Lower Bounds for Local Search by Quantum Arguments

Scott Aaronson*

ABSTRACT

The problem of finding a local minimum of a black-box function is central for understanding local search as well as quantum adiabatic algorithms. For functions on the Boolean hypercube $\{0,1\}^n$, we show a lower bound of $\Omega\left(2^{n/4}/n\right)$ on the number of queries needed by a quantum computer to solve this problem. More surprisingly, our approach, based on Ambainis's quantum adversary method, also yields a lower bound of $\Omega\left(2^{n/2}/n^2\right)$ on the problem's *classical* randomized query complexity. This improves and simplifies a 1983 result of Aldous. Finally, in both the randomized and quantum cases, we give the first nontrivial lower bounds for finding local minima on grids of constant dimension $d \geq 3$.

Categories and Subject Descriptors

F.1.2 [**Computation by Abstract Devices**]: Modes of Computation

General Terms

Theory

Keywords

local search, local optima, query complexity (black box, decision tree), quantum computing, PLS

1. INTRODUCTION

This paper deals with the following problem.

LOCAL SEARCH. *Given an undirected graph $G = (V, E)$ and a function $f : V \to \mathbb{N}$, find a local minimum of f—that is, a vertex v such that $f(v) \leq f(w)$ for all neighbors w of v.*

We are interested in the number of *queries* that an algorithm needs to solve this problem, where a query just returns $f(v)$ given v. We consider deterministic, randomized, and quantum algorithms. Section 2 motivates the problem theoretically and practically; this section explains our results.

We start with some simple observations. If G is the complete graph of size N, then clearly $\Omega(N)$ queries are needed to find a local minimum (or $\Omega\left(\sqrt{N}\right)$ with a quantum computer [8]). At the other extreme, if G is a line of length N, then even a deterministic algorithm can find a local minimum in $O(\log N)$ queries, using binary search: query the middle two vertices, v and w. If $f(v) \leq f(w)$, then search the line of length $(N-2)/2$ connected to v; otherwise search the line connected to w. Continue recursively in this manner until a local minimum is found.

So the interesting case is when G is a graph of 'intermediate' connectedness: for example, the Boolean hypercube $\{0,1\}^n$, with two vertices adjacent if and only if they have Hamming distance 1. For this graph, Llewellyn, Tovey, and Trick [18] showed a $\Omega(2^n/\sqrt{n})$ lower bound on the number of queries needed by any deterministic algorithm, using a simple adversary argument. Intuitively, until the set of vertices queried so far comprises a *vertex cut* (that is, splits the graph into two or more connected components), an adversary is free to return a descending sequence of f-values: $f(v_1) = 2^n$ for the first vertex v_1 queried by the algorithm, $f(v_2) = 2^n - 1$ for the second vertex queried, and so on. Moreover, once the set of queried vertices does comprise a cut, the adversary can choose the largest connected component of unqueried vertices, and restrict the problem recursively to that component. So to lower-bound the deterministic query complexity, it suffices to lower-bound the size of any cut that splits the graph into two reasonably large components.[1] For the Boolean hypercube, Llewellyn et al. showed that the best one can do is essentially to query all $\Omega(2^n/\sqrt{n})$ vertices of Hamming weight $n/2$.

Llewellyn et al.'s argument fails completely in the case of randomized algorithms. By Yao's minimax principle, what we want here is a fixed *distribution* $\mathcal{D}$ over functions $f : \{0,1\}^n \to \mathbb{N}$, such that any deterministic algorithm needs many queries to find a local minimum of f, with high probability if f is drawn from $\mathcal{D}$. Taking $\mathcal{D}$ to be uniform will not do, since a local minimum of a uniform random func-

*University of California, Berkeley. Email: aaronson@cs.berkeley.edu. Parts of this work were done at the Hebrew University (Jerusalem, Israel) and the Perimeter Institute (Waterloo, Canada). Supported by an NSF Graduate Fellowship, by NSF ITR Grant CCR-0121555, and by the Defense Advanced Research Projects Agency (DARPA).

STOC'04, June 13–15, 2004, Chicago, Illinois, USA.

[1] Llewellyn et al. actually give a tight characterization of deterministic query complexity in terms of vertex cuts.

tion is easily found. However, Aldous [3] had the idea of defining $\mathcal{D}$ via a *random walk*, as follows. Choose a vertex $v_0 \in \{0,1\}^n$ uniformly at random; then perform an unbiased walk[2] $v_0, v_1, v_2, \ldots$ starting from v_0. For each vertex v, set $f(v)$ equal to the first hitting time of the walk at v—that is, $f(v) = \min\{t : v_t = v\}$. Clearly any f produced in this way has a unique local minimum at v_0, since for all $t > 0$, if vertex v_t is visited for the first time at step t then $f(v_t) > f(v_{t-1})$. Using sophisticated random walk analysis, Aldous managed to show a lower bound of $2^{n/2-o(n)}$ on the expected number of queries needed by any randomized algorithm to find v_0.[3] (As we will see in Section 3, this lower bound is close to tight.) Intuitively, since a random walk on the hypercube mixes in $O(n \log n)$ steps, an algorithm that has not queried a v with $f(v) < 2^{n/2}$ has almost no useful information about where the unique minimum v_0 is, so its next query will just be a "stab in the dark."

However, Aldous's result leaves several questions about Local Search unanswered. What if the graph G is a 3-D cube, on which a random walk does *not* mix very rapidly? Can we still lower-bound the randomized query complexity of finding a local minimum? More generally, what parameters of G make the problem hard or easy? Also, what is the quantum query complexity of Local Search?

This paper presents a new approach to Local Search, which we believe points the way to a complete understanding of its complexity. Our approach is based on the *quantum adversary method*, introduced by Ambainis [5] to prove lower bounds on quantum query complexity. Surprisingly, our approach yields new and simpler lower bounds for the problem's *classical* randomized query complexity, in addition to quantum lower bounds. Thus, along with recent work by Kerenidis and de Wolf [16] and by Aharonov and Regev [2], this paper illustrates how quantum ideas can help to resolve classical open problems.

Our results are as follows. For the Boolean hypercube $G = \{0,1\}^n$, we show that any quantum algorithm needs $\Omega\left(2^{n/4}/n\right)$ queries to find a local minimum on G, and any randomized algorithm needs $\Omega\left(2^{n/2}/n^2\right)$ queries (improving the $2^{n/2-o(n)}$ lower bound of Aldous [3]). Our proofs are elementary and do not require random walk analysis. By comparison, the best known upper bounds are $O\left(2^{n/3}n^{1/6}\right)$ for a quantum algorithm and $O\left(2^{n/2}\sqrt{n}\right)$ for a randomized algorithm. If G is a d-dimensional grid of size $N^{1/d} \times \cdots \times N^{1/d}$, where $d \geq 3$ is a constant, then we show that any quantum algorithm needs $\Omega\left(\sqrt{N^{1/2-1/d}/\log N}\right)$ queries to find a local minimum on G, and any randomized algorithm needs $\Omega\left(N^{1/2-1/d}/\log N\right)$ queries. No nontrivial lower bounds (randomized or quantum) were previously known in this case.[4]

In an earlier version of this paper, we raised as our "most ambitious" conjecture that the deterministic and quantum query complexities of local search are polynomially related for *every* family of graphs. At the time, it was not even known whether deterministic and *randomized* query complexities were polynomially related, not even for simple examples such as the 2-dimensional square grid. Recently Santha and Szegedy [21] spectacularly resolved our conjecture, by showing that the quantum query complexity is at least the 19^{th} root (!) of the deterministic complexity. Given that their result generalizes ours to such an extent, we feel obligated to explain why this paper is still relevant. First, for specific graphs such as the hypercube, our lower bounds are close to tight; those of Santha and Szegedy are not. Second, we give randomized lower bounds that are quadratically better than our quantum lower bounds; Santha and Szegedy give only quantum lower bounds.

In another recent development, Ambainis (personal communication) has improved our $\Omega\left(2^{n/4}/n\right)$ quantum lower bound for local search on the hypercube to $2^{n/3}/n^{O(1)}$, using a hybrid argument. Note that Ambainis' lower bound matches the upper bound up to a polynomial factor.

