New Upper and Lower Bounds for Randomized and Quantum Local Search [*]

Shengyu Zhang

Computer Science Department, Princeton University
35 Olden Street, Princeton, NJ 08540, USA.
szhang@cs.princeton.edu

ABSTRACT

Local Search problem, which finds a local minimum of a black-box function on a given graph, is of both practical and theoretical importance to combinatorial optimization, complexity theory and many other areas in theoretical computer science. In this paper, we study the problem in the randomized and quantum query models and give new lower and upper bound techniques in both models.

The lower bound technique works for any graph that contains a product graph as a subgraph. Applying it to the Boolean hypercube $\{0,1\}^n$ and the constant dimensional grids $[n]^d$, two particular product graphs that recently drew much attention, we get the following tight results:

$$RLS(\{0,1\}^n) = \Theta(2^{n/2}n^{1/2}),\ QLS(\{0,1\}^n) = \Theta(2^{n/3}n^{1/6});$$
$$RLS([n]^d) = \Theta(n^{d/2}), \forall d \geq 4,\ QLS([n]^d) = \Theta(n^{d/3}), \forall d \geq 6.$$

Here $RLS(G)$ and $QLS(G)$ are the randomized and quantum query complexities of Local Search on G, respectively. These improve the previous results by Aaronson [2], Ambainis (unpublished) and Santha and Szegedy[20].

Our new algorithms work well when the underlying graph expands slowly. As an application to $[n]^2$, a new quantum algorithm using $O(\sqrt{n}(\log\log n)^{1.5})$ queries is given. This improves the previous best known upper bound of $O(n^{2/3})$ (Aaronson, [2]), and implies that Local Search on grids exhibits different properties in low dimensions.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory

Keywords

local search, query complexity (decision tree complexity), quantum algorithm, randomized algorithm, lower bound

1. INTRODUCTION

Many important combinatorial optimization problems arising in both theory and practice are **NP**-hard, which forces people to resort to heuristic searches in practice. One popular approach is Local Search, by which one first defines a *neighborhood structure*, then finds a solution that is locally optimal with respect to this neighborhood structure. In the past two decades, Local Search approach has been extensively developed and "has reinforced its position as a standard approach in combinatorial optimization" in practice [1]. Besides the practical applications, the problem also has many connections to the complexity theory, especially to the complexity classes **PLS** [1] and **TFNP** [2]. For example, the 2SAT-FLIP problem is Local Search on the Boolean hypercube graph $\{0,1\}^n$, with the objective function being the sum of the weights of the clauses that the truth assignment $x \in \{0,1\}^n$ satisfies. This problem is complete in **PLS**, implying that the Boolean hypercube $\{0,1\}^n$ has a central position in the studies of Local Search. Local Search is also related to physical systems including folding proteins and to the quantum adiabatic algorithms [2]. We refer readers to the papers [2, 19, 20] for more discussions and the book [3] for a comprehensive introduction.

Precisely, Local Search on an undirected graph $G = (V, E)$ is defined as follows. Given a function $f : V \to \mathbb{N}$, find a vertex $v \in V$ such that $f(v) \leq f(w)$ for all neighbors w of v. A class of *generic algorithms* that has been widely used in practice is as follows: we first set out with an initial point $v \in V$, then repeatedly search a better/best neighbor until it reaches a local minimum. Though empirically this class of algorithms work very well in most applications, relatively few theoretical results are known about how good the generic algorithms are, especially for the randomized (and quantum) algorithms.

Among models for the theoretical studies, the query model has drawn much attention [2, 4, 5, 16, 17, 20]. In this model, $f(v)$ can only be accessed by querying v, and the randomized (and quantum) query complexity, denote by $RLS(G)$ (and $QLS(G)$) is the minimum number of queries needed by a

[*]This research was supported in part by NSF grants CCR-0310466 and CCF-0426582.

[1]Polynomial Local Search, introduced by Johnson, Papadimitriou, and Yannakakis [14].
[2]The family of total function problems, introduced by Megiddo and Papadimitriou [18].

randomized (and quantum) algorithm that finds a local minimum on G with high probability. Previous upper bounds on a general N-vertex graph G are $RLS(G) = O(\sqrt{N\delta})$ by Aldous [4] and $QLS(G) = O(N^{1/3}\delta^{1/6})$ by Aaronson [2], where δ is the maximum degree of G. Both algorithms actually fall into the category of generic algorithms mentioned above, with the initial point picked as a best one over a certain number of random samples. Immediately, two questions can be asked:

1. On what graphs are these simple algorithms optimal?
2. For other graphs, what better algorithms can we have?

Clearly the first one is a lower bound question and the second one is an upper bound question.

Previously for lower bounds, Aaronson [2] showed the following results on two special classes of graphs: the Boolean hypercube $\{0,1\}^n$ and the constant dimensional grid $[n]^d$:

$$RLS(\{0,1\}^n) = \Omega(2^{\frac{n}{2}}/n^2), \quad QLS(\{0,1\}^n) = \Omega(2^{\frac{n}{4}}/n),$$
$$RLS([n]^d) = \Omega(\tfrac{n^{\frac{d}{2}-1}}{\log n}), \qquad QLS([n]^d) = \Omega(\tfrac{n^{\frac{d}{4}-\frac{1}{2}}}{\sqrt{\log n}}).$$

It has also been shown that $QLS([n]^2) = \Omega(n^{1/4})$ by Santha and Szegedy in [20], besides their main result that the deterministic and the quantum query complexities of Local Search on any graph are polynomially related. However,

3. What are the final values of QLS and RLS on $\{0,1\}^n$ and $[n]^d$?

remains an open problem, explicitly asked in an earlier version of [2] and also (partially) in [20].

In this paper, we answer questions 1 and 2 for large classes of graphs by giving both new lower and upper bound techniques for randomized and quantum query algorithms. As a consequence, we completely solve the question 3, except for a few small d's where our new bounds also significantly improve the old ones.

Our lower bound technique works for any graph that contains a product graph as a subgraph. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their product $G_1 \times G_2$ is the graph $G = (V, E)$ where $V = V_1 \times V_2$ and

$$\begin{aligned} E = \ & \{((v_1,v_2),(v_1',v_2)) : (v_1,v_1') \in E_1, v_2 \in V_2\} \\ & \bigcup \{((v_1,v_2),(v_1,v_2')) : (v_2,v_2') \in E_2, v_1 \in V_1\}. \end{aligned}$$

We will also use the notion of random walk on graphs to state the theorem. Given a graph $G = (V, E)$, a random walk is a mapping $W : V \to 2^V$ where $W(u) \subseteq \{u\} \cup \{v : (u,v) \in E\}$. Intuitively, at each step the random walk W goes from the current vertex u to a uniformly random vertex in $W(u)$. The walk W is *regular* if $|W(u)| = c$ for each $u \in V$. Denote by $p(u,v,t)$ the probability that the random walk starting at u is at v after exactly t steps. Let $p_t = \max_{u,v} p(u,v,t)$. The following theorem is a special case of the general one (Theorem 9) in Section 3.

THEOREM 1. *Suppose G contains the product graph $G_1 \times G_2$ as a subgraph, and L is the length of the longest self-avoiding path in G_2. Let $T = \lfloor L/2 \rfloor$, then for any regular random walk W on G_1, we have*

$$RLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} p_t}\right), \quad QLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} \sqrt{p_t}}\right).$$

The proof uses the quantum adversary method, which was originally proposed by Ambainis [7] and later generalized in different ways [6, 8, 15, 23]. Recently Spalek and Szegedy made the picture clear by showing that all these generalizations are equivalent in power [21]. On the other hand, in proving a particular problem, some of the methods might be easier to apply than the others. In our case, the technique in [23], which generalizes the one in [6] slightly in the form though, turns out to work very well. Our proofs for the randomized lower bounds will use the relational adversary method, which was proposed by Aaronson [2] inspired by the quantum adversary method.

