rceil$ vertices uniformly at random to be designated as 1-*pegs*. Then form 1-*clusters* by assigning each vertex in G to its closest 1-peg, as in a Voronoi diagram. (Ties are broken randomly.) Let $v(C)$ be the peg of cluster C. Next, split up any 1-cluster C with more than n_1 vertices into $\lceil |C|/n_1 \rceil$ arbitrarily-chosen 1-clusters, each with size at most n_1 and with $v(C)$ as its 1-peg. Observe that

$$\sum_{i=1}^{\lceil n/n_1 \rceil} \left\lceil \frac{|C_i|}{n_1} \right\rceil \leq 2 \left\lceil \frac{n}{n_1} \right\rceil,$$

where $n = |C_1| + \cdots + \left|C_{\lceil n/n_1 \rceil}\right|$. Therefore, the splitting-up step can at most double the number of clusters.

In the next iteration, set $n_2 = n_1^{1/\beta}$, for some constant $\beta \in (2/d, 1)$. Choose $2\lceil n/n_2 \rceil$ vertices uniformly at random as 2-pegs. Then form 2-clusters by assigning each 1-cluster C to the 2-peg that is closest to the 1-peg $v(C)$. Given a 2-cluster C', let $|C'|$ be the number of 1-clusters in C'. Then as before, split up any C' with $|C'| > n_2/n_1$ into $\lceil |C'|/(n_2/n_1) \rceil$ arbitrarily-chosen 2-clusters, each with size at most n_2/n_1 and with $v(C')$ as its 2-peg. Continue recursively in this manner, setting $n_R = n_{R-1}^{1/\beta}$

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

and choosing $2^{R-1}\lceil n/n_R\rceil$ vertices as R-pegs for each R. Stop at the maximum R such that $n_R \leq n$. For technical convenience, set $n_0 = 1$, and consider each vertex v to be the 0-peg of the 0-cluster $\{v\}$.

For $R \geq 1$, define the *radius* of an R-cluster C to be the maximum, over all $(R-1)$-clusters C' in C, of the distance from $v(C)$ to $v(C')$. Also, call an R-cluster *good* if it has radius at most ℓ_R, where $\ell_R = \left(\frac{2}{\kappa} n_R \ln n\right)^{1/d}$.

Lemma 6.2. *With probability $1-o(1)$ over the choice of clusters, all clusters are good.*

Proof. Let v be the $(R-1)$-peg of an $(R-1)$-cluster. Then $|B(v,\ell)| \geq \kappa \ell^d$, where $B(v,\ell)$ is the ball of radius ℓ about v. So the probability that v has distance greater than ℓ_R to the nearest R-peg is at most

$$\left(1 - \frac{\kappa \ell_R^d}{n}\right)^{\lceil n/n_R \rceil} \leq \left(1 - \frac{2\ln n}{n/n_R}\right)^{n/n_R} < \frac{1}{n^2}.$$

Furthermore, the total number of pegs is easily seen to be $O(n)$. It follows by the union bound that *every* $(R-1)$-peg for *every* R has distance at most ℓ_R to the nearest R-peg, with probability $1 - O(1/n) = 1 - o(1)$ over the choice of clusters. □

At the end we have a tree of clusters, which can be searched recursively just as in the hypercube case. Lemma 6.2 gives us a guarantee on the time needed to move a level down (from a peg of an R-cluster to a peg of an $R-1$-cluster contained in it) or a level up. Also, let $K'(C)$ be the number of $(R-1)$-clusters in R-cluster C; then $K'(C) \leq K(R)$ where $K(R) = 2\lceil n_R/n_{R-1}\rceil$. If $K'(C) < K(R)$, then place $K(R) - K'(C)$ "dummy" $(R-1)$-clusters in C, each of which has $(R-1)$-peg $v(C)$. Now, every R-cluster contains an equal number of $R-1$ clusters.

Our algorithm is similar to Section 5.2 but the basis states now have the form $|v,z,C\rangle$, where v is a vertex, z is an answer bit, and C is the label of the cluster currently being searched. (Unfortunately, because multiple R-clusters can have the same peg, a single auxiliary qubit no longer suffices.)