The paper is organized as follows. Section 2 motivates lower bounds on Local Search, pointing out connections to simulated annealing, quantum adiabatic algorithms, and the complexity class TFNP of total function problems. Section 3 defines notation and reviews basic facts about Local Search, including upper bounds. In Section 4 we give an intuitive explanation of Ambainis's quantum adversary method, then state and prove a classical analogue of Ambainis's main lower bound theorem. Section 5 introduces *snakes*, a construction by which we apply the two adversary methods to Local Search. We show there that to prove lower bounds for any graph G, it suffices to upper-bound a combinatorial parameter ε of a 'snake distribution' on G. Section 6 applies this framework to specific examples of graphs: the Boolean hypercube in Section 6.1, and the d-dimensional grid in Section 6.2.

2. MOTIVATION

Local search is the most effective weapon ever devised against hard optimization problems. For many real applications, neither backtrack search, nor approximation algorithms, nor even Grover's algorithm (assuming we had a quantum computer) can compare. Furthermore, along with quantum computing, local search (broadly defined) is one of the most interesting links between computer science and Nature. It is related to evolutionary biology via genetic algorithms, and to the physics of materials via simulated annealing. Thus it is both practically and scientifically important to understand its performance.

The conventional wisdom is that, although local search performs well in practice, its central (indeed defining) flaw is a tendency to get stuck at local optima. If this were correct, one corollary would be that the reason local search performs so well is that the problem it really solves—finding a local optimum—is intrinsically easy. It would thus be unnecessary to seek further explanations for its performance. Another corollary would be that, for *unimodal* functions (which have no local optima besides the global optimum), the global optimum would be easily found.

However, the conventional wisdom is false. The results of Llewellyn et al. [18] and Aldous [3] show that even if f is unimodal, any classical algorithm that treats f as a black box needs exponential time to find the global minimum of f

[2]Actually, Aldous used a continuous-time random walk, so the functions would be from $\{0,1\}^n$ to $\mathbb{R}$.

[3]Independently and much later, Droste et al. [12] showed the weaker bound $2^{g(n)}$ for any $g(n) = o(n)$.

[4]A lower bound on deterministic query complexity is known for such graphs [17].

in general. Our results extend this conclusion to quantum algorithms. In our view, the practical upshot of these results is that they force us to confront the question: What is it about 'real-world' problems that makes it easy to find a local optimum? That is, why do exponentially long chains of descending values, such as those used for lower bounds, almost never occur in practice (even in functions with large range sizes)? We do not know a good answer to this.

Our results are also relevant for physics. Many physical systems, including folding proteins and networks of springs and pulleys, can be understood as performing 'local search' through an energy landscape to reach a locally-minimal energy configuration. A key question is, how long will the system take to reach its ground state (that is, a globally-minimal configuration)? Of course, if there are local optima, the system might *never* reach its ground state, just as a rock in a mountain crevice does not roll to the bottom by going up first. But what if the energy landscape is unimodal? And moreover, what if the physical system is quantum? Our results show that, for certain energy landscapes, even a quantum system would take exponential time to reach its ground state, regardless of what Hamiltonian is applied to it. So in particular, the quantum adiabatic algorithm proposed by Farhi et al. [14], which can be seen as a quantum analogue of simulated annealing, needs exponential time to find a local minimum in the worst case.

Finally, our results have implications for so-called *total function problems* in complexity theory. Megiddo and Papadimitriou [19] defined a complexity class[5] TFNP, consisting (informally) of those NP search problems for which a solution always exists. For example, we might be given a function $f : \{0,1\}^n \rightarrow \{0,1\}^{n-1}$ as a Boolean circuit, and asked to find any distinct x, y pair such that $f(x) = f(y)$. This particular problem belongs to a subclass of TFNP called PPP (Polynomial Pigeonhole Principle). Notice that no promise is involved: the combinatorial nature of the problem itself forces a solution to exist, even if we have no idea how to find it. In a recent talk, Papadimitriou [20] asked broadly whether such 'nonconstructive existence problems' might be good candidates for efficient quantum algorithms. In the case of PPP problems, the collision lower bound of Aaronson [1] (improved by Shi [22] and others) implies a negative answer in the black-box setting. For other subclasses of TFNP, such as PODN (Polynomial Odd-Degree Node), a quantum black-box lower bound follows easily from the optimality of Grover's search algorithm.

However, there is one important subclass of TFNP for which no quantum lower bound was previously known. This is PLS (Polynomial Local Search), defined by Johnson, Papadimitriou, and Yannakakis [15] as a class of optimization problems whose cost function f and neighborhood function η (that is, the set of neighbors of a given point) are both computable in polynomial time. Given such a problem, the task is to output any local minimum of the cost function: that is, a v such that $f(v) \leq f(w)$ for all $w \in \eta(v)$. The lower bound of Llewellyn et al. [18] yields an oracle A relative to which $\mathsf{FP}^A \neq \mathsf{PLS}^A$, by a standard diagonalization argument along the lines of Baker, Gill, and Solovay [6]. Likewise, the lower bound of Aldous [3] yields an oracle relative to which $\mathsf{PLS} \not\subseteq \mathsf{FBPP}$, where FBPP is simply the function version of BPP. Our results yield the first oracle relative to which $\mathsf{PLS} \not\subseteq \mathsf{FBQP}$. In light of this oracle separation, we raise an admittedly vague question: is there a nontrivial "combinatorial" subclass of TFNP that we can show *is* contained in FBQP?

[5]See www.cs.berkeley.edu/~aaronson/zoo.html for details about the complexity classes mentioned in this paper.

3. PRELIMINARIES

In the Local Search problem, we are given an undirected graph $G = (V, E)$ with $N = |V|$, and oracle access to a function $f : V \rightarrow \mathbb{N}$. The goal is to find any *local minimum* of f, defined as a vertex $v \in V$ such that $f(v) \leq f(w)$ for all neighbors w of v. Clearly such a local minimum exists. We want to find one using as few queries as possible, where a query returns $f(v)$ given v. Queries can be adaptive; that is, can depend on the outcomes of previous queries. We assume G is known in advance, so that only f needs to be queried. Since we care only about query complexity, not computation time, there is no difficulty in dealing with an infinite range for f—though for our lower bounds, it will turn out that a range of size $O(|V|)$ suffices.

Our model of query algorithms is the standard one; see [10] for a survey. Given a graph G, the deterministic query complexity of Local Search on G, which we denote $\mathrm{DLS}(G)$, is $\min_\Gamma \max_f T(\Gamma, f, G)$ where the minimum ranges over all deterministic algorithms Γ, the maximum ranges over all f, and $T(\Gamma, f, G)$ is the number of queries made to f by Γ before it halts and outputs a local minimum of f (or ∞ if Γ fails to do so). The randomized query complexity $\mathrm{RLS}(G)$ is defined similarly, except that now the algorithm has access to an infinite random string R, and must only output a local minimum with probability at least $2/3$ over R. For simplicity, we assume that the number of queries T is the same for all R; clearly this assumption changes the complexity by at most a constant factor.

In the quantum model, an algorithm's state has the form $\sum_{v,z,s} \alpha_{v,z,s} |v, z, s\rangle$, where v is the label of a vertex in G, and z and s are strings representing the answer register and workspace respectively. The $\alpha_{v,z,s}$'s are complex amplitudes satisfying $\sum_{v,z,s} |\alpha_{v,z,s}|^2 = 1$. Starting from an arbitrary (fixed) initial state, the algorithm proceeds by an alternating sequence of *queries* and *algorithm steps*. A query maps each $|v, z, s\rangle$ to $|v, z \oplus f(v), s\rangle$, where $\oplus$ denotes bitwise exclusive-OR. An algorithm step multiplies the vector of $\alpha_{v,z,s}$'s by an arbitrary unitary matrix that does not depend on f. Letting $\mathcal{M}_f$ denote the set of local minima of f, the algorithm succeeds if at the end $\sum_{v,z,s \,:\, v \in \mathcal{M}_f} |\alpha_{v,z,s}|^2 \geq \frac{2}{3}$. Then the bounded-error quantum query complexity, or $\mathrm{QLS}(G)$, is defined as the minimum number of queries used by a quantum algorithm that succeeds on every f.

It is immediate that $\mathrm{QLS}(G) \leq \mathrm{RLS}(G) \leq \mathrm{DLS}(G) \leq N$. Also, letting δ be the maximum degree of G, we have the following trivial lower bound.

Proposition 1. $\mathrm{RLS}(G) = \Omega(\delta)$ *and* $\mathrm{QLS}(G) = \Omega\left(\sqrt{\delta}\right)$.