Both the quantum adversary method and the relational adversary method are parameterized by input sets and weight functions of input pairs. While previous works [2, 20] also use random walks on graphs, a key innovation that distinguishes our work from the previous ones and yields better lower bounds is that we decompose the graph into two parts, the tensor product of which is the original graph. We perform the random walk only in one part, and perform a simple one-way walk in a self-avoiding path in the other part, which serves as a "clock" to record the number of steps taken by the random walk in the first part. The tensor product of these two walks is a random path in the original graph. A big advantage of adding a clock is that the "passing probability", the probability that the random path *passes* a vertex v *within* some number of steps, is now the "hitting probability", the probability that the random walk in the first graph *hits* v *after* exactly some number of steps, because the time elapses one-way and never comes back. The fact that the hitting probability is much smaller than the passing probability enables us to achieve the better lower bounds. Another advantage of the clock is that since the walk in the second part is self-avoiding, the resulting random path in the original graph is self-avoiding too, which makes the analysis much easier.

Applying it to the two graphs $\{0,1\}^n$ and $[n]^d$, we improve previous results and show tight bounds on both RLS and QLS (except for a few cases in the low dimensional cubes).

THEOREM 2.

$$RLS(\{0,1\}^n) = \Theta(2^{n/2}n^{1/2}), \ QLS(\{0,1\}^n) = \Theta(2^{n/3}n^{1/6}).$$

THEOREM 3.

$$RLS([n]^d) = \begin{cases} \Theta(n^{d/2}) & \text{if } d \ge 4, \\ \Omega((n^3/\log n)^{1/2}) & \text{if } d = 3, \\ \Omega(n^{2/3}) & \text{if } d = 2. \end{cases}$$

$$QLS([n]^d) = \begin{cases} \Theta(n^{d/3}) & \text{if } d \ge 6, \\ \Omega((n^5/\log n)^{1/3}) & \text{if } d = 5, \\ \Omega(n^{6/5}),\ \Omega(n^{3/4}),\ \Omega(n^{2/5}) & \text{if } d = 4, 3, 2. \end{cases}$$

It is worth to note that to apply Theorem 1, we need not only know the mixing time of the random walk in G_1, but also know *its behavior before mixing*. So the applications are not simply using standard upper bounds on the mixing times, but involving heavy analysis on the whole mixing processes.

When proving Theorem 3 by Theorem 1, one difficulty arises: to decompose the grid $[n]^d$ into two parts $[n]^m$ and $[n]^{d-m}$, we implicitly require that m is an integer. This lets us get lower bounds weaker than Theorem 3, especially for low dimension cases. We get around this problem by cutting one of the m dimensions into many blocks, and use different block to distinguish different time windows. Between adjacent blocks are pairwise disjoint path segments, which thus thread all the blocks into a very long one. Using this technique, we can apply Theorem 1 for any read-number dimension $m \leq d-1$.

In the second part of the paper, we consider upper bounds for Local Search. While the generic algorithms [2, 4] are simple and proven to be optimal for many graphs such as the ones mentioned above, they are far from optimal for some other graphs. For example, it is not hard to see an $O(\log N)$ *deterministic* algorithm for the line graph G. Therefore, a natural question is to characterize those graphs on which Local Search is easy. It turns out that the expansion speed plays a key role. For a graph $G = (V, E)$, the distance $|u-v|$ between two vertices u and v is the length of the shortest path connecting them. (Here the length of a path is the number of edges on the path.) Let $c(k) = \max_{v\in V} |\{u : |u-v| \leq k\}|$. Apparently, the smaller $c(k)$ is, the more slowly the graph expands. (Actually $c(k)$ is an upper bound of the standard definition of the expanding speed.) As a special case of Theorem 12 in Section 5, the following upper bounds in terms of $c(k)$ hold.

THEOREM 4. *If $c(k) = O(k^\alpha)$ for some constant $\alpha \geq 1$, then*

$$RLS(G) = \begin{cases} O\left(d^{\alpha-1}\log\log d\right) & \text{if } \alpha > 1 \\ O(\log d \log\log d) & \text{if } \alpha = 1 \end{cases},$$

$$QLS(G) = \begin{cases} O\left(d^{\frac{\alpha-1}{2}}(\log\log d)^{1.5}\right) & \text{if } \alpha > 1 \\ O(\log d \log\log d) & \text{if } \alpha = 1 \end{cases}.$$

where d is the diameter of the graph G.

As a special case, on the line graph we get $\alpha = 1$ and hence $RLS = O(\log n \log\log n)$, which helps to explain why Local Search on the line graph is easy. Also, it immediately gives a new upper bound for $QLS([n]^2)$ as follows. Together with Theorem 3, this implies that Local Search on grids exhibits different properties in low dimensions.

THEOREM 5. $QLS([n]^2) = O(\sqrt{n}(\log\log n)^{1.5})$

Other related results. After the preliminary version of this paper appeared, Verhoeven independently showed an upper bound in terms of the genus of the graph [22], giving an $O(\sqrt{n}\log\log n)$ quantum algorithm for $[n]^2$. There is also an unpublished result on $QLS(\{0,1\}^n)$: it is mentioned in [2] that Ambainis showed $QLS(\{0,1\}^n) = \Omega(2^{n/3}/n^{O(1)})$. [3]

2. PRELIMINARIES AND NOTATIONS

We use $[M]$ to denote the set $\{1, 2, ..., M\}$. For an n-bit binary string $x = x_0...x_{n-1} \in \{0,1\}^n$, let $x^{(i)} = x_0...x_{i-1}(1-x_i)x_{i+1}...x_{n-1}$ be the string obtained by flipping the coordinate i.

For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we say that G_1 is a subgraph of G_2 if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. Apparently, any local optimum in G_2 is also a local optimum in G_1 (but not the other way around in general), therefore any lower bound for G_1 is also a lower bound for G_2.

We will use $v_1 \otimes v_2$ to range over the set $V_1 \times V_2$. There are various ways to define a product graph $G_1 \times G_2 = (V_1 \times V_2, E)$ by different choices of E. Three possibilities are

1. $E = \{(v_1\otimes v_2, v_1'\otimes v_2) : (v_1, v_1') \in E_1, v_2 \in V_2\} \cup \{(v_1 \otimes v_2, v_1 \otimes v_2') : (v_2, v_2') \in E_2, v_1 \in V_1\}$;
2. $E' = \{(v_1\otimes v_2, v_1'\otimes v_2') : (v_1, v_1') \in E_1 \cup I_{V_1}$ and $(v_2, v_2') \in E_2 \cup I_{V_2}\} - I_{V_1\times V_2}$, where $I_V = \{(v, v) : v \in V\}$;
3. $E'' = \{(v_1\otimes v_2, v_1'\otimes v_2') : (v_1, v_1') \in E_1 \cup I_{V_1}$ or $(v_2, v_2') \in E_2 \cup I_{V_2}\} - I_{V_1\times V_2}$.

It is clear that $E \subseteq E' \subseteq E''$, and our lower bound theorem will use the first definition E, making the theorem as general as possible.

A path X in a graph $G = (V, E)$ is a sequence $(v_1, ..., v_l)$ of vertices such that for any pair (v_i, v_{i+1}) of vertices, either $v_i = v_{i+1}$ or $(v_i, v_{i+1}) \in E$. We use $set(X)$ to denote the set of distinct vertices on path X. A path is self-avoiding if v_1, ..., v_l are all distinct. The length of a path $(v_1, ..., v_l)$ is $l-1$. For two vertices $u, v \in V$, the distance $|u-v|_G$ is the length of a shortest path from u to v. The subscript G may be omitted if no confusion is caused.