The algorithm $\mathcal{A}_R$ from Section 5.2 now does the following, when invoked on the initial state $|v(C),0,C\rangle$, where C is an R-cluster. If $R = 0$, then $\mathcal{A}_R$ uses a query transformation to prepare the state $|v(C),1,C\rangle$ if $v(C)$ is the marked vertex and $|v(C),0,C\rangle$ otherwise. If $R \geq 1$ and C is not a dummy cluster, then $\mathcal{A}_R$ performs m_R steps of amplitude amplification on $\mathcal{U}_R$, where m_R is the largest integer such that $2m_R + 1 \leq \sqrt{n_R/n_{R-1}}$.[6] If C is a dummy cluster, then $\mathcal{A}_R$ does nothing for an appropriate number of steps, and then returns that no marked item was found.

We now describe the subroutine $\mathcal{U}_R$, for $R \geq 1$. When invoked with $|v(C),0,C\rangle$ as its initial state, $\mathcal{U}_R$ first prepares a uniform superposition

$$|\phi_C\rangle = \frac{1}{\sqrt{K(R)}} \sum_{i=1}^{K(R)} |v(C_i),0,C_i\rangle.$$

It does this by first constructing a spanning tree T for C, rooted at $v(C)$ and having minimal depth, and then moving amplitude along the edges of T so as to prepare $|\phi_C\rangle$. After $|\phi_C\rangle$ has been prepared, $\mathcal{U}_R$ then calls $\mathcal{A}_{R-1}$ recursively, to search $C_1,\ldots,C_{K(R)}$ in superposition and amplify the results. Note that,

[6] In the hypercube case, we performed fewer amplifications in order to lower the running time from $\sqrt{n}\,\mathrm{polylog}\,n$ to $\sqrt{n}$. Here, though, the splitting-up step produces a $\mathrm{polylog}\,n$ factor anyway.

because of the cluster labels, there is no reason why amplitude being routed through C should not pass through some other cluster C' along the way—but there is also no advantage in our analysis for allowing this.

We now analyze the running time and success probability of $\mathcal{A}_R$.

Lemma 6.3. *$\mathcal{A}_R$ uses $O\left(\sqrt{n_R}\log^{1/d} n\right)$ steps, assuming that all clusters are good.*

Proof. Let $T_{\mathcal{A}}(R)$ and $T_{\mathcal{U}}(R)$ be the time used by $\mathcal{A}_R$ and $\mathcal{U}_R$ respectively in searching an R-cluster. Then we have

$$\begin{aligned} T_{\mathcal{A}}(R) &\leq \sqrt{n_R/n_{R-1}}\,T_{\mathcal{U}}(R), \\ T_{\mathcal{U}}(R) &\leq \ell_R + T_{\mathcal{A}}(R-1) \end{aligned}$$

with the base case $T_{\mathcal{A}}(0) = 1$. Combining,

$$\begin{aligned} T_{\mathcal{A}}(R) &\leq \sqrt{n_R/n_{R-1}}\,(\ell_R + T_{\mathcal{A}}(R-1)) \\ &\leq \sqrt{n_R/n_{R-1}}\ell_R + \sqrt{n_R/n_{R-2}}\ell_{R-1} + \cdots + \sqrt{n_R/n_0}\ell_1 \\ &= \sqrt{n_R}\cdot O\left(\frac{(n_R \ln n)^{1/d}}{\sqrt{n_{R-1}}} + \cdots + \frac{(n_1 \ln n)^{1/d}}{\sqrt{n_0}}\right) \\ &= \sqrt{n_R}\left(\ln^{1/d} n\right)\cdot O\left(n_R^{1/d-\beta/2} + \cdots + n_1^{1/d-\beta/2}\right) \\ &= \sqrt{n_R}\left(\ln^{1/d} n\right)\cdot O\left(n_1^{1/d-\beta/2} + \left(n_1^{1/d-\beta/2}\right)^{1/\beta} + \cdots + \left(n_1^{1/d-\beta/2}\right)^{(1/\beta)^{R-1}}\right) \\ &= O\left(\sqrt{n_R}\log^{1/d} n\right), \end{aligned}$$

where the last line holds because $\beta > 2/d$ and therefore $n_1^{1/d-\beta/2} < 1$. □

Lemma 6.4. *$\mathcal{A}_R$ succeeds with probability $\Omega(1/\operatorname{polylog} n_R)$ in searching a graph of size $n = n_R$, assuming there is a unique marked vertex.*