Proof. Let v be a vertex of G with degree δ. Choose a neighbor w of v uniformly at random, and let $f(w) = 1$. Let $f(v) = 2$, and $f(u) = 3$ for all neighbors u of v other than w. Let S be the neighbor set of v (including v itself); then for all $x \notin S$, let $f(x) = 3 + \Delta(x, S)$ where $\Delta(x, S)$ is the minimum distance from x to a vertex in S. Clearly f has a unique local minimum at w. However, finding y requires

exhaustive search among the δ neighbors of v, which takes $\Omega\left(\sqrt{\delta}\right)$ quantum queries by Bennett et al. [8]. □

A corollary of Proposition 1 is that classically, zero-error randomized query complexity is equivalent to bounded-error up to a constant factor. For given a candidate local minimum v, one can check using $O(\delta)$ queries that v is indeed a local minimum. Since $\Omega(\delta)$ queries are needed anyway, this verification step does not affect the overall complexity.

As pointed out by Aldous [3], a classical randomized algorithm can find a local minimum of f with high probability in $O\left(\sqrt{N\delta}\right)$ queries. The algorithm just queries $\sqrt{N\delta}$ vertices uniformly at random, and lets v_0 be a queried vertex for which $f(v)$ is minimal. It then follows v_0 to a local minimum by steepest descent. That is, for $t = 0, 1, 2, \ldots$, it queries all neighbors of v_t, halts if v_t is a local minimum, and otherwise sets v_{t+1} to be the neighbor w of v_t for which $f(w)$ is minimal (breaking ties by lexicographic ordering). A similar idea yields an improved quantum upper bound.

PROPOSITION 2. *For any G, $\mathrm{QLS}(G) = O\left(N^{1/3}\delta^{1/6}\right)$.*

PROOF. The algorithm first chooses $N^{2/3}\delta^{1/3}$ vertices of G uniformly at random, then uses Grover search to find a chosen vertex v_0 for which $f(v)$ is minimal. By a result of Dürr and Høyer [13], this can be done with high probability in $O\left(N^{1/3}\delta^{1/6}\right)$ queries. Next, for $t = 0, 1, 2, \ldots$, the algorithm performs Grover search over all neighbors of v_t, looking for a neighbor w such that $f(w) < f(v_t)$. If it finds such a w, then it sets $v_{t+1} := w$ and continues to the next iteration. Otherwise, it repeats the Grover search $\log(N/\delta)$ times before finally giving up and returning v_t as a claimed local minimum.

The expected number of u such that $f(u) < f(v_0)$ is at most $N/\left(N^{2/3}\delta^{1/3}\right) = (N/\delta)^{1/3}$. Since $f(v_{t+1}) < f(v_t)$ for all t, clearly the number of such u provides an upper bound on t. Furthermore, assuming there exists a w such that $f(w) < f(v_t)$, the expected number of repetitions of Grover's algorithm until such a w is found is $O(1)$. Since each repetition takes $O\left(\sqrt{\delta}\right)$ queries, by linearity of expectation the total expected number of queries used by the algorithm is therefore

$$O\left(N^{1/3}\delta^{1/6} + (N/\delta)^{1/3}\sqrt{\delta} + \log(N/\delta)\sqrt{\delta}\right)$$

or $O\left(N^{1/3}\delta^{1/6}\right)$. To see that the algorithm finds a local minimum with high probability, observe that for each t, the probability of not finding a w such that $f(w) < f(v_t)$, given that one exists, is at most $c^{-\log(N/\delta)} \leq (\delta/N)^{1/3}/10$ for a suitable constant c. So by the union bound, the probability that the algorithm returns a 'false positive' is at most $(N/\delta)^{1/3} \cdot (\delta/N)^{1/3}/10 = 1/10$. □

4. RELATIONAL ADVERSARY METHOD

We know of essentially two methods for proving lower bounds on quantum query complexity: the polynomial method of Beals et al. [7], and the quantum adversary method of Ambainis [5].[6] For a few problems, such as the collision problem [1, 22], the polynomial method succeeded where the adversary method failed. However, for problems that lack permutation symmetry (such as LOCAL SEARCH), the adversary method has proven more effective.[7]

How could a quantum lower bound method possibly be applied classically? When proving randomized lower bounds, the tendency is to attack "bare-handed": fix a distribution over inputs, and let $x_1, \ldots, x_t$ be the locations queried so far by the algorithm. Show that for small t, the posterior distribution over inputs, *conditioned* on $x_1, \ldots, x_t$, is still 'hard' with high probability—so that the algorithm knows almost nothing even about which location x_{t+1} to query next. This is essentially the approach taken by Aldous [3] to prove a $2^{n/2-o(n)}$ lower bound on $\mathrm{RLS}(\{0,1\}^n)$.

In the quantum case, however, it is unclear how to specify what an algorithm 'knows' after a given number of queries. So we are almost *forced* to step back, and identify general combinatorial properties of input sets that make them hard to distinguish. Once we have such properties, we can then try to exhibit them in functions of interest.

We believe this "gloved" attack can be useful for classical lower bounds as well as quantum ones. In our *relational adversary method*, we assume there exists a T-query randomized algorithm for function F. We consider a set $\mathcal{A}$ of 0-inputs of F, a set $\mathcal{B}$ of 1-inputs, and an arbitrary real-valued *relation function* $R(A, B) \geq 0$ for $A \in \mathcal{A}$ and $B \in \mathcal{B}$. Intuitively, $R(A, B)$ should be large if A and B differ in only a few locations. We then fix a probability distribution $\mathcal{D}$ over inputs; by Yao's minimax principle, there exists a T-query deterministic algorithm Γ^* that succeeds with high probability on inputs drawn from $\mathcal{D}$. Let W_A be the set of 0-inputs and W_B the set of 1-inputs on which Γ^* succeeds. Using the relation function R, we define a *separation measure* S between W_A and W_B, and show that (1) initially $S = 0$, (2) by the end of the computation S must be large, and (3) S increases by only a small amount as the result of each query. It follows that T must be large.

Undoubtedly any randomized lower bound proved using our relational method could also be proved "bare-handed," without any quantum intuition. However, our method makes it easier to focus on what is unique about a problem, and ignore what is common among many problems.

Our starting point is the "most general" adversary theorem in Ambainis's original paper (Theorem 6 in [5]), which he introduced to prove a quantum lower bound for the problem of inverting a permutation. Here the input is a permutation $\sigma(1), \ldots, \sigma(N)$, and the task is to output 0 if $\sigma^{-1}(1) \leq N/2$ and 1 otherwise. To lower-bound this problem's query complexity, what we would like to say is this:

Given any 0-input σ and any location x, if we choose a random 1-input τ that is 'related' to σ, then the probability $\theta(\sigma, x)$ over τ that $\sigma(x)$ does not equal $\tau(x)$ is small. In other words, the algorithm is unlikely to distinguish σ from a random neighbor τ of σ by querying x.

Unfortunately, the above claim is false. Letting $x = \sigma^{-1}(1)$, we have that $\sigma(x) \neq \tau(x)$ for *every* 1-input τ, and thus $\theta(\sigma, x) = 1$. Ambainis resolves this difficulty by letting us take the maximum, over all 0-inputs σ and 1-inputs τ that are related and differ at x, of the *geometric mean* $\sqrt{\theta(\sigma, x)\,\theta(\tau, x)}$. Even if $\theta(\sigma, x) = 1$, the geometric

[6] We are thinking here of the hybrid method [8] as a cousin of the adversary method.

[7] Indeed, Ambainis [4] has given problems for which the adversary method provably yields a better lower bound than the polynomial method.

mean is still small provided that $\theta(\tau, x)$ is small. More formally:

THEOREM 3 (AMBAINIS). *Let $\mathcal{A} \subseteq F^{-1}(0)$ and $\mathcal{B} \subseteq F^{-1}(1)$ be sets of inputs to function F. Let $R(A,B) \geq 0$ be a real-valued function, and for $A \in \mathcal{A}$, $B \in \mathcal{B}$, and location x, let*

$$\theta(A,x) = \frac{\sum_{B^* \in \mathcal{B} \,:\, A(x) \neq B^*(x)} R(A,B^*)}{\sum_{B^* \in \mathcal{B}} R(A,B^*)},$$

$$\theta(B,x) = \frac{\sum_{A^* \in \mathcal{A} \,:\, A^*(x) \neq B(x)} R(A^*,B)}{\sum_{A^* \in \mathcal{A}} R(A^*,B)},$$

where the denominators are all nonzero. Then the number of quantum queries needed to evaluate F with at least $9/10$ probability is $\Omega(1/\upsilon_{\text{geom}})$, where

$$\upsilon_{\text{geom}} = \max_{\substack{A \in \mathcal{A},\ B \in \mathcal{B},\ x \,:\\ R(A,B)>0,\ A(x) \neq B(x)}} \sqrt{\theta(A,x)\,\theta(B,x)}.$$

To illustrate we show the following.