The (k,l)-hypercube $G_{k,l} = (V, E)$ where $V = [k]^l$ and whose edge set is $E = \{(u, v) : \exists i \in \{0, ..., l-1\},\ s.t.\ |u_i - v_i| = 1, \text{ and } u_j = v_j,\ \forall j \neq i\}$. Sometimes we abuse the notation by using $[k]^l$ to denote $G_{k,l}$. Note that both the Boolean hypercube and the constant dimension grid are special hypercubes.[4]

In an N-vertex graph $G = (V, E)$, a Hamilton path is a path $X = (v_1, ..., v_N)$ such that $(v_i, v_{i+1}) \in E$ for any $i \in [N-1]$ and $set(X) = V$. It is easy to check by induction that every hypercube $[k]^l$ has a Hamilton path. Actually, for $l = 1$, $[k]$ has a Hamilton path $(1, ..., k)$. Now suppose $[k]^l$ has a Hamilton path P, then a Hamilton path for $[k]^{l+1}$ can be constructed as follows. First fix the last coordinate to be 1 and go through P, then change the last coordinate to be 2 and go through P in the reverse order, and then change the last coordinate to be 3 and go through P, and so on. For each (k, l), let $HamPath_{k,l} = (v_1, ..., v_N)$ be the Hamilton path constructed as above (where $N = k^l$), and we define the successor function $H_{k,l}(v_i) = v_{i+1}$ for $i \in [N-1]$.

We use $R_2(f)$ and $Q_2(f)$ to denote the double-sided error random and quantum query complexities of function f. For more details on deterministic, randomized and quantum query models and the corresponding query complexities, we refer to [10] as an excellent survey.

2.1 One quantum adversary method and the relational adversary method

The quantum adversary method is one of the two powerful tools to prove lower bounds on quantum query complexity; see [13] for an comprehensive survey of the research area of quantum lower bounds. In this paper, we will use the quantum adversary method proposed in [23]. The definition and theorem given here are a little more general than the original ones, but the proof remains unchanged.

DEFINITION 1. *Let $F : I^N \to [M]$ be an N-variate function. Let $R \subseteq I^N \times I^N$ be a relation such that $F(x) \neq F(y)$*

[3]Another unpublished result was mentioned in [20] that Verhoeven showed $RLS([n]^2) = \Omega(n^{1-\delta})$ for any constant $\delta > 0$. But according to an author of [20], the proof was never written up and this question should be considered now to be still open.

[4]Here we identify the Boolean hypercube $\{0,1\}^n$ and $G_{2,n}$ since they are isomorphic.

for any $(x,y) \in R$. A weight scheme consists of three weight functions $w(x,y) > 0$, $u(x,y,i) > 0$ and $v(x,y,i) > 0$ satisfying $u(x,y,i)v(x,y,i) \geq w^2(x,y)$ for all $(x,y) \in R$ and $i \in [N]$ with $x_i \neq y_i$. We further put

$$w_x = \sum_{y':(x,y')\in R} w(x,y'), \qquad w_y = \sum_{x':(x',y)\in R} w(x',y),$$

$$u_{x,i} = \sum_{\substack{y':(x,y')\in R,\\ x_i\neq y'_i}} u(x,y',i), \quad v_{y,i} = \sum_{\substack{x':(x',y)\in R,\\ x'_i\neq y_i}} v(x',y,i).$$

Theorem 6. [Zhang, [23]] *For any F, R and any weight scheme w, u, v as in Definition 1, we have*

$$Q_2(F) = \Omega\left(\min_{(x,y)\in R, i\in[N]:\ x_i\neq y_i} \sqrt{\frac{w_x w_y}{u_{x,i} v_{y,i}}}\right)$$

Inspired by the quantum adversary method, Aaronson [2] gives a nice technique for proving lower bounds on randomized query complexities. We restate it using a language similar to that in Theorem 6.

Theorem 7. [Aaronson, [2]] *Let $F : I^N \to [M]$ be an N-variate function. Let $R \subseteq I^N \times I^N$ be a relation such that $F(x) \neq F(y)$ for any $(x,y) \in R$. For any weight function $w : R \to \mathbb{R}^+$, we have*

$$R_2(F) = \Omega\left(\min_{(x,y)\in R, i\in[N]:\ x_i\neq y_i} \max\left\{\frac{w_x}{w_{x,i}}, \frac{w_y}{w_{y,i}}\right\}\right)$$

where

$$w_{x,i} = \sum_{\substack{y':(x,y')\in R,\\ x_i\neq y'_i}} w(x,y'), \quad w_{y,i} = \sum_{\substack{x':(x',y)\in R,\\ x'_i\neq y_i}} w(x',y).$$

Note that we can think of Theorem 7 as having a weight scheme as in Theorem 6 too, but requiring that $u(x,y,i) = v(x,y,i) = w(x,y)$. This simple observation is used in the proofs of the lower bound theorems.

3. LOWER BOUNDS FOR LOCAL SEARCH ON GENERAL PRODUCT GRAPHS

In this section we prove a theorem which is stronger than Theorem 1 due to a relaxation on the conditions of the random walk. Suppose we are given a graph $G = (V,E)$, a starting vertex v_0 and an assignment $W : V \times \mathbb{N} \to 2^V$ *s.t.* for each $u \in V$ and $t \in \mathbb{N}$, it holds that $W(u,t) \subseteq \{u\} \cup \{v : (u,v) \in E\}$ and that $|W(u,t)| = c_t$ for some function c of t. Intuitively, W gives the candidates that the walk goes to for the next step, and the random walk (G, v_0, W) on graph G proceeds as follows. It starts at v_0, and at step $t \in \mathbb{N}$, it goes from the current vertex v_{t-1} to a uniformly random vertex in $W(v_{t-1},t)$. We say *a path $(v_0, v_1, ..., v_T)$ is generated by the random walk* if $v_t \in W(v_{t-1},t)$ for all $t \in [T]$. Denote by $p(u,t_1,v,t_2)$ the probability that the random walk is at v after step t_2 under the condition that the walk is at u after step t_1. Let $p_t = \max_{u,v,t_1,t_2:\ t_2-t_1=t} p(u,t_1,v,t_2)$. For $(u,u') \in E$, let $q(u,u',t_1,v,t_2)$ be the probability that the walk is at v after step t_2, under the conditions that 1) the walk is at u after step t_1, and 2) the walk does not go to u' at step t_1+1. The following lemma on the relation of the two probabilities is obvious.

Lemma 8. *If $|W(u,t_1+1)| > 1$, then $q(u,u',t_1,v,t_2) \leq 2p(u,t_1,v,t_2)$.*

Proof. Easy by considering the two cases of the step t_1+1 (going to u' or not). □

Theorem 9. *Suppose G contains $G^w \times G^c$ as a subgraph, and L is the length of the longest self-avoiding path in G^c. Let $T = \lfloor L/2 \rfloor$, then for any random walk (G^w, v_0^w, W) on G^w, we have*

$$RLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^T p_t}\right), \quad QLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^T \sqrt{p_t}}\right).$$

Proof. Without loss of generality, we assume $G = G^w \times G^c$, as Local Search on a subgraph is no harder than Local Search on the original graph. We shall construct a random walk on G by the random walk (G^w, v_0^w, W) on G^w and a simple one-way walk on G^c. Starting from some fixed vertex in G, the walk is proceeded by one step of walk in G^w followed by two steps of walk in G^c. (We perform *two* steps of walk in G^c mainly for some technical reasons, and this is also where the factor of 2 in the definition $T = \lfloor L/2 \rfloor$ comes from.) Precisely, fix a self-avoiding path $(z^c_{0,0}, z^c_{1,0}, z^c_{1,1}, z^c_{2,1}, z^c_{2,2}, ..., z^c_{T,T-1}, z^c_{T,T})$ of length $2T$ in G^c. Let the set P contain all the paths $X = (x_0^w \otimes z^c_{0,0}, x_1^w \otimes z^c_{0,0}, x_1^w \otimes z^c_{1,0}, x_1^w \otimes z^c_{1,1}, ..., x_T^w \otimes z^c_{T-1,T-1}, x_T^w \otimes z^c_{T,T-1}, x_T^w \otimes z^c_{T,T})$ in G such that $x_0^w = v_0^w$ and $(x_0^w, x_1^w, ..., x_T^w)$ is a path generated by the random walk (G^w, v_0^w, W). Define a problem Path_P: given a path $X \in P$, find the end point $x_T^w \otimes z^c_{T,T}$. To access X, we can ask whether $v \in set(X)$ for any vertex $v \in V$, and an oracle O will give us the Yes/No answer.[5] The following claim says that the Path_P problem is not much harder than Local Search problem.