Proof. For all $R \geq 0$, let C_R be the R-cluster that contains the marked vertex, and let $P_{\mathcal{A}}(R)$ and $P_{\mathcal{U}}(R)$ be the success probabilities of $\mathcal{A}_R$ and $\mathcal{U}_R$ respectively when searching C_R. Then for all $R \geq 1$, we have $P_{\mathcal{U}}(R) = P_{\mathcal{A}}(R-1)/K(R)$, and therefore

$$\begin{aligned} P_{\mathcal{A}}(R) &\geq \left(1 - \frac{(2m_R+1)^2}{3}P_{\mathcal{U}}(R)\right)(2m_R+1)^2 P_{\mathcal{U}}(R) \\ &= \left(1 - \frac{(2m_R+1)^2}{3}\cdot\frac{P_{\mathcal{A}}(R-1)}{K(R)}\right)(2m_R+1)^2\frac{P_{\mathcal{A}}(R-1)}{K(R)} \\ &= \Omega\left(P_{\mathcal{A}}(R-1)\right) \\ &= \Omega\left(1/\operatorname{polylog} n_R\right). \end{aligned}$$

Here the third line holds because $(2m_R+1)^2 \approx n_R/n_{R-1} \approx K(R)/2$, and the last line because $R = \Theta(\log\log n_R)$. □

Finally, we repeat $\mathcal{A}_R$ itself $O(\text{polylog}\, n_R)$ times, to achieve success probability $\Omega(1)$ using $O\left(\sqrt{n_R}\,\text{polylog}\, n_R\right)$ steps in total. Again, if n is not equal to n_R for any R, then we simply find the largest R such that $n_R < n$, and then add one more level of recursion that searches a random R-cluster and amplifies the result $\Theta\left(\sqrt{n/n_R}\right)$ times. The resulting algorithm uses $O\left(\sqrt{n}\,\text{polylog}\, n\right)$ steps, thereby establishing part (i) of Theorem 6.1 for the case of a unique marked vertex. The generalization to multiple marked vertices is straightforward.

Corollary 6.5. *If G is d-dimensional for a constant $d > 2$, then*

$$Q\left(\text{OR}^{(\geq k)}, G\right) = O\left(\frac{\sqrt{n}\,\text{polylog}\,\frac{n}{k}}{k^{1/2-1/d}}\right).$$

Proof. Assume without loss of generality that $k = o(n)$, since otherwise a marked item is trivially found in $O\left(n^{1/d}\right)$ steps. As in Theorem 5.16, we give an algorithm $\mathcal{B}$ consisting of $\log_2(n/k)+1$ iterations. In iteration $j = 0$, choose $\lceil n/k \rceil$ vertices $w_1, \ldots, w_{\lceil n/k \rceil}$ uniformly at random. Then run the algorithm for the unique marked vertex case, but instead of taking all vertices in G as 0-pegs, take only $w_1, \ldots, w_{\lceil n/k \rceil}$. On the other hand, still choose the 1-pegs, 2-pegs, and so on uniformly at random from among all vertices in G. For all R, the number of R-pegs should be $\lceil (n/k)/n_R \rceil$. In general, in iteration j of $\mathcal{B}$, choose $\left\lceil n/\left(2^j k\right)\right\rceil$ vertices $w_1, \ldots, w_{\lceil n/(2^j k) \rceil}$ uniformly at random, and then run the algorithm for a unique marked vertex as if $w_1, \ldots, w_{\lceil n/(2^j k) \rceil}$ were the only vertices in the graph.

It is easy to see that, assuming there are k or more marked vertices, with probability $\Omega(1)$ there exists an iteration j such that exactly one of $w_1, \ldots, w_{\lceil n/(2^j k) \rceil}$ is marked. Hence $\mathcal{B}$ succeeds with probability $\Omega(1)$. It remains only to upper-bound $\mathcal{B}$'s running time.