PROPOSITION 4 (AMBAINIS). *The quantum query complexity of inverting a permutation is $\Omega\left(\sqrt{N}\right)$.*

PROOF. Let $\mathcal{A}$ be the set of all permutations σ such that $\sigma^{-1}(1) \leq N/2$, and $\mathcal{B}$ be the set of permutations τ such that $\tau^{-1}(1) > N/2$. Given $\sigma \in \mathcal{A}$ and $\tau \in \mathcal{B}$, let $R(\sigma,\tau) = 1$ if σ and τ differ only at locations $\sigma^{-1}(1)$ and $\tau^{-1}(1)$, and $R(\sigma,\tau) = 0$ otherwise. Then given σ, τ with $R(\sigma,\tau) = 1$, if $x \neq \sigma^{-1}(1)$ then $\theta(\sigma,x) = 2/N$, and if $x \neq \tau^{-1}(1)$ then $\theta(\tau,x) = 2/N$. So $\max_{x \,:\, \sigma(x) \neq \tau(x)} \sqrt{\theta(\sigma,x)\,\theta(\tau,x)} = \sqrt{2/N}$. □

The only difference between Theorem 3 and our relational adversary theorem is that in the latter, we take the *minimum* of $\theta(A,x)$ and $\theta(B,x)$ instead of the geometric mean. Taking the reciprocal then gives up to a quadratically better lower bound: for example, we obtain that the randomized query complexity of inverting a permutation is $\Omega(N)$. However, the proofs of the two theorems are quite different.

THEOREM 5. *Let $\mathcal{A}, \mathcal{B}, R, \theta$ be as in Theorem 3. Then the number of randomized queries needed to evaluate F with at least $9/10$ probability is $\Omega(1/\upsilon_{\min})$, where*

$$\upsilon_{\min} = \max_{\substack{A \in \mathcal{A},\ B \in \mathcal{B},\ x \,:\\ R(A,B)>0,\ A(x) \neq B(x)}} \min\{\theta(A,x), \theta(B,x)\}.$$

PROOF. Let Γ be a randomized algorithm that, given an input A, returns $F(A)$ with at least $9/10$ probability. Let T be the number of queries made by Γ. For all $A \in \mathcal{A}$, $B \in \mathcal{B}$, define

$$M(A) = \sum_{B^* \in \mathcal{B}} R(A,B^*),$$

$$M(B) = \sum_{A^* \in \mathcal{A}} R(A^*,B),$$

$$M = \sum_{A^* \in \mathcal{A}} M(A^*) = \sum_{B^* \in \mathcal{B}} M(B^*).$$

Now let $\mathcal{D}_A$ be the distribution over $A \in \mathcal{A}$ in which each A is chosen with probability $M(A)/M$; and let $\mathcal{D}_B$ be the distribution over $B \in \mathcal{B}$ in which each B is chosen with probability $M(B)/M$. Let $\mathcal{D}$ be an equal mixture of $\mathcal{D}_A$ and $\mathcal{D}_B$. By Yao's minimax principle, there exists a deterministic algorithm Γ^* that makes T queries, and succeeds with at least $9/10$ probability given an input drawn from $\mathcal{D}$. Therefore Γ^* succeeds with at least $4/5$ probability given an input drawn from $\mathcal{D}_A$ alone, or from $\mathcal{D}_B$ alone. In other words, letting W_A be the set of $A \in \mathcal{A}$ and W_B the set of $B \in \mathcal{B}$ on which Γ^* succeeds, we have

$$\sum_{A \in W_A} M(A) \geq \frac{4}{5}M, \quad \sum_{B \in W_B} M(B) \geq \frac{4}{5}M.$$

Define a predicate $P^{(t)}(A,B)$, which is true if Γ^* has distinguished $A \in \mathcal{A}$ from $B \in \mathcal{B}$ by the t^{th} query and false otherwise. (To distinguish A from B means to query an index x for which $A(x) \neq B(x)$, given either A or B as input.) Also, for all $A \in \mathcal{A}$, define a score function

$$S^{(t)}(A) = \sum_{B^* \in \mathcal{B} \,:\, P^{(t)}(A,B^*)} R(A,B^*).$$

This function measures how much "progress" has been made so far in separating A from $\mathcal{B}$-inputs, where the $\mathcal{B}$-inputs are weighted by $R(A,B)$. Similarly, for all $B \in \mathcal{B}$ define

$$S^{(t)}(B) = \sum_{A^* \in \mathcal{A} \,:\, P^{(t)}(A^*,B)} R(A^*,B).$$

It is clear that for all t,

$$\sum_{A \in \mathcal{A}} S^{(t)}(A) = \sum_{B \in \mathcal{B}} S^{(t)}(B).$$

So we can denote the above sum by $S^{(t)}$ and think of it as a global progress measure. We will show the following about $S^{(t)}$:

(i) $S^{(0)} = 0$ initially.

(ii) $S^{(T)} \geq 3M/5$ by the end.

(iii) $\Delta S^{(t)} \leq 3\upsilon_{\min} M$ for all t, where $\Delta S^{(t)} = S^{(t)} - S^{(t-1)}$ is the amount by which $S^{(t)}$ increases as the result of a single query.

It follows from (i)-(iii) that

$$T \geq \frac{3M/5}{3\upsilon_{\min} M} = \frac{1}{5\upsilon_{\min}}$$

which establishes the theorem. Part (i) is obvious. For part (ii), observe that for every pair (A,B) with $A \in W_A$ and $B \in W_B$, the algorithm Γ^* must query an x such that $A(x) \neq B(x)$. Thus

$$S^{(T)} \geq \sum_{A \in W_A,\ B \in W_B} R(A,B)$$

$$\geq \sum_{A \in W_A} M(A) - \sum_{B \notin W_B} M(B) \geq \frac{4}{5}M - \frac{1}{5}M.$$

It remains only to show part (iii). Suppose $\Delta S^{(t)} > 3\upsilon_{\min} M$ for some t; we will obtain a contradiction. Let

$$\Delta S^{(t)}(A) = S^{(t)}(A) - S^{(t-1)}(A),$$

and let C_A be the set of $A \in \mathcal{A}$ for which $\Delta S^{(t)}(A) > \upsilon_{\min} M(A)$. Since

$$\sum_{A \in \mathcal{A}} \Delta S^{(t)}(A) = \Delta S^{(t)} > 3\upsilon_{\min} M,$$

it follows by Markov's inequality that

$$\sum_{A \in C_A} \Delta S^{(t)}(A) \geq \frac{2}{3} \Delta S^{(t)}.$$

Similarly, if we let C_B be the set of $B \in \mathcal{B}$ for which $\Delta S^{(t)}(B) > \upsilon_{\min} M(B)$, we have

$$\sum_{B \in C_B} \Delta S^{(t)}(B) \geq \frac{2}{3} \Delta S^{(t)}.$$

In other words, at least 2/3 of the increase in $S^{(t)}$ comes from (A, B) pairs such that $A \in C_A$, and at least 2/3 comes from (A, B) pairs such that $B \in C_B$. Hence, by a 'pigeonhole' argument, there exists an $A \in C_A$ and $B \in C_B$ with $R(A, B) > 0$ that are distinguished by the t^{th} query. In other words, there exists an x with $A(x) \neq B(x)$, such that the t^{th} index queried by Γ^* is x whether the input is A or B. Then since $A \in C_A$, we have $\upsilon_{\min} M(A) < \Delta S^{(t)}(A)$, and hence

$$\upsilon_{\min} < \frac{\Delta S^{(t)}(A)}{M(A)} \leq \frac{\sum_{B^* \in \mathcal{B} \,:\, A(x) \neq B^*(x)} R(A, B^*)}{\sum_{B^* \in \mathcal{B}} R(A, B^*)}$$

which equals $\theta(A, x)$. Similarly $\upsilon_{\min} < \theta(B, x)$ since $B \in C_B$. This contradicts the definition

$$\upsilon_{\min} = \max_{\substack{A \in \mathcal{A},\ B \in \mathcal{B},\ x\,: \\ R(A,B)>0,\ A(x) \neq B(x)}} \min\{\theta(A, x), \theta(B, x)\},$$

and we are done. □

5. SNAKES

For our lower bounds, it will be convenient to generalize random walks to arbitrary distributions over paths, which we call *snakes*.