Claim 1. $R_2(\text{Path}_P) \leq 2RLS(G), Q_2(\text{Path}_P) \leq 2QLS(G)$.

Proof. Suppose we have an Q-query randomized or quantum algorithm $\mathcal{A}$ for Local Search, we shall give a $2Q$ corresponding algorithm $\mathcal{B}$ for Path_P. For any path $X \in P$, we define a function f_X essentially in the same way as in [2, 20]: for each vertex $v \in G$, let

$$f_X(v) = \begin{cases} |v - x_0^w \otimes z^c_{0,0}|_G + 3T & \text{if } v \notin set(X) \\ 3(T-k) & \text{if } v = x_k^w \otimes z^c_{k,k} \\ 3(T-k) - 1 & \text{if } v = x_{k+1}^w \otimes z^c_{k,k} \neq x_k^w \otimes z^c_{k,k} \\ 3(T-k) - 2 & \text{if } v = x_{k+1}^w \otimes z^c_{k+1,k} \end{cases}$$

It is easy to verify that the only local minimum is $x_T^w \otimes z^c_{T,T}$.

Given an oracle O and an input X of the Path problem, $\mathcal{B}$ simulates $\mathcal{A}$ to find the local minimum of f_X, which is also the end point of X. Whenever $\mathcal{A}$ needs to make a query on v to get $f_X(v)$, $\mathcal{B}$ asks O whether $v \in set(X)$. If $v \notin set(X)$, then $f_X(v) = |v - x_0^w \otimes z^c_{0,0}|_G + 3T$; otherwise, $v = x^w \otimes z^c_{k+1,k}$ or $v = x^w \otimes z^c_{k,k}$ for some $x^w \in V^w$ and k. Note that k is known for any given vertex v. So if $v = x^w \otimes z^c_{k+1,k}$,

[5]Note that it is actually an oracle for the following function $g : V^w \times V^c \to \{0,1\}$, with $g(x) = 1$ *iff* $x \in set(X)$. So strictly speaking, an input of Path_P should be specified as $set(X)$ rather than X, because in general, it is possible that $X \neq Y$ but $set(X) = set(Y)$. For our problem, however, it is easy to verify that for any $X, Y \in P$, it holds that $X = Y \Leftrightarrow set(X) = set(Y)$. Actually, if $X \neq Y$, suppose the first diverging place is k, *i.e.* $x^w_{k-1} = y^w_{k-1}$, but $x_k^w \neq y_k^w$. Then Y will never pass $x_k^w \otimes z^c_{k,k-1}$ because the clock immediately ticks and the time always advances forward. (Or more rigorously, the only point that Y passes through $z^c_{k,k-1}$ is $y_k^w \otimes z^c_{k,k-1}$. Since $y_k^w \neq x_k^w$, $x_k^w \otimes z^c_{k,k-1} \notin set(Y)$.)

then $x^w = x^w_{k+1}$ and thus $f_X(v) = 3(T-k)-2$. Now consider the case that $v = x^w \otimes z^c_{k,k}$. If $k = 0$, then let $f_X(v) = 3T$ if $v = x^w_0 \otimes z^c_{0,0}$ and $f_X(v) = 3T-1$ otherwise. If $k \geq 1$, then $\mathcal{B}$ asks O whether $x^w \otimes z^c_{k,k-1} \in set(X)$. If yes, then $v = x^w_k \otimes z^c_{k,k}$ and thus $f_X(v) = 3(T-k)$; if no, then $v = x^w_{k+1} \otimes z^c_{k,k} \neq x^w_k \otimes z^c_{k,k}$ and thus $f_X(v) = 3(T-k)-1$. Therefore, at most 2 queries on O can simulate one query on f_X, so we have a $2Q$ algorithm for PATH$_P$ in both randomized and quantum cases. □

(Continue the proof of Theorem 9) By the claim, it is sufficient to prove lower bounds for PATH$_P$. We define a relation R_P as follows.

$R_P = \{(X,Y) \in P \times P : X$ and Y has different end points$\}$

For any pair $(X,Y) \in R_P$, where $X = (x^w_0 \otimes z^c_{0,0}, x^w_1 \otimes z^c_{0,0}, x^w_1 \otimes z^c_{1,0}, x^w_1 \otimes z^c_{1,1}, ..., x^w_T \otimes z^c_{T-1,T-1}, x^w_T \otimes z^c_{T,T-1}, x^w_T \otimes z^c_{T,T})$ and $Y = (y^w_0 \otimes z^c_{0,0}, y^w_1 \otimes z^c_{0,0}, y^w_1 \otimes z^c_{1,0}, y^w_1 \otimes z^c_{1,1}, ..., y^w_T \otimes z^c_{T-1,T-1}, y^w_T \otimes z^c_{T,T-1}, y^w_T \otimes z^c_{T,T})$, we write $X \wedge Y = k$ if $x^w_0 = y^w_0$, ..., $x^w_{k-1} = y^w_{k-1}$ but $x^w_k \neq y^w_k$. Note that if $X \wedge Y = k$, then $x^w_k, y^w_k \in W(x^w_{k-1}, k)$ and thus $|W(x^w_{k-1}, k)| \geq 2$. By Lemma 8, this implies that $q(x^w_{k-1}, x^w_k, k-1, v^w, j) \leq 2p_{j-k+1}$.

We choose the weight functions in Theorem 6 by letting

$$\begin{aligned} w(X,Y) &= 1/|\{Y' \in P : Y' \wedge X = k\}| \\ &= 1/|\{X' \in P : X' \wedge Y = k\}| \\ &= 1/[(c_k - 1)c_{k+1}...c_T]. \end{aligned}$$

To calculate $w_X = \sum_{Y':(X,Y') \in R_P} w(X,Y')$, we group those Y' that diverge from X at the same place k':

$$\begin{aligned} w_X &= \sum_{k'=1}^{T} \sum_{\substack{Y':(X,Y') \in R_P \\ X \wedge Y' = k'}} w(X,Y') \\ &= \sum_{k'=1}^{T} \sum_{\substack{Y':(X,Y') \in R_P \\ X \wedge Y' = k'}} \frac{1}{|\{Y' \in P : Y' \wedge X = k'\}|} \\ &= \sum_{k'=1}^{T} \mathbf{Pr}_{Y'}[(X,Y') \in R_P | Y' \wedge X = k'] \\ &= \sum_{k'=1}^{T} \mathbf{Pr}_{Y'}[(y')^w_T \neq x^w_T | Y' \wedge X = k'] \end{aligned}$$

The third equality holds because all paths diverging from X firstly at k' have the same probability $1/[(c_{k'} - 1)c_{k'}...c_T]$. Note that the probability in the last equality is nothing but $1 - q(x^w_{k'-1}, x^w_{k'}, k'-1, x^w_T, T)$, which is at least $1 - 2p_{T-k'+1}$. So we have

$$w_X \geq T - 2\sum_{k'=1}^{T} p_{T-k'+1} = T - 2\sum_{t=1}^{T} p_t.$$

And similarly, we have $w_Y \geq T - 2\sum_{t=1}^{T} p_t$ too.