In iteration j, notice that Lemma 6.2 goes through if we use $\ell_R^{(j)} := \left(\frac{2}{\kappa} 2^j k n_R \ln \frac{n}{k}\right)^{1/d}$ instead of ℓ_R. That is, with probability $1 - O(k/n) = 1 - o(1)$ over the choice of clusters, every R-cluster has radius at most $\ell_R^{(j)}$. So letting $T_{\mathcal{A}}(R)$ be the running time of $\mathcal{A}_R$ on an R-cluster, the recurrence in Lemma 6.3 becomes

$$T_{\mathcal{A}}(R) \leq \sqrt{n_R/n_{R-1}}\left(\ell_R^{(j)} + T_{\mathcal{A}}(R-1)\right) = O\left(\sqrt{n_R}\left(2^j k \log(n/k)\right)^{1/d}\right),$$

which is

$$O\left(\frac{\sqrt{n}\log^{1/d}\frac{n}{k}}{(2^j k)^{1/2-1/d}}\right)$$

if $n_R = \Theta\left(n/\left(2^j k\right)\right)$. As usual, the case where there is no R such that $n_R = \Theta\left(n/\left(2^j k\right)\right)$ is trivially handled by adding one more level of recursion. If we factor in the $O(1/\text{polylog}\, n_R)$ repetitions of $\mathcal{A}_R$ needed to boost the success probability to $\Omega(1)$, then the total running time of iteration j is

$$O\left(\frac{\sqrt{n}\,\text{polylog}\,\frac{n}{k}}{(2^j k)^{1/2-1/d}}\right).$$

Therefore $\mathcal{B}$'s running time is

$$O\left(\sum_{j=0}^{\log_2(n/k)} \frac{\sqrt{n}\,\text{polylog}\, n}{(2^j k)^{1/2-1/d}}\right) = O\left(\frac{\sqrt{n}\,\text{polylog}\, n}{k^{1/2-1/d}}\right).$$

□

For the $d = 2$ case, the best upper bound we can show is $\sqrt{n}2^{O(\sqrt{\log n})}$. This is obtained by simply modifying $\mathcal{A}_R$ to have a deeper recursion tree. Instead of taking $n_R = n_{R-1}^{1/\mu}$ for some μ, we take $n_R = 2^{\sqrt{\log n}} n_{R-1} = 2^{R\sqrt{\log n}}$, so that the total number of levels is $\lceil \sqrt{\log n} \rceil$. Lemma 6.2 goes through without modification, while the recurrence for the running time becomes

$$\begin{aligned} T_{\mathcal{A}}(R) &\leq \sqrt{n_R/n_{R-1}}\,(\ell_R + T_{\mathcal{A}}(R-1)) \\ &\leq \sqrt{n_R/n_{R-1}}\,\ell_R + \sqrt{n_R/n_{R-2}}\,\ell_{R-1} + \cdots + \sqrt{n_R/n_0}\,\ell_1 \\ &= O\left(2^{\sqrt{\log n}(R/2)}\sqrt{\ln n} + \cdots + 2^{\sqrt{\log n}(R/2)}\sqrt{\ln n}\right) \\ &= \sqrt{n}2^{O(\sqrt{\log n})}. \end{aligned}$$

Also, since the success probability decreases by at most a constant factor at each level, we have that $P_{\mathcal{A}}(R) = 2^{-O(\sqrt{\log n})}$, and hence $2^{O(\sqrt{\log n})}$ amplification steps suffice to boost the success probability to $\Omega(1)$. Handling multiple marked items adds an additional factor of $\log n$, which is absorbed into $2^{O(\sqrt{\log n})}$. This completes Theorem 6.1.

6.1 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* be marked, and we know which h those are. Then how much time is needed to determine whether any of the h *is* marked? 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 $O(\sqrt{n}\,\mathrm{polylog}\,n)$ steps suffice. However, we wish to use fewer steps, taking advantage of the fact that h might be much smaller than n. Formally, suppose we are given a graph G with n vertices, of which h are potentially marked. Let $\mathrm{OR}^{(h,\geq k)}$ be the problem of deciding whether G has no marked vertices or at least k of them, given that one of these is the case.

Proposition 6.6. *For all integer constants $d \geq 2$, there exists a d-dimensional graph G such that*

$$Q\left(\mathrm{OR}^{(h,\geq k)}, G\right) = \Omega\left(n^{1/d}\left(\frac{h}{k}\right)^{1/2-1/d}\right).$$

Proof. Let G be the d-dimensional hypercube $\mathcal{L}_d(n)$. Create h/k subcubes of potentially marked vertices, each having k vertices and side length $k^{1/d}$. Space these subcubes out in $\mathcal{L}_d(n)$ so that the distance between any pair of them is $\Omega\left((nk/h)^{1/d}\right)$. Then choose a subcube C uniformly at random and mark all k vertices in C. This enables us to consider each subcube as a single vertex, having distance $\Omega\left((nk/h)^{1/d}\right)$ to every other vertex. The lower bound now follows by a hybrid argument essentially identical to that of Theorem 5.13. □

In particular, if $d = 2$ then $\Omega(\sqrt{n})$ time is always needed, since the potentially marked vertices might all be far from the start vertex. The lower bound of Proposition 6.6 can be achieved up to a polylogarithmic factor.