DEFINITION 6. *Given a vertex h in G and a positive integer L, a snake distribution $\mathcal{D}_{h,L}$ (parameterized by h and L) is a probability distribution over paths $(x_0, \ldots, x_{L-1})$ in G, such that each x_t is either equal or adjacent to x_{t+1}, and $x_{L-1} = h$. Let $D_{h,L}$ be the support of $\mathcal{D}_{h,L}$. Then an element of $D_{h,L}$ is called a snake; the part near x_0 is the tail and the part near $x_{L-1} = h$ is the head.*

Given a snake X and integer t, we use $X[t]$ as shorthand for $\{x_0, \ldots, x_t\}$.

DEFINITION 7. *We say a snake $X \in D_{h,L}$ is ε-good if the following holds. Choose j uniformly at random from $\{0, \ldots, L-1\}$, and let $Y = (y_0, \ldots, y_{L-1})$ be a snake drawn from $\mathcal{D}_{h,L}$ conditioned on $x_t = y_t$ for all $t > j$. Then*

(i) Letting $S_{X,Y}$ be the set of vertices v in $X \cap Y$ such that $\min\{t : x_t = v\} = \min\{t : y_t = v\}$, we have

$$\Pr_{j,Y}[X \cap Y = S_{X,Y}] \geq 9/10.$$

(ii) For all vertices v, $\Pr_{j,Y}[v \in Y[j]] \leq \varepsilon$.

The procedure above—wherein we choose a j uniformly at random, then draw a Y from $\mathcal{D}_{h,L}$ consistent with X on all steps later than j—will be important in what follows. We call it *the snake X flicking its tail*. Intuitively, a snake is good if it is spread out fairly evenly in G—so that when it flicks its tail, (1) with high probability the old and new tails do not intersect, and (2) any particular vertex is hit by the new tail with probability at most ε.

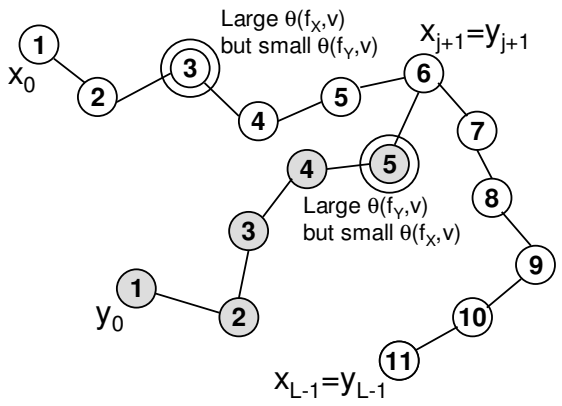

Figure 1: For every vertex v such that $f_X(v) \neq f_Y(v)$, either when snake X flicks its tail v is not hit with high probability, or when snake Y flicks its tail v is not hit with high probability.

We now explain our 'snake method' for proving lower bounds for LOCAL SEARCH. Given a snake X, we define an input f_X with a unique local minimum at x_0, and f-values that decrease along X from head to tail. Then, given inputs f_X and f_Y with $X \cap Y = S_{X,Y}$, we let the relation function $R(f_X, f_Y)$ be proportional to the probability that snake Y is obtained by X flicking its tail. (If $X \cap Y \neq S_{X,Y}$ we let $R = 0$.) Let f_X and g_Y be inputs with $R(f_X, g_Y) > 0$, and let v be a vertex such that $f_X(v) \neq g_Y(v)$. Then if all snakes were good, there would be two mutually exclusive cases: (1) v belongs to the tail of X, or (2) v belongs to the tail of Y. In case (1), v is hit with small probability when Y flicks its tail, so $\theta(f_Y, v)$ is small. In case (2), v is hit with small probability when X flicks its tail, so $\theta(f_X, v)$ is small. In either case, then, the *geometric mean* $\sqrt{\theta(f_X, v)\,\theta(f_Y, v)}$ and *minimum* $\min\{\theta(f_X, v), \theta(f_Y, v)\}$ are small. So even though $\theta(f_X, v)$ or $\theta(f_Y, v)$ could be large individually, Theorems 3 and 5 yield a good lower bound, as in the case of inverting a permutation (see Figure 1).

One difficulty is that not all snakes are good; at best, a large fraction of them are. We could try deleting all inputs f_X such that X is not good, but that might ruin some remaining inputs, which would then have fewer neighbors. So we would have to delete *those* inputs as well, and so on ad infinitum. What we need is basically a way to replace "all inputs" by "most inputs" in Theorems 3 and 5.

Fortunately, a simple graph-theoretic lemma can accomplish this. The lemma (see Diestel [11, p.6] for example) says that any graph with average degree at least k contains an induced subgraph with *minimum* degree at least $k/2$. Here we prove a weighted analogue of the lemma.

LEMMA 8. *Let $p(1), \ldots, p(m)$ be positive reals summing to 1. Also let $w(i, j)$ for $i, j \in \{1, \ldots, m\}$ be nonnegative reals satisfying $w(i, j) = w(j, i)$ and $\sum_{i,j} w(i, j) \geq r$. Then there exists a nonempty subset $U \subseteq \{1, \ldots, m\}$ such that for all $i \in U$, $\sum_{j \in U} w(i, j) \geq r\,p(i)/2$.*

PROOF. If $r = 0$ then the lemma trivially holds, so as-

sume $r > 0$. We construct U via an iterative procedure. Let $U(0) = \{1, \ldots, m\}$. Then for all t, if there exists an $i^* \in U(t)$ for which

$$\sum_{j \in U(t)} w(i^*, j) < \frac{r}{2} p(i^*),$$

then set $U(t+1) = U(t) \setminus \{i^*\}$. Otherwise halt and return $U = U(t)$. To see that the U so constructed is nonempty, observe that when we remove i^*, the sum $\sum_{i \in U(t)} p(i)$ decreases by $p(i^*)$, while $\sum_{i,j \in U(t)} w(i,j)$ decreases by at most

$$\sum_{j \in U(t)} w(i^*, j) + \sum_{j \in U(t)} w(j, i^*) < r p(i^*).$$

So since $\sum_{i,j \in U(t)} w(i,j)$ was positive to begin with, it must still be positive at the end of the procedure; hence U must be nonempty. □

We can now prove the main result of the section.

THEOREM 9. *Suppose a snake drawn from $\mathcal{D}_{h,L}$ is ε-good with probability at least 9/10. Then*

$$\mathrm{RLS}(G) = \Omega(1/\varepsilon), \qquad \mathrm{QLS}(G) = \Omega\left(\sqrt{1/\varepsilon}\right).$$

PROOF. Given a snake $X \in D_{h,L}$, we construct an input function f_X as follows. For each $v \in X$, let $f_X(v) = \min\{t : x_t = v\}$; and for each $v \notin X$, let $f_X(v) = \Delta(v,h) + L$ where $\Delta(v,h)$ is the distance from v to h in G. Clearly f_X so defined has a unique local minimum at x_0. To obtain a decision problem, we stipulate that querying x_0 reveals an answer bit (0 or 1) in addition to $f_X(x_1)$; the algorithm's goal is then to return the answer bit. Obviously a lower bound for the decision problem implies a corresponding lower bound for the search problem. Let us first prove the theorem in the case that all snakes in $D_{h,L}$ are ε-good. Let $p(X)$ be the probability of drawing snake X from $\mathcal{D}_{h,L}$. Also, given snakes X, Y and $j \in \{0, \ldots, L-1\}$, let $q_j(X,Y)$ be the probability that $X^* = Y$, if X^* is drawn from $\mathcal{D}_{h,L}$ conditioned on agreeing with X on all steps later than j. Then define

$$w(X,Y) = \frac{p(X)}{L} \sum_{j=0}^{L-1} q_j(X,Y).$$

Our first claim is that w is symmetric; that is, $w(X,Y) = w(Y,X)$. It suffices to show that

$$p(X)\, q_j(X,Y) = p(Y)\, q_j(Y,X)$$

for all j. We can assume X agrees with Y on all steps later than j, since otherwise $q_j(X,Y) = q_j(Y,X) = 0$. Given an $X^* \in D_{h,L}$, let A denote the event that X^* agrees with X (or equivalently Y) on all steps later than j, and let B_X (resp. B_Y) denote the event that X^* agrees with X (resp. Y) on steps 1 to j. Then

$$\begin{aligned} p(X)\, q_j(X,Y) &= \Pr[A] \Pr[B_X|A] \cdot \Pr[B_Y|A] \\ &= p(Y)\, q_j(Y,X). \end{aligned}$$