Now we define $u(X,Y,i)$ and $v(X,Y,i)$, where i is a point $x^w_{j+r} \otimes z^c_{j+s,j} \in set(X) - set(Y)$ or $y^w_{j+r} \otimes z^c_{j+s,j} \in set(Y) - set(X)$. Here $(r,s) \in \{(0,0),(1,0),(1,1)\}$, and $0 \leq j \leq j+r \leq T$. Let

$$\begin{aligned} u(X,Y,x^w_{j+r} \otimes z^c_{j+s,j}) &= a_{k,j,r,s} w(X,Y), \\ u(X,Y,y^w_{j+r} \otimes z^c_{j+s,j}) &= b_{k,j,r,s} w(X,Y), \\ v(X,Y,x^w_{j+r} \otimes z^c_{j+s,j}) &= b_{k,j,r,s} w(X,Y), \\ v(X,Y,y^w_{j+r} \otimes z^c_{j+s,j}) &= a_{k,j,r,s} w(X,Y). \end{aligned}$$

where $a_{k,j,r,s}$ and $b_{k,j,r,s}$ will be given later (satisfying that $a_{k,j,r,s} b_{k,j,r,s} = 1$, which makes u, v, w really a weight scheme). We shall calculate $u_{X,i}$ and $v_{Y,i}$ for $i = x^w_{j+r} \otimes z^c_{j+s,j} \in set(X) - set(Y)$; the other case $i = y^w_{j+r} \otimes z^c_{j+s,j}$ is just symmetric. Note that if $x^w_{j+r} \otimes z^c_{j+s,j} \notin set(Y')$ and $X \wedge Y' = k'$, then $k' \leq j + r$.

$$\begin{aligned} u_{X,x^w_{j+r} \otimes z^c_{j+s,j}} &= \sum_{k'=1}^{j+r} \sum_{\substack{Y':(X,Y') \in R_P, X \wedge Y' = k' \\ x^w_{j+r} \otimes z^c_{j+s,j} \notin set(Y')}} a_{k',j,r,s} w(X,Y') \\ &\leq \sum_{k'=1}^{j+r} \sum_{Y':X \wedge Y' = k'} a_{k',j,r,s} w(X,Y') = \sum_{k'=1}^{j+r} a_{k',j,r,s} \end{aligned}$$

The computation for $v_{Y,x^w_{j+r} \otimes z^c_{j+s,j}}$ is a little more complicated. By definition,

$$\begin{aligned} & v_{Y,x^w_{j+r} \otimes z^c_{j+s,j}} \\ =\ & \sum_{k'=1}^{j+r} \sum_{\substack{X':(X',Y) \in R_P,\ X' \wedge Y = k', \\ x^w_{j+r} \otimes z^c_{j+s,j} \in set(X')}} b_{k',j,r,s} w(X',Y) \\ \leq\ & \sum_{k'=1}^{j+r} \sum_{\substack{X':X' \wedge Y = k', \\ x^w_{j+r} \otimes z^c_{j+s,j} \in set(X')}} b_{k',j,r,s} w(X',Y) \\ =\ & \sum_{k'=1}^{j+r} b_{k',j,r,s} \mathbf{Pr}_{X'}[x^w_{j+r} \otimes z^c_{j+s,j} \in set(X') | X' \wedge Y = k'] \end{aligned}$$

We can see that by adding the clock, the passing probability $\mathbf{Pr}_{X'}[x^w_{j+r} \otimes z^c_{j+s,j} \in set(X') | X' \wedge Y = k']$ is roughly the hitting probability $q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+r}, j) + q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+r}, j+1)$. Actually, define $Bound_{k,j,r,s} = 2p_{j-k+2} \cdot \lambda[s = 1 \vee j < T] + 2p_{j-k+1} \cdot \lambda[s = 0 \wedge (k \leq j \vee r = 0)]$, where the Boolean function $\lambda[\phi] = 1$ if ϕ is true and 0 otherwise. Then

CLAIM 2.

$$\mathbf{Pr}_{X'}[x^w_{j+r} \otimes z^c_{j+s,j} \in set(X') | X' \wedge Y = k'] \leq Bound_{k',j,r,s}.$$

PROOF. We study the probability case by case. If $s = 1$, then $r = 1$ and $x^w_{j+1} \otimes z^c_{j+1,j} \in set(X')$ if and only if $x^w_{j+1} = (x')^w_{j+1}$. So the probability is just $q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+1}, j+1)$, which is no more than $2p_{j-k'+2}$ by Lemma 8. If $s = 0$, then $x^w_{j+r} \otimes z^c_{j,j} \in set(X')$ if and only if "$x^w_{j+r} = (x')^w_j$ or $x^w_{j+r} = (x')^w_{j+1}$". Also note that

$$\mathbf{Pr}_{X'}[x^w_{j+r} = (x')^w_j | X' \wedge Y = k'] = q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+r}, j)$$

unless $k' = j+1$ and $r = 1$, in which case $\mathbf{Pr}_{X'}[x^w_{j+r} = (x')^w_j | X' \wedge Y = k'] = 0$ because $x^w_{j+1} \otimes z^c_{j,j} \notin set(Y)$ but $(x')^w_j \otimes z^c_{j,j} = y^w_j \otimes z^c_{j,j} \in set(Y)$. The other probability $\mathbf{Pr}_{X'}[x^w_{j+r} = (x')^w_{j+1} | X' \wedge Y = k']$ is just $q(y^w_{k'-1}, y^w_{k'}, k'-1, x^w_{j+r}, j+1)$ if $j \leq T-1$ and it is 0 if $j = T$. Putting all cases together, we get the desired result. □

(Continue the proof of Theorem 9) The claim implies that $v_{Y,x^w_{j+r} \otimes z^c_{j+s,j}} \leq \sum_{k'=1}^{j+r} b_{k',j,r,s} Bound_{k',j,r,s}$. The symmetric case of $u(X,Y,i)$ and $v(X,Y,i)$ where i is a point $y^w_{j+r} \otimes z^c_{j+s,j} \in set(Y) - set(X)$ can be dealt with in the same way, yielding $u_{X,y^w_{j+r} \otimes z^c_{j+s,j}} \leq \sum_{k'=1}^{j+r} b_{k',j,r,s} Bound_{k',j,r,s}$ and $v_{Y,y^w_{j+r} \otimes z^c_{j+s,j}} \leq \sum_{k'=1}^{j+r} a_{k',j,r,s}$.

By the definition of $Bound_{k',j,r,s}$, it holds that for any (j,r,s), we have $\sum_{k'=1}^{j+r} Bound_{k',j,r,s} \le 4\sum_{t=1}^{T} p_t$, and similarly $\sum_{k'=1}^{j+r} \sqrt{Bound_{k',j,r,s}} \le 4\sum_{t=1}^{T}\sqrt{p_t}$. Now for the randomized lower bound, $a_{k',j,r,s} = b_{k',j,r,s} = 1$. We get

$$\begin{aligned} & RLS(G) \\ = \; & \Omega\left(\min_{j,r,s}\max\left\{\frac{T-2\sum_{t=1}^{T}p_t}{j+r}, \frac{T-2\sum_{t=1}^{T}p_t}{\sum_{k'=1}^{j+r} Bound_{k',j,r,s}}\right\}\right) \\ = \; & \Omega\left(\frac{T}{\sum_{t=1}^{T}p_t}\right). \end{aligned}$$

For the quantum lower bound, pick $a_{k',j,r,s} = \sqrt{Bound_{k',j,r,s}}$, and $b_{k',j,r,s} = 1/\sqrt{Bound_{k',j,r,s}}$. Then

$$\begin{aligned} QLS(G) &= \Omega\left(\min_{j,r,s}\sqrt{\frac{\left(T-2\sum_{t=1}^{T}p_t\right)^2}{\left(\sum_{k'=1}^{j+r}\sqrt{Bound_{k',j,r,s}}\right)^2}}\right) \\ &= \Omega\left(\frac{T}{\sum_{t=1}^{T}\sqrt{p_t}}\right) \end{aligned}$$

This completes the proof of Theorem 9.