Proposition 6.7. *If G is d-dimensional for a constant $d > 2$, then*

$$Q\left(\mathrm{OR}^{(h,\geq k)}, G\right) = O\left(n^{1/d}\left(\frac{h}{k}\right)^{1/2-1/d} \mathrm{polylog}\,\frac{h}{k}\right).$$

Proof. Assume without loss of generality that $k = o(h)$, since otherwise a marked item is trivially found. Use algorithm $\mathcal{B}$ from Corollary 6.5, with the following simple change. In iteration j, choose $\lceil h/\left(2^j k\right)\rceil$ potentially marked vertices $w_1, \ldots, w_{\lceil h/(2^j k)\rceil}$ uniformly at random, and then run the algorithm for a unique marked vertex as if $w_1, \ldots, w_{\lceil h/(2^j k)\rceil}$ were the only vertices in the graph. That is, take $w_1, \ldots, w_{\lceil h/(2^j k)\rceil}$ as 0-pegs; then for all $R \geq 1$, choose $\lceil h/\left(2^j k n_R\right)\rceil$ vertices of G uniformly at random as R-pegs. Lemma 6.2 goes through if we use $\widehat{\ell}_R^{(j)} := \left(\frac{2}{\kappa}\frac{n}{h} 2^j k n_R \ln\frac{h}{k}\right)^{1/d}$ instead of ℓ_R. So following Corollary 6.5, the running time of iteration j is now

$$O\left(\sqrt{n_R}\left(\frac{n}{h}2^j k\right)^{1/d} \mathrm{polylog}\,\frac{h}{k}\right) = O\left(n^{1/d}\left(\frac{h}{2^j k}\right)^{1/2-1/d} \mathrm{polylog}\,\frac{h}{k}\right)$$

if $n_R = \Theta\left(h/\left(2^j k\right)\right)$. Therefore the total running time is

$$O\left(\sum_{j=0}^{\log_2(h/k)} n^{1/d}\left(\frac{h}{2^j k}\right)^{1/2-1/d} \mathrm{polylog}\,\frac{h}{k}\right) = O\left(n^{1/d}\left(\frac{h}{k}\right)^{1/2-1/d} \mathrm{polylog}\,\frac{h}{k}\right).$$

□

Intuitively, Proposition 6.7 says that the worst case for search occurs when the h potential marked vertices are scattered evenly throughout the graph.

7 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 [12] observed that in the quantum setting, Alice and Bob can solve this problem using only $O(\sqrt{n}\log n)$ qubits of communication. This was subsequently improved by Høyer and de Wolf [20] 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 5.3, we can improve this to $O(\sqrt{n})$, which matches the celebrated $\Omega(\sqrt{n})$ lower bound of Razborov [23].

Theorem 7.1. *The quantum communication complexity of the disjointness problem is $O(\sqrt{n})$.*

Proof. The protocol is as follows. Alice and Bob both store their inputs in a 3-D cube $\mathcal{L}_3(n)$ (Figure 3); 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 $j,k,l \in \left\{0,\ldots,n^{1/3}-1\right\}$. To

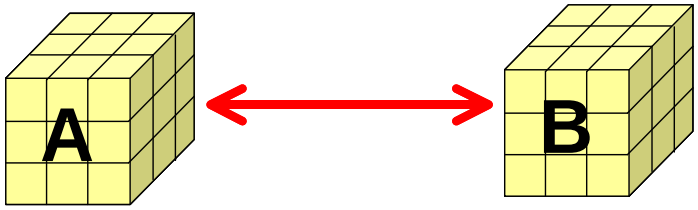

Figure 3: Alice and Bob 'synchronize' locations on their respective cubes.

decide whether there exists a (j,k,l) with $x_{jkl} = y_{jkl} = 1$, Alice simply runs our search algorithm for an unknown number of marked items. If the search algorithm is in the state

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

then the joint state of Alice and Bob will be

$$\sum \alpha_{j,k,l,z,c} \left|v_{jkl}\right\rangle \otimes \left|z\right\rangle \otimes \left|c\right\rangle \otimes \left|v_{jkl}\right\rangle, \tag{7.1}$$

where Alice holds the first $\left|v_{jkl}\right\rangle$ and $|z\rangle$, Bob holds the second $\left|v_{jkl}\right\rangle$, and $|c\rangle$ is the communication channel. Thus, whenever Alice is at location (j,k,l) of her cube, Bob is at location (j,k,l) of his cube.