Now let $E(X,Y)$ denote the event that $X \cap Y = S_{X,Y}$, where $S_{X,Y}$ is as in Definition 7. Also, let f_X be the input obtained from X that has answer bit 0, and g_X be the input that has answer bit 1. To apply Theorems 3 and 5, take $\mathcal{A} = \{f_X : X \in D_{h,L}\}$ and $\mathcal{B} = \{g_X : X \in D_{h,L}\}$. Then take $R(f_X, g_Y) = w(X,Y)$ if $E(X,Y)$ holds, and $R(f_X, g_Y) = 0$ otherwise. Given $f_X \in \mathcal{A}$ and $g_Y \in \mathcal{B}$ with $R(f_X, g_Y) > 0$, and letting v be a vertex such that $f_X(v) \neq g_Y(v)$, we must then have either $v \notin X$ or $v \notin Y$. Suppose the former case; then

$$\begin{aligned} &\sum_{f_{X^*} \in \mathcal{A} \,:\, f_{X^*}(v) \neq g_Y(v)} R(f_{X^*}, g_Y) \\ &\leq \sum_{f_{X^*} \in \mathcal{A} \,:\, f_{X^*}(v) \neq g_Y(v)} \frac{p(Y)}{L} \sum_{j=0}^{L-1} q_j(Y, X^*) \leq \varepsilon p(Y), \end{aligned}$$

since Y is ε-good. Thus $\theta(g_Y, v)$ equals

$$\frac{\sum_{f_{X^*} \in \mathcal{A} \,:\, f_{X^*}(v) \neq g_Y(v)} R(f_{X^*}, g_Y)}{\sum_{f_{X^*} \in \mathcal{A}} R(f_{X^*}, g_Y)} \leq \frac{\varepsilon p(Y)}{9p(Y)/10}.$$

Similarly, if $v \notin Y$ then $\theta(f_X, v) \leq 10\varepsilon/9$ by symmetry. Hence

$$\upsilon_{\min} = \max_{\substack{f_X \in \mathcal{A},\ g_Y \in \mathcal{B},\ v: \\ R(f_X, g_Y) > 0, \\ f_X(v) \neq g_Y(v)}} \min\{\theta(f_X, v), \theta(g_Y, v)\} \leq \frac{\varepsilon}{9/10},$$

$$\upsilon_{\mathrm{geom}} = \max_{\substack{f_X \in \mathcal{A},\ g_Y \in \mathcal{B},\ v: \\ R(f_X, g_Y) > 0, \\ f_X(v) \neq g_Y(v)}} \sqrt{\theta(f_X, v)\, \theta(g_Y, v)} \leq \sqrt{\frac{\varepsilon}{9/10}},$$

the latter since $\theta(f_X, v) \leq 1$ and $\theta(g_Y, v) \leq 1$ for all f_X, g_Y and v. We now turn to the general case, in which a snake drawn from $\mathcal{D}_{h,L}$ is ε-good with probability at least 9/10. Let $G(X)$ denote the event that X is ε-good. Take $\mathcal{A}^* = \{f_X \in \mathcal{A} : G(X)\}$ and $\mathcal{B}^* = \{g_Y \in \mathcal{B} : G(Y)\}$, and take $R(f_X, g_Y)$ as before. Then since

$$\sum_{X,Y \,:\, E(X,Y)} w(X,Y) \geq \sum_X \frac{9}{10} p(X) \geq \frac{9}{10},$$

by the union bound we have

$$\begin{aligned} &\sum_{f_X \in \mathcal{A}^*,\ g_Y \in \mathcal{B}^*} R(f_X, g_Y) \\ &\geq \sum_{X,Y \,:\, G(X) \wedge G(Y) \wedge E(X,Y)} w(X,Y) \\ &\qquad - \sum_{X \,:\, \neg G(X)} p(X) - \sum_{Y \,:\, \neg G(Y)} p(Y) \\ &\geq \frac{9}{10} - \frac{1}{10} - \frac{1}{10} = \frac{7}{10}. \end{aligned}$$

So by Lemma 8, there exist subsets $\widetilde{\mathcal{A}} \subseteq \mathcal{A}^*$ and $\widetilde{\mathcal{B}} \subseteq \mathcal{B}^*$ such that for all $f_X \in \widetilde{\mathcal{A}}$ and $g_Y \in \widetilde{\mathcal{B}}$,

$$\sum_{g_{Y^*} \in \widetilde{\mathcal{B}}} R(f_X, g_{Y^*}) \geq \frac{7p(X)}{20},$$

$$\sum_{f_{X^*} \in \widetilde{\mathcal{A}}} R(f_{X^*}, g_Y) \geq \frac{7p(Y)}{20}.$$

So for all f_X, g_Y with $R(f_X, g_Y) > 0$, and all v such that $f_X(v) \neq g_Y(v)$, either $\theta(f_X, v) \leq 20\varepsilon/7$ or $\theta(g_Y, v) \leq 20\varepsilon/7$. Hence $\upsilon_{\min} \leq 20\varepsilon/7$ and $\upsilon_{\mathrm{geom}} \leq \sqrt{20\varepsilon/7}$. □

6. SPECIFIC GRAPHS

In this section we apply the 'snake method' developed in Section 5 to specific examples of graphs: the Boolean hypercube in Section 6.1, and the d-dimensional cubic grid (for $d \geq 3$) in Section 6.2.

6.1 Boolean Hypercube

Abusing notation, we let $\{0,1\}^n$ denote the n-dimensional Boolean hypercube—that is, the graph whose vertices are n-bit strings, with two vertices adjacent if and only if they have Hamming distance 1. Given a vertex $v \in \{0,1\}^n$, we let $v[0], \ldots, v[n-1]$ denote the n bits of v, and let $v^{(i)}$ denote the neighbor obtained by flipping bit $v[i]$. In this section we lower-bound $\mathrm{RLS}(\{0,1\}^n)$ and $\mathrm{QLS}(\{0,1\}^n)$.

Fix a 'snake head' $h \in \{0,1\}^n$ and take $L = 2^{n/2}/100$. We define the snake distribution $\mathcal{D}_{h,L}$ via what we call a *coordinate loop*, as follows. Starting from $x_0 = h$, for each t take $x_{t+1} = x_t$ with 1/2 probability, and $x_{t+1} = x_t^{(t \bmod n)}$ with 1/2 probability. The following is a basic fact about this distribution.

PROPOSITION 10. *The coordinate loop mixes completely in n steps, in the sense that if $t^* \geq t + n$, then x_{t^*} is a uniform random vertex conditioned on x_t.*

We could also use the random walk distribution, following Aldous [3]. However, not only is the coordinate loop distribution easier to work with (since it produces fewer self-intersections), it also yields a better lower bound (since it mixes completely in n steps, as opposed to approximately in $n \log n$ steps).

We first upper-bound the probability, over X, j, and $Y[j]$, that $X \cap Y \neq S_{X,Y}$ (where $S_{X,Y}$ is as in Definition 7).