4. APPLICATIONS TO THE TWO SPECIAL GRAPHS

In this section, we will apply Theorem 9 to the two special graphs. Note that in both cases, the probability p_t is not easy to upper bound.

4.1 Lower bounds for Local Search on the Boolean Hypercube

To apply Theorem 9 to $\{0,1\}^n$, we decompose the whole graph into the two parts $\{0,1\}^m$ and $\{0,1\}^{n-m}$. Pick the random walk $(\{0,1\}^m, v_0^w, W)$, where $v_0^w = 0^m \in \{0,1\}^m$ and $W(x,t) = \{x^{(i)} : i \in \{0,...,m-1\}\}$ for each vertex $x = x_0...x_{m-1} \in \{0,1\}^m$ and each $t \in \mathbb{N}$. Finally, note that the longest self-avoiding path of the graph $\{0,1\}^{n-m}$ is a Hamilton path with length $L = 2^{n-m}-1$.

The following bounds on p_t are rather loose for $10 < t \le m^2$ but sufficient for our purpose.

LEMMA 10. *For any $t \in \mathbb{N}$, we have*

$$p_t = \begin{cases} O(m^{-\lceil t/2\rceil}) & \text{if } t \le 10 \\ O(m^{-5}) & \text{if } 10 < t \le m^2 \\ O(2^{-m}) & \text{if } t > m^2 \end{cases}$$

The proof of the lemma uses the generating function $(z_1 + ... + z_m)^t$ and some techniques in a Fourier analysis flavor. We leave the details in the full version [24].

PROOF. *Omitted. See the full version [24].* □

Now it is easy to prove Theorem 2 using this lemma. For the randomized lower bound, let $m = \lfloor (n+\log_2 n)/2 \rfloor$, then $T = \Theta(2^{n/2}/n^{1/2})$ and $\sum_{t=1}^{T} p_t = O(1/n)$. Thus we have $RLS(\{0,1\}^n) = \Omega(\sqrt{n}2^{n/2})$. For the quantum lower bound, let $m = \lfloor (2n+\log_2 n)/3 \rfloor$, then $T = \Theta(2^{n/3}/n^{1/3})$ and $\sum_{t=1}^{T}\sqrt{p_t} = O(1/\sqrt{n})$. Thus $QLS(\{0,1\}^n) = \Omega(2^{n/3}n^{1/6})$.

4.2 Lower bounds for Local Search on the constant dimensional grid

In this section we shall first prove a lower bound weaker than Theorem 3 in Section 4.2.1, and then improve it to Theorem 3 in Section 4.2.2.

4.2.1 A weaker family of lower bounds

As in Section 4.1, we decompose the grid into two parts, $[n]^m$ and $[n]^{d-m}$. For each vertex $x = x_0...x_{m-1} \in [n]^m$ and each $i \in \{0,...,m-1\}$, define

$$x^{(i),-} = x_0...x_{i-1}\max\{x_i - 1, 1\}x_{i+1}...x_{m-1},$$
$$x^{(i),+} = x_0...x_{i-1}\min\{x_i + 1, n\}x_{i+1}...x_{m-1}.$$

We perform the random walk $([n]^m, v_0^w, W)$ where $v_0^w = 00...0 \in [n]^m$, $W(x,t) = \{x^{((t-1) \bmod m),+}, x^{((t-1) \bmod m),-}\}$. To analyze the probability p_t in Theorem 9, we first consider the following simpler "line walk". Suppose a particle is initially put at point $i \in \{1,...,n\}$, and in each step the particle moves either to $\max\{1, i-1\}$ or to $\min\{n, i+1\}$, each with probability 1/2. Let $p_{ij}^{(t)}$ denote the probability that the particle starting from point i is at point j after exact t steps of the walk. For $t \ge 1$, the following proposition gives a very good (actually tight) estimate on $\max_{ij} p_{ij}^{(t)}$.

PROPOSITION 11. *For any $t \ge 1$,*

$$\max_{i,j} p_{ij}^{(t)} = \begin{cases} O(1/\sqrt{t}) & \text{if } t \le n^2 \\ O(1/n) & \text{if } t > n^2 \end{cases}$$

If there are not the two barriers (1 and n) then $p_{ij}^{(t)}$ is very easy to calculate: $p_{ij}^{(t)} = \binom{t}{t/2+(j-i)/2}$ if $j-i$ and t have the same parity, and 0 otherwise. However, since we now have the two barriers, it is hard to count the number of paths from i to j after exactly t steps. Here inspired by the so-called *reflecting rule*, we will construct a series of 1-1 correspondences to reduce the problem step by step to the no-barrier case. For more details please see the full version of the paper [24].

PROOF. *Omitted. See the full version [24].* □

Now we use Proposition 11 to prove the weaker lower bounds for grids. Since the random walk $([n]^m, v_0^w, W)$ is just a product of m line walks, it is not hard to see that the p_t in the random walk $([n]^m, v_0^w, W)$ is equal to $O(1/\sqrt{t^m})$ if $t \le n^2$, and $O(1/n^m)$ if $t > n^2$. Now for the randomized lower bounds, when $d > 4$ we pick $m = \lceil d/2 \rceil > 2$ and we get $RLS([n]^d) = \Omega\left(\frac{n^{d-m}}{O(1)+n^{d-m}/n^m}\right) = \Omega(n^{\lfloor d/2 \rfloor})$, which is $\Omega(n^{\frac{d}{2}})$ if d is odd, and $\Omega(n^{\frac{d}{2}-\frac{1}{2}})$ if d is even. For $d = 4,3,2$, we let $m = 2,2,1$ respectively, and get $RLS([n]^4) = \Omega(n^2/(\log n + 1)) = \Omega(n^2/\log n)$, $RLS([n]^3) = \Omega(n/(\log n + 1/n)) = \Omega(n/\log n)$, and $RLS([n]^2) = \Omega(n/(\sqrt{n}+1)) = \Omega(\sqrt{n})$.

For the quantum lower bounds, if $d > 6$, we let m be the integer closest to $2d/3$, thus $m > 4$. We get $QLS([n]^d) = \Omega\left(\frac{n^{d-m}}{O(1)+n^{d-m}/n^{m/2}}\right)$, which is $\Omega(N^{\frac{1}{3}})$ if $d = 3d'$, $\Omega(N^{\frac{1}{3}-\frac{1}{3d}})$ if $d = 3d'+1$, and $\Omega(N^{\frac{1}{3}-\frac{1}{6d}})$ if $d = 3d'+2$. For $d = 6$, let $m = 4$ and we have $QLS([n]^6) = \Omega(n^2/\log n)$. For $d = 5,4,3$, let $m = d-2$ then $QLS([n]^d) = \Omega(n^2/(n^{2-(d-2)/2} + n^{2-(d-2)/2})) = \Omega(n^{d/2-1})$, which is $\Omega(n^{5/2}), \Omega(n^2), \Omega(n^{3/2})$, respectively. For $d = 2$, let $m = 1$ and $QLS([n]^2) = \Omega(\frac{n}{n^{3/4}}) = \Omega(n^{1/4})$.