(1) To simulate a query, Alice sends $|z\rangle$ and an auxiliary qubit holding x_{jkl} to Bob. Bob performs $|z\rangle \to |z \oplus y_{jkl}\rangle$, conditional on $x_{jkl} = 1$. He then returns both bits to Alice, and finally Alice returns the auxiliary qubit to the $|0\rangle$ state by exclusive-OR'ing it with x_{jkl}.

(2) To simulate a non-query transformation that does not change $\left|v_{jkl}\right\rangle$, Alice just performs it herself.

(3) By examining Algorithms 5.5 and 5.6, we see that there are two transformations that change $\left|v_{jkl}\right\rangle$. We deal with them separately.

First, step 1 of Algorithm 5.5 uses a classical C-local transformation $\left|v_{j,k,l}\right\rangle \to |v_{j',k',l'}\rangle$. This transformation can be simulated by Alice and Bob each separately applying $|v_{j,k,l}\rangle \to |v_{j',k',l'}\rangle$.

Second, step 2 of Algorithm 5.6 applies transformations Z_1, Z_2, and Z_3. For brevity, we restrict ourselves to discussing Z_1. This transformation maps an initial state $\left|v_{j,k,l}, 0\right\rangle$ to a uniform superposition over $|v_{j',k,l}, 0\rangle$ for all (j',k,l) lying in the same C_i as (j,k,l). We can decompose this into a sequence of transformations mapping $|v_{j',k,l}\rangle$ to $\alpha|v_{j',k,l}\rangle + \beta|v_{j'+1,k,l}\rangle$ for some α, β. This can be implemented in three steps, using an auxiliary qubit. The auxiliary qubit is initialized to $|0\rangle$ and is initially held by Alice. At the end, the auxiliary qubit is returned to $|0\rangle$. The sequence of transformations is

$$\begin{aligned} |v_{j',k,l}\rangle\,|0\rangle\,|v_{j',k,l}\rangle &\to \alpha|v_{j',k,l}\rangle\,|0\rangle\,|v_{j',k,l}\rangle + \beta|v_{j',k,l}\rangle\,|1\rangle\,|v_{j',k,l}\rangle \\ &\to \alpha|v_{j',k,l}\rangle\,|0\rangle\,|v_{j',k,l}\rangle + \beta|v_{j',k,l}\rangle\,|1\rangle\,|v_{j'+1,k,l}\rangle \\ &\to \alpha|v_{j',k,l}\rangle\,|0\rangle\,|v_{j',k,l}\rangle\beta|v_{j',k,l}\rangle\,|0\rangle\,|v_{j'+1,k,l}\rangle. \end{aligned}$$

The first transformation is performed by Alice who then sends the auxiliary qubit to Bob. The second transformation is performed by Bob, who then sends the auxiliary qubit back to Alice, who performs the third transformation.

Since the algorithm uses $O(\sqrt{n})$ steps, and each step is simulated using a constant amount of communication, the number of qubits communicated in the disjointness protocol is therefore also $O(\sqrt{n})$. □

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. In particular, can any Z-local or H-local unitary be approximated by a product of a small number of C-local unitaries? Also, is it true that $Q(f,G) = \Theta\left(Q^Z(f,G)\right) = \Theta\left(Q^H(f,G)\right)$ for all f,G?

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 $Q(f,G) = O\left(n^{3/4}\right)$, and we conjecture that this is optimal. However, we were unable to show any lower bound better than $\Omega(\sqrt{n})$.

Finally, what is the complexity of finding a unique marked vertex on a 2-D square grid? As mentioned in Section 1.2, Ambainis, Kempe, and Rivosh [3] showed that $Q\left(\mathrm{OR}^{(1)}, \mathcal{L}_2\right) = O(\sqrt{n}\log n)$. Can the remaining factor of $\log n$ be removed?