LEMMA 11. *Suppose X is drawn from $\mathcal{D}_{h,L}$, j is drawn uniformly from $\{0, \ldots, L-1\}$, and $Y[j]$ is drawn from $\mathcal{D}_{x_j,j}$. Then $\Pr_{X,j,Y[j]}[X \cap Y = S_{X,Y}] \geq 0.9999$.*

PROOF. Call a *disagreement* a vertex v such that

$$\min\{t : x_t = v\} \neq \min\{t^* : y_{t^*} = v\}.$$

Clearly if there are no disagreements then $X \cap Y = S_{X,Y}$. If v is a disagreement, then by the definition of $\mathcal{D}_{h,L}$ we cannot have both $t > j - n$ and $t^* > j - n$. So by Proposition 10, either y_{t^*} is uniformly random conditioned on X, or x_t is uniformly random conditioned on $Y[j]$. Hence $\Pr_{X,j,Y[j]}[x_t = y_{t^*}] = 1/2^n$. So by the union bound,

$$\Pr_{X,j,Y[j]}[X \cap Y \neq S_{X,Y}] \leq \frac{L^2}{2^n} = 0.0001.$$

□

We now argue that, unless X spends a 'pathological' amount of time in one part of the hypercube, the probability of any vertex v being hit when X flicks its tail is small. To prove this, we define a notion of *sparseness*, and then show that (1) almost all snakes drawn from $\mathcal{D}_{h,L}$ are sparse (Lemma 13), and (2) sparse snakes are unlikely to hit any given vertex v (Lemma 14).

DEFINITION 12. *Given vertices v, w and $i \in \{0, \ldots, n-1\}$, let $\Delta(x, v, i)$ be the number of steps needed to reach v from x by first setting $x[i] := v[i]$, then setting $x[i-1] := v[i-1]$, and so on. (After we set $x[0]$ we wrap around to $x[n-1]$.) Then X is sparse if there exists a constant c such that for all $v \in \{0,1\}^n$ and all k,*

$$|\{t : \Delta(x_t, v, t \bmod n) = k\}| \leq cn\left(n + \frac{L}{2^{n-k}}\right).$$

LEMMA 13. *If X is drawn from $\mathcal{D}_{h,L}$, then X is sparse with probability $1 - o(1)$.*

PROOF. For each $i \in \{0, \ldots, n-1\}$, the number of $t \in \{0, \ldots, L-1\}$ such that $t \equiv i \pmod n$ is at most L/n. For such a t, let $E_t^{(v,i,k)}$ be the event that $\Delta(x_t, v, i) \leq k$; then $E_t^{(v,i,k)}$ holds if and only if

$$x_t[i] = v[i], \ldots, x_t[i-k+1] = v[i-k+1]$$

(where we wrap around to $x_t[n-1]$ after reaching $x_t[0]$). This occurs with probability $2^k/2^n$ over X. Furthermore, by Proposition 10, the $E_t^{(v,i,k)}$ events for different t's are independent. So let

$$\mu_k = \frac{L}{n} \cdot \frac{2^k}{2^n};$$

then for fixed v, i, k, the expected number of t's for which $E_t^{(v,i,k)}$ holds is at most μ_k. Thus by a Chernoff bound, if $\mu_k \geq 1$ then

$$\Pr_X\left[\left|\left\{t : E_t^{(v,i,k)}\right\}\right| > cn \cdot \mu_k\right] < \left(\frac{e^{cn-1}}{(cn)^{cn}}\right)^{\mu_k} < \frac{1}{2^{2n}}$$

for sufficiently large c. Similarly, if $\mu_k < 1$ then

$$\Pr_X\left[\left|\left\{t : E_t^{(v,i,k)}\right\}\right| > cn\right] < \left(\frac{e^{cn/\mu_k - 1}}{(cn/\mu_k)^{cn/\mu_k}}\right)^{\mu_k} < \frac{1}{2^{2n}}$$

for sufficiently large c. By the union bound, then,

$$\left|\left\{t : E_t^{(v,i,k)}\right\}\right| \leq cn \cdot (1 + \mu_k) = c\left(n + \frac{L}{2^{n-k}}\right)$$

for every v, i, k triple *simultaneously* with probability at least $1 - n^2 2^n / 2^{2n} = 1 - o(1)$. Summing over all i's produces the additional factor of n. □

LEMMA 14. *If X is sparse, then for every $v \in \{0,1\}^n$,*

$$\Pr_{j,Y}[v \in Y[j]] = O\left(\frac{n^2}{L}\right).$$

PROOF. By assumption, for every $k \in \{0, \ldots, n\}$,

$$\begin{aligned}\Pr_j[\Delta(x_j, v, j \bmod n) = k] &\leq \frac{|\{t : \Delta(x_t, v, t \bmod n) = k\}|}{L} \\ &\leq \frac{cn}{L}\left(n + \frac{L}{2^{n-k}}\right).\end{aligned}$$

Consider the probability that $v \in Y[j]$ in the event that $\Delta(x_j, v, j \bmod n) = k$. Clearly

$$\Pr_Y[v \in \{y_{j-n+1}, \ldots, y_j\}] = \frac{1}{2^k}.$$

Also, Proposition 10 implies that for every $t \leq j - n$, the probability that $y_t = v$ is 2^{-n}. So by the union bound,

$$\Pr_Y[v \in \{y_0, \ldots, y_{j-n}\}] \leq \frac{L}{2^n}.$$

Then $\Pr_{j,Y}[v \in Y[j]]$ equals

$$\begin{aligned}&\sum_{k=0}^{n}\left(\begin{array}{c}\Pr_j[\Delta(x_j, v, j \bmod n) = k] \cdot \\ \Pr_Y[v \in Y[j] \mid \Delta(x_j, v, j \bmod n) = k]\end{array}\right) \\ &\leq \sum_{k=0}^{n}\frac{cn}{L}\left(n + \frac{L}{2^{n-k}}\right)\left(\frac{1}{2^k} + \frac{L}{2^n}\right) = O\left(\frac{cn^2}{L}\right)\end{aligned}$$

as can be verified by breaking the sum into cases and doing some manipulations. □

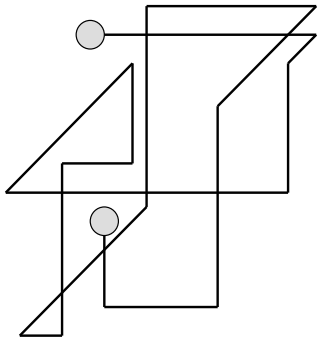

Figure 2: In $d = 3$ dimensions, a snake drawn from $\mathcal{D}_{h,L}$ moves a random distance left or right, then a random distance up or down, then a random distance inward or outward, etc.

The main result follows easily:

THEOREM 15.

$$\mathrm{RLS}\left(\{0,1\}^n\right) = \Omega\left(\frac{2^{n/2}}{n^2}\right), \quad \mathrm{QLS}\left(\{0,1\}^n\right) = \Omega\left(\frac{2^{n/4}}{n}\right).$$

PROOF. Take $\varepsilon = n^2/2^{n/2}$. Then by Theorem 9, it suffices to show that a snake X drawn from $\mathcal{D}_{h,L}$ is $O(\varepsilon)$-good with probability at least 9/10. First, since

$$\Pr_{X,j,Y[j]}\left[X \cap Y = S_{X,Y}\right] \geq 0.9999$$

by Lemma 11, Markov's inequality shows that

$$\Pr_X\left[\Pr_{j,Y[j]}\left[X \cap Y = S_{X,Y}\right] \geq \frac{9}{10}\right] \geq \frac{19}{20}.$$

Second, by Lemma 13, X is sparse with probability $1 - o(1)$, and by Lemma 14, if X is sparse then

$$\Pr_{j,Y}\left[v \in Y[j]\right] = O\left(\frac{n^2}{L}\right) = O(\varepsilon)$$

for every v. So both requirements of Definition 7 hold simultaneously with probability at least 9/10. □

6.2 Constant-Dimensional Grid Graph

In the Boolean hypercube case, we defined $\mathcal{D}_{h,L}$ by a 'coordinate loop' instead of the usual random walk mainly for convenience. When we move to the d-dimensional grid, though, the drawbacks of random walks become more serious: first, the mixing time is too long, and second, there are too many self-intersections, particularly if $d \leq 4$. Our snake distribution will instead use straight lines of randomly chosen lengths attached at the endpoints, as in Figure 2. Let $G_{d,N}$ be a d-dimensional grid graph with $d \geq 3$. That is, $G_{d,N}$ has N vertices of the form $v = (v[0], \ldots, v[d-1])$, where each $v[i]$ is in $\left\{1, \ldots, N^{1/d}\right\}$ (we assume for simplicity that N is a d^{th} power). Vertices v and w are adjacent if and only if $|v[i] - w[i]| = 1$ for some $i \in \{0, \ldots, d-1\}$, and $v[j] = w[j]$ for all $j \neq i$ (so $G_{d,N}$ does not wrap around at the boundaries).