4.2.2 *Improvement*

One weakness of the above proof is the integer constraint of the dimension m. We now show a way to get around the problem, allowing m to be any real number between 0 and $d-1$. The idea is to partition the grid into many blocks, with different blocks representing different time slots, and the blocks are threaded into one very long block by many paths that are pairwise disjoint. Roughly speaking, we view $[n]^d$ as the product of d line graph $[n]$. For each of the first $d-1$ line graphs, we cut it into n^{1-r} parts evenly, each of size n^r. (Here $r = m/(d-1)$). Then $[n]^{d-1}$ is partitioned into $n^{(d-1)(1-r)}$ smaller grids, all isomorphic to $[n^r]^{d-1}$. Putting the last dimension back, we have $n^{(d-1)(1-r)}$ *blocks*, all isomorphic to $[n^r]^{d-1} \times [n]$. Now the random walk will begin in the first block, and within each block, it is just one step of random walk in $[n^r]^{d-1}$ followed by two steps of one-way walk in the last dimension space $[n]$. When the walk runs out of the clock $[n]$, the walk will move to the next block via a particular block-changing path. All block-changing paths are carefully designed to be disjoint, and they "thread" all the blocks to form a $[n^r]^{d-1} \times [L]$ grid, where $L = (n - 2n^r)n^{(1-r)(d-1)}$. ($L$ is not $n \cdot n^{(1-r)(d-1)}$ because we need $2n^r$ points for the block-changing paths.)

We now describe the partition and the walk precisely. For $x = x_0...x_{d-1}$ in $[n]^d$, let $x^{(k)=l} = x_0...x_{k-1}lx_{k+1}...x_{d-1}$, and $x^{(k)=(k)+i} = x_0...x_{k-1}(x_k + i)x_{k+1}...x_{d-1}$, where i satisfies $x_k + i \in [n]$. Recall that $x^{(i),-} = x^{(i)=\max\{x_i-1,1\}}$ and $x^{(i),+} = x^{(i)=\min\{x_i+1,n\}}$.

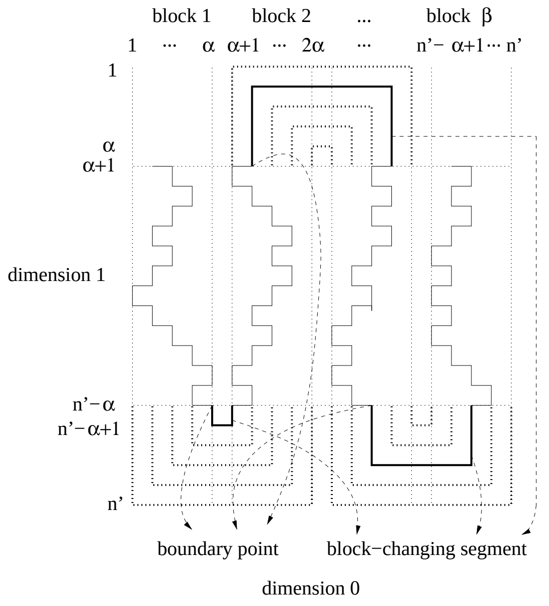

Figure 1: Illustration for changing a block in 2 dimensional grid

For any fixed constant $r \in (0,1)$, let $\alpha = \lfloor n^r \rfloor$, $\beta = \lfloor n^{1-r} \rfloor$ and $n' = \alpha\beta$. Note that $n' \geq (n^r - 1)(n^{1-r} - 1) = n - o(n)$. We now consider the slightly smaller grid $[n']^d$. Let V_1 be the set $[n']^{d-1} = \{x_0...x_{d-2} : x_i \in [n']\}$. We cut V_1 into β^{d-1} parts $\{x_0...x_{d-2} : (k_i - 1)\alpha < x_i \leq k_i\alpha\}_{k_0...k_{d-2} \in [\beta]^{d-1}}$, each of which is a small grid isomorphic to $[\alpha]^{d-1}$. We then refer to the set $\{x_0...x_{d-2}x_{d-1} : (k_i - 1)\alpha < x_i \leq k_i\alpha, i = 0, ..., d-2, \alpha < x_{d-1} \leq n' - \alpha\}$ as the "*block* $(k_0, ..., k_{d-2})$". Note that $(k_0, ..., k_{d-2})$ can be also viewed as a point in grid $[\beta]^{d-1}$, and there is a Hamilton path $HamPath_{\beta,d-1}$ in $[\beta]^{d-1}$, as defined in Section 2. We call the block $(k'_0, ..., k'_{d-2})$ *the next block* of the block $(k_0, ..., k_{d-2})$ if $(k'_0, ..., k'_{d-2})$, viewed as the point in $[\beta]^{d-1}$, is the next point of $(k_0, ..., k_{d-2})$ in $HamPath_{\beta,d-1}$. Note that by our definition of $HamPath_{\beta,d-1}$, we know that $\exists i \in \{0, ..., d-2\}$ *s.t.* $k'_i \in \{k_i + 1, k_i - 1\}$ and for all other $j \neq i$, $k'_j = k_j$. That is, adjacent blocks have only one coordinate to be different, and this difference is 1. We call the the block $(k_0, ..., k_{d-2})$ *the last block* if $(k_0, ..., k_{d-2})$ is the last point in $HamPath_{\beta,d-1}$.

Now we define the random walk by describing how a particle may go from start to end. The path set is just all the possible paths the particle goes along. Intuitively, within one block, the last dimension $d-1$ serves as the clock space. So as before, we perform one step of line walk (in the dimension which is the circularly next dimension of the last one that the walk just goes in), followed by two steps of walk in the clock space. If we run out of clock, we say we reach *a boundary point* at the current block, and we move to the next block via a path segment called *block-changing segment*. In what follows, we specify how the particle may move during the whole random walk process, including going through block-changing segments. We always use $x_0...x_{d-1}$ to denote the current position of the particle, and assume $x_i = (k_i - 1)\alpha + y_i$, *i.e.* x is in the block $(k_0, ..., k_{d-2})$ with the offsets $(y_0, ..., y_{d-1})$. Thus the instruction $x_0 = x_0 + 1$, for example, means that the particle moves from $x_0...x_{d-1}$ to $(x_0 + 1)x_1...x_{d-1}$.

```
1. $x_0 = ... = x_{d-2} = 0$, $x_{d-1} = \alpha+1$, $k_0 = ... = k_{d-2} = 1$.
2. for $t = 1$ to $(n' - 2\alpha)\beta^{d-1}$,
     Let $t' = \lfloor \frac{t-1}{n'-2\alpha} \rfloor$, $i = (t-1) \bmod (d-1)$
     do either $x_i = \max\{x_i - 1, (k_i - 1)\alpha + 1\}$ or $x_i = \min\{x_i + 1, k_i\alpha\}$ randomly
     if $t \neq k(n' - 2\alpha)$ for some positive integer $k$,
       do $x_{d-1} = x_{d-1} + (-1)^{t'}$ twice
     else   (the particle is now at a boundary point)
       if the particle is not in the last block
       (Suppose the current block changes to the next block by increasing $k_j$ by $b \in \{-1, 1\}$)
         do $x_{d-1} = x_{d-1} + (-1)^{t'}$ for $(\alpha + 1 - y_j)$ times
         do $x_j = x_j + b$ for $2(\alpha + 1 - y_j) - 1$ times
         do $x_{d-1} = x_{d-1} + (-1)^{t'+1}$ for $(\alpha + 1 - y_j)$ times
         $k_j = k_j + b$
       else
         The particle stops and the random walk ends
```

It is easy to verify that every boundary point has one unique block-changing segment, and different block-changing segments do not intersect. Thus the block-changing segments thread all the blocks to form a $[\alpha]^{d-1} \times [L]$ grid, where $L = (n' - 2\alpha)\beta^{d-1}$.