9 Acknowledgments

We thank Andrew Childs, Julia Kempe, Neil Shenvi, and Ronald de Wolf for helpful conversations; Jakob Bekenstein, Raphael Bousso, and John Preskill for discussions relevant to Section 2; and the anonymous reviewers for comments on the manuscript and a simpler proof of Lemma 5.15.

References

[1] * D. AHARONOV, A. AMBAINIS, J. KEMPE, AND U. VAZIRANI: Quantum walks on graphs. In *Proc. ACM STOC*, pp. 50–59, 2001. [STOC:380752.380758, arXiv:quant-ph/0012090]. 1.1, 1.2, 3

[2] * A. AMBAINIS: Quantum lower bounds by quantum arguments. *J. Comput. Sys. Sci.*, 64:750–767, 2002. [JCSS:10.1006/jcss.2002.1826, STOC:335305.335394, arXiv:quant-ph/0002066]. 3

[3] * A. AMBAINIS, J. KEMPE, AND A. RIVOSH: Coins make quantum walks faster. In *Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2005. To appear. [arXiv:quant-ph/0402107]. 1.2, 2, 8

[4] * R. BEALS, H. BUHRMAN, R. CLEVE, M. MOSCA, AND R. DE WOLF: Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. Earlier version in IEEE FOCS 1998. [JACM:502090.502097, FOCS:10.1109/SFCS.1998.743485, arXiv:quant-ph/9802049]. 1.1, 3, 4

[5] * J. D. BEKENSTEIN: A universal upper bound on the entropy to energy ratio for bounded systems. *Phys. Rev. D*, 23(2):287–298, 1981. [PRD:10.1103/PhysRevD.23.287]. 2

[6] * P. BENIOFF: Space searches with a quantum robot. In S. J. Lomonaco and H. E. Brandt, editors, *Quantum Computation and Information*, Contemporary Mathematics Series. AMS, 2002. [arXiv:quant-ph/0003006]. 1.2, 2

[7] * C. BENNETT, E. BERNSTEIN, G. BRASSARD, AND U. VAZIRANI: Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997. [SICOMP:30093, arXiv:quant-ph/9701001]. 1.1, 3, 4

[8] * R. BOUSSO: Positive vacuum energy and the N-bound. *J. High Energy Phys.*, 0011(038), 2000. [arXiv:hep-th/0010252]. 2

[9] * R. BOUSSO: The holographic principle. *Reviews of Modern Physics*, 74(3), 2002. [arXiv:hep-th/0203101]. 1, 2

[10] * M. BOYER, G. BRASSARD, P. HØYER, AND A. TAPP: Tight bounds on quantum searching. *Fortschritte Der Physik*, 46(4-5):493–505, 1998. [arXiv:quant-ph/9605034]. 5.4, 5.5

[11] * G. BRASSARD, P. HØYER, M. MOSCA, AND A. TAPP: Quantum amplitude amplification and estimation. In S. J. Lomonaco and H. E. Brandt, editors, *Quantum Computation and Information*, Contemporary Mathematics Series. AMS, 2002. [arXiv:quant-ph/0005055]. 1.1, 5, 5.1, 5.1

[12] * H. BUHRMAN, R. CLEVE, AND A. WIGDERSON: Quantum vs. classical communication and computation. In *Proc. ACM STOC*, pp. 63–68, 1998. [STOC:276698.276713, arXiv:quant-ph/9702040]. 7

[13] * A. M. CHILDS, R. CLEVE, E. DEOTTO, E. FARHI, S. GUTMANN, AND D. A. SPIELMAN: Exponential algorithmic speedup by quantum walk. In *Proc. ACM STOC*, pp. 59–68, 2003. [STOC:780542.780552, arXiv:quant-ph/0209131]. 1.2

[14] * A. M. CHILDS, E. FARHI, AND S. GUTMANN: An example of the difference between quantum and classical random walks. *Quantum Information and Computation*, 1(1-2):35–43, 2002. [arXiv:quant-ph/0103020]. 1.2

[15] * A. M. CHILDS AND J. GOLDSTONE: Spatial search and the Dirac equation. *Phys. Rev. A*, 70(042312), 2004. [PRA:10.1103/PhysRevA.70.042312, arXiv:quant-ph/0405120]. 1.2, 2

[16] * A. M. CHILDS AND J. GOLDSTONE: Spatial search by quantum walk. *Phys. Rev. A*, 70(022314), 2004. [PRA:10.1103/PhysRevA.70.022314, arXiv:quant-ph/0306054]. 1.2, 2, 1.2