We take $L = \sqrt{N}/100$, and define the snake distribution $\mathcal{D}_{h,L}$ as follows. Starting from $x_0 = h$, for each T we take $x_{N^{1/d}(T+1)}$ identical to $x_{N^{1/d}T}$, but with the $(T \bmod d)^{th}$ coordinate $x_{N^{1/d}(T+1)}[T \bmod d]$ replaced by a uniform random value in $\left\{1, \ldots, N^{1/d}\right\}$. We then take the vertices $x_{N^{1/d}T+1}, \ldots, x_{N^{1/d}T+N^{1/d}-1}$ to lie along the shortest path from $x_{N^{1/d}T}$ to $x_{N^{1/d}(T+1)}$, 'stalling' at $x_{N^{1/d}(T+1)}$ once that vertex has been reached. We call

$$\Phi_T = \left(x_{N^{1/d}T}, \ldots, x_{N^{1/d}T+N^{1/d}-1}\right)$$

a *line* of vertices, whose *direction* is $T \bmod d$. As in the Boolean hypercube case, we have:

PROPOSITION 16. *$\mathcal{D}_{h,L}$ mixes completely in $dN^{1/d}$ steps, in the sense that if $T^* \geq T + d$, then $x_{N^{1/d}T^*}$ is a uniform random vertex conditioned on $x_{N^{1/d}T}$.*

Lemma 11 in Section 6.1 goes through essentially without change.

DEFINITION 17. *Letting $\Delta(x, v, i)$ be as before, we say X is sparse if there exists a constant c (possibly dependent on d) such that for all vertices v and all k,*

$$\left|\left\{t : \Delta\left(x_t, v, \left\lfloor t/N^{1/d}\right\rfloor \bmod d\right) = k\right\}\right| \leq (c \log N)\left(N^{1/d} + \frac{L}{N^{1-k/d}}\right).$$

LEMMA 18. *If X is drawn from $\mathcal{D}_{h,L}$, then X is sparse with probability $1 - o(1)$.*

PROOF. Similar to Lemma 13. Let Φ_T be a line of vertices with direction $i = T \bmod d$, and notice that $\Delta(x_t, v, i)$ is the same for every vertex x_t in Φ_T. Let $E_T^{(v,i,k)}$ denote the event that $\Delta(x_t, v, i) \leq k$ for the x_t's in Φ_T. Then $E_T^{(v,i,k)}$ occurs with probability $N^{(k-1)/d}/N$ over X. Furthermore, if $|T - T^*| \geq d$ then $E_T^{(v,i,k)}$ and $E_{T^*}^{(v,i,k)}$ are independent events. So let

$$\mu_k = L \cdot \frac{N^{(k-1)/d}}{N};$$

then for fixed v, i, k, the expected number of lines for which $E_T^{(v,i,k)}$ holds is at most μ_k. Thus, by a Chernoff bound, if $\mu_k \geq 1$ then

$$\Pr_X\left[\left|\left\{T : E_T^{(v,i,k)}\right\}\right| > c \log N \cdot \mu_k\right] < \left(\frac{e^{c \log N - 1}}{(c \log N)^{c \log N}}\right)^{\mu_k}$$

which is at most $1/N^2$ for sufficiently large c. Similarly, if $\mu_k < 1$ then letting $m = (c \log N)/\mu_k$,

$$\Pr_X\left[\left|\left\{T : E_T^{(v,i,k)}\right\}\right| > c \log N\right] < \left(\frac{e^{m-1}}{m^m}\right)^{\mu_k} < \frac{1}{N^2}$$

for sufficiently large c. So with probability $1 - o(1)$ it holds that for all v, k, letting $i_t = \left\lfloor t/N^{1/d}\right\rfloor \bmod d$,

$$\left|\{t : \Delta(x_t, v, i_t) = k\}\right| \leq c \log N \cdot (1 + \mu_k) \cdot N^{1/d} = (c \log N)\left(N^{1/d} + \frac{L}{N^{1-k/d}}\right).$$

□

LEMMA 19. *If X is sparse, then for every $v \in G_{d,N}$,*

$$\Pr_{j,Y}\left[v \in Y[j]\right] = O\left(\frac{N^{1/d} \log N}{L}\right),$$

where the big-O hides a constant dependent on d.

PROOF. As in Lemma 14, setting $i_j = \left\lfloor j/N^{1/d} \right\rfloor \bmod d$ we obtain that $\Pr_{j,Y}[v \in Y[j]]$ equals

$$\sum_{k=1}^{d} \Pr_j [\Delta(x_j, v, i_j) = k] \Pr_Y [v \in Y[j] \mid \Delta(x_j, v, i_j) = k]$$
$$\leq \sum_{k=1}^{d} \frac{c \log N}{L} \left(N^{1/d} + \frac{L}{N^{1-k/d}} \right) \left(\frac{1}{N^{(k-1)/d}} + \frac{L}{N} \right)$$
$$= O\left(\frac{N^{1/d} \log N}{L} \right).$$

□

Taking $\varepsilon = (\log N)/N^{1/2-1/d}$ we get, by the same proof as for Theorem 15:

THEOREM 20. *Neglecting a constant dependent on d, for all $d \geq 3$*

$$\mathrm{RLS}(G_{d,N}) = \Omega\left(\frac{N^{1/2-1/d}}{\log N} \right),$$
$$\mathrm{QLS}(G_{d,N}) = \Omega\left(\sqrt{\frac{N^{1/2-1/d}}{\log N}} \right).$$

7. ACKNOWLEDGMENTS

I thank Andris Ambainis for suggesting an improvement to Proposition 2; David Aldous, Christos Papadimitriou, Yuval Peres, and Umesh Vazirani for discussions during the early stages of this work; and Ronald de Wolf and the anonymous reviewers for helpful comments.

8. REFERENCES

[1] S. Aaronson. Quantum lower bound for the collision problem, *Proc. ACM STOC*, pp. 635–642, 2002. quant-ph/0111102.

[2] D. Aharonov and O. Regev. Approximating the shortest and closest vector in a lattice to within $\sqrt{n}$ are in $\mathsf{NP} \cap \mathsf{coNP}$, unpublished.

[3] D. Aldous. Minimization algorithms and random walk on the d-cube, *Annals of Probability* 11(2):403–413, 1983.

[4] A. Ambainis. Polynomial degree vs. quantum query complexity, *Proc. IEEE FOCS*, pp. 230–239, 2003. quant-ph/0305028.

[5] A. Ambainis. Quantum lower bounds by quantum arguments, *J. Comput. Sys. Sci.* 64:750–767, 2002. Earlier version in *STOC 2000*. quant-ph/0002066.

[6] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question, *SIAM J. Comput.* 4:431–442, 1975.

[7] 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 *FOCS 1998*. quant-ph/9802049.

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

[9] H. Buhrman, R. Cleve, R. de Wolf, and Ch. Zalka. Bounds for small-error and zero-error quantum algorithms, *Proc. IEEE FOCS*, pp. 358–368, 1999. cs.CC/9904019.

[10] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey, *Theoretical Comput. Sci.* 288:21–43, 2002.

[11] R. Diestel. *Graph Theory* (2nd edition), Springer-Verlag, 2000.

[12] S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization, ECCC TR03-048, 2003.

[13] C. Dürr and P. Høyer. A quantum algorithm for finding the minimum, 1996. quant-ph/9607014.

[14] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem, *Science* 292:472–476, 2001. quant-ph/0104129.

[15] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search?, *J. Comput. Sys. Sci.* 37:79–100, 1988.

[16] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument, *Proc. ACM STOC*, pp. 106–115, 2003. quant-ph/0208062.

[17] D. C. Llewellyn and C. Tovey. Dividing and conquering the square, *Discrete Appl. Math* 43:131–153, 1993.

[18] D. C. Llewellyn, C. Tovey, and M. Trick. Local optimization on graphs, *Discrete Appl. Math* 23:157–178, 1989. Erratum: 46:93–94, 1993.

[19] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems, and computational complexity, *Theoret. Comp. Sci.* 81:317–324, 1991.

[20] C. H. Papadimitriou. Talk at UC Berkeley, February 6, 2003.

[21] M. Santha and M. Szegedy. Quantum and classical query complexities of local search are polynomially related, this *Proceedings*, 2004.

[22] Y. Shi. Quantum lower bounds for the collision and the element distinctness problems, *Proc. IEEE FOCS*, pp. 513–519, 2002. quant-ph/0112086.