[17] * L. K. GROVER: A fast quantum mechanical algorithm for database search. In *Proc. ACM STOC*, pp. 212–219, 1996. [STOC:237814.237866, arXiv:quant-ph/9605043]. 1, 3

[18] * L. K. GROVER: Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79(2):325–328, 1997. [PRL:10.1103/PhysRevLett.79.325, arXiv:quant-ph/9706033]. 1

[19] * L. K. GROVER: A framework for fast quantum mechanical algorithms. In *Proc. ACM STOC*, pp. 53–62, 1998. [STOC:276698.276712, arXiv:quant-ph/9711043]. 1.1, 5, 5.1

[20] * P. HØYER AND R. DE WOLF: Improved quantum communication complexity bounds for disjointness and equality. In *Proc. Intl. Symp. on Theoretical Aspects of Computer Science (STACS)*, pp. 299–310, 2002. [STACS:c22a2t8cewg784jh, arXiv:quant-ph/0109068]. 1.1, 7

[21] * S. LLOYD: Computational capacity of the universe. *Phys. Rev. Lett.*, 88, 2002. [PRL:10.1103/PhysRevLett.88.237901, arXiv:quant-ph/0110141]. 3

[22] * M. NIELSEN AND I. CHUANG: *Quantum Computation and Quantum Information*. Cambridge University Press, 2000. 3.1

[23] * A. A. RAZBOROV: Quantum communication complexity of symmetric predicates. *Izvestiya Math. (English version)*, 67(1):145–159, 2003. [arXiv:quant-ph/0204025]. 1.1, 7

[24] * T. RUDOLPH AND L. GROVER: Quantum searching a classical database (or how we learned to stop worrying and love the bomb). 2002. [arXiv:quant-ph/0206066]. 2

[25] * B. S. RYDEN: *Introduction to Cosmology*. Addison-Wesley, 2002. 3

[26] * 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. [arXiv:astro-ph/9812133]. 2, 3

[27] * A. SAHAI AND S. VADHAN: A complete promise problem for statistical zero-knowledge. *J. ACM*, 50(2):196–249, 2003. Earlier version in IEEE FOCS 1997. [JACM:636865.636868, FOCS:10.1109/SFCS.1997.646133, ECCC:TR00-084]. 2

[28] * N. SHENVI, J. KEMPE, AND K. B. WHALEY: A quantum random walk search algorithm. *Phys. Rev. A*, 67(5), 2003. [PRA:10.1103/PhysRevA.67.052307, arXiv:quant-ph/0210064]. 1.2

[29] * P. SHOR: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Earlier version in IEEE FOCS 1994. [SICOMP:29317, FOCS:10.1109/SFCS.1994.365700, arXiv:quant-ph/9508027]. 1, 3

[30] * V. STRASSEN: Gaussian elimination is not optimal. *Numerische Mathematik*, 14(13):354–356, 1969. 1.2

[31] * J. WATROUS: On one-dimensional quantum cellular automata. In *Proc. IEEE FOCS*, pp. 528–537, 1995. [FOCS:10.1109/SFCS.1995.492583]. 1.1

[32] * CH. ZALKA: Could Grover's algorithm help in searching an actual database? 1999. [arXiv:quant-ph/9901068]. 2

AUTHORS

Scott Aaronson
Postdoctoral Fellow
Institute for Quantum Computing
University of Waterloo
aaronson@ias.edu
http://www.scottaaronson.com

Andris Ambainis
Assistant Professor
Department of Combinatorics and Optimization
University of Waterloo
ambainis@math.uwaterloo.ca
http://www.math.uwaterloo.ca/~ambainis

ABOUT THE AUTHORS

SCOTT AARONSON works on computational complexity, quantum computing, and the foundations of quantum mechanics. He holds a Ph. D. in computer science from UC Berkeley (supervised by Umesh Vazirani), an undergraduate degree from Cornell University, and a New York State GED. He was born in Philadelphia.

ANDRIS AMBAINIS was born in Daugavpils, Latvia. After undergraduate studies at the University of Latvia, he received his Ph. D. from the University of California, Berkeley in 2001, supervised by Umesh Vazirani. Andris joined the University of Waterloo in August 2004. His research interests include quantum algorithms, quantum complexity theory, quantum cryptography as well as the classical theory of computation.