CLAIM 3. *we can use Theorem 9 with $G^w = [n^r]^{d-1}$ and $G^c = [L]$ to prove lower bounds on $RLS([n]^d)$ and $QLS([n]^d)$.*

PROOF. *Omitted. See the full version [24].* □

Now $T = \lfloor L/2 \rfloor$ and $p_t = O(1/\sqrt{t^{d-1}})$ for $t \leq n^{2r}$ and $p_t = O(1/n^{r(d-1)})$ for $t > n^{2r}$. For the randomized lower bounds, if $d \geq 4$, then let $r = d/(2d-2)$ and we get $RLS([n]^d) = \Omega(L/(\sum_{t=1}^{n^{d/(d-1)}} 1/\sqrt{t^{d-1}} + L/n^{d/2}))) = \Omega(n^{d/2})$. If $d = 3$, let $r = 3/4 - \log\log n/(4\log n)$, then we have $RLS([n]^3) = \Omega((n^3/\log n)^{1/2})$. For $d = 2$, let $r = 2/3$ and we get $RLS([n]^2) = \Omega(n^{2/3})$.

For the quantum lower bounds, if $d \geq 6$, let $r = 2d/(3d-3)$ and we get $QLS([n]^d) = \Omega(n^{d/3})$. If $d = 5$, then let $r = 5/6 - \log\log n/(6\log n)$ and $QLS([n]^5) = \Omega((n^5/\log n)^{1/3})$. For $2 \leq d \leq 4$, we let $r = d/(d+1)$, then $QLS([n]^d) = \Omega(n^{d/2-d/(d+1)})$.

4.2.3 Further improvement on the 2-dimensional grid

Some other random walk can be used to further improve the lower bound on low dimension grid cases. For example, we can cut the 2-dimensional grid into $n^{2/5}$ blocks (each of size $n^{4/5} \times n^{4/5}$), and let blocks to serve as the clock. Using a random walk similar to Aaronson's in [2] (with some modifications to move to the next block after each step), we can prove [6]

CLAIM 4. *$QLS([n]^2) = \Omega(n^{2/5})$.*

PROOF. *Omitted. See the full version [24].* □

This completes the proof of Theorem 3.

5. NEW ALGORITHMS FOR LOCAL SEARCH ON GENERAL GRAPHS

In [4, 2], a randomized and a quantum algorithm for Local Search on general graphs are given as follows. Do a random sampling over all the vertices, find a vertex v in them with the minimum f-value. (For the minimum f-value finding procedure, The randomized algorithm in [4] just queries all these vertices and find the minimum, while the quantum algorithm in [2] uses the algorithm by Durr and Hoyer [11] based on Grover search [12] to get a quadratic speedup.) If v is a local minimum, then return v; otherwise we follow a *decreasing path* as follows. Find a neighbor of v with the minimum f-value, and continue this minimum-value-neighbor search process until getting to a local minimum. We can see that the algorithms actually fall into the generic algorithm category (see Section 1), with the initial point picked as the best one over some random samples.

In this section, we give new randomized and quantum algorithms, which work better than this simple "random sampling + steepest descent" method when the graph expands slowly. Here the idea is that after finding the minimum vertex v of the sampled points, instead of following the decreasing path of v, we start over within a smaller range, which contains those vertices "close to" v. If this smaller range contains a local minimum for sure, then we can simply search a local minimum in it and do this procedure recursively. But one caveat here is that a straightforward recursion does not work, because a local minimum u in the smaller range may be not a local minimum in the original larger graph G (since u may have more neighbors in G). So we shall find a small range which has a "good" boundary in the sense that all vertices on the boundary have a large f-value.

[6]Note that this walk suffers from the fact that the "passing probability" is now $n^{4/5}$ times the "hitting probability". So it only it only gives better results for $QLS([n]^2)$.

Now we describe the algorithm precisely, with some notations as follows. For $G = (V, E)$, a given function $f : V \to \mathbb{N}$, a vertex $v \in V$ and a set $S \subseteq V$, let $n(v, S) = |\{u \in S : f(u) < f(v)\}|$. The boundary $B(S)$ of the set $S \subseteq V$ is defined by $B(S) = \{u \in S : \exists v \in V - S \ s.t. \ (u, v) \in E\}$. In particular, $B(V) = \emptyset$. A decreasing path from a vertex $v \in V$ is a sequence of vertices $v_0, v_1, ..., v_k$ such that $v_0 = v$, v_k is a local minimum and $f(v_{i+1}) = \min_{v:(v_i,v)\in E} f(v) < f(v_i)$ for $i = 0, ..., k-1$. We write $f(u) \leq f(S)$ if $f(u) \leq f(v)$ for all $v \in S$. In particular, it always holds that $f(u) \leq f(\emptyset)$. Suppose $d = \max_{u,v\in V} |u - v|$ is the diameter of the graph, and $\delta = \max_{v\in V} |\{u : (u, v) \in E\}|$ is the max degree of the graph. In the following algorithm, the asymptotical numbers at the end of some command lines are the numbers of randomized or quantum queries needed for the step. For those commands without any number, no query is needed.

```
1. $m_0 = d$, $U_0 = V$;
2. $i = 0$;
3. **while** ($|m_i| > 10$) **do**
   (a) Randomly pick (with replacement) $\lceil \frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1} \rceil$ vertices from $U_i$, where $\epsilon_1 = 1/(10\log_2 d)$;
   (b) Search the sampled vertices for one $v_i$ with the minimal $f$ value.
       - Randomized algorithm: query all the sampled vertices and get $v_i$.   — $O\left(\frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1}\right)$
       - Quantum algorithm: use Durr and Hoyer's algorithm [11] with the error probability at most $\epsilon_2 = 1/(10\log_2 d)$.   — $O\left(\sqrt{\frac{8|U_i|}{m_i}} \log \frac{1}{\epsilon_1} \log \frac{1}{\epsilon_2}\right)$
   (c) **if** $i = 0$, **then** $u_{i+1} = v_i$;
       **else if** $f(u_i) \leq f(v_i)$, **then** $u_{i+1} = u_i$;
           **else** $u_{i+1} = v_i$;
   (d) **for** $j = 1, 2, ...$
       i. Randomly pick $m_{ij} \in M_i = \{m : m_i/8 \leq m \leq m_i/2, \ |W(m)| \leq 10|U_i|/m_i\}$, where $W(m) = \{w \in U_i : |w - u_{i+1}| = m\}$. Let $W_{ij} = W(m_{ij})$.
       ii. Test whether $f(u_{i+1}) \leq f(W_{ij})$
           - Randomized algorithm: query all vertices in $W_{ij}$.   — $O(|W_{ij}|)$
           - Quantum algorithm: use Durr and Hoyer's algorithm [11] on $W_{ij}$ with the error probability at most $\epsilon_3 = 1/(200\log_2 d)$.   — $O\left(\sqrt{|W_{ij}|} \log \frac{1}{\epsilon_3}\right)$
       iii. If the answer is Yes, jump out of this **for** loop and go to Step 3e.
   (e) $J_i = j$, $m_{i+1} = m_{ij}$, $W_i = W_{ij}$, $U_{i+1} = \{u \in U_i : |u - u_{i+1}| \leq m_{i+1}\}$;
   (f) $i = i + 1$;
4. $I = i$;
```

5. Follow a decreasing path of u_I to find a local minimum.

   - Randomized algorithm: in each step, query all the neighbors — $O(\delta)$

   - Quantum algorithm: in each step, use Durr and Hoyer's algorithm with the error probability at most $1/100$ — $O(\sqrt{\delta})$

Define $c(k) = \max_{v\in V} |\{u : |u-v| \le k\}|$. Apparently, the expanding speed of a graph is upper bounded by $c(k)$. The following theorem says that the algorithm is efficient if $c(k)$ is small.

THEOREM 12. *The algorithm outputs a local minimum with probability at least 1/2. The randomized algorithm uses $O\left(\sum_{i=0}^{I-1} \frac{c(m_i)}{m_i} \log\log d\right)$ queries in expectation, and the quantum algorithm uses $O\left(\sum_{i=0}^{I-1} \sqrt{\frac{c(m_i)}{m_i}}(\log\log d)^{1.5}\right)$ queries in expectation.*

In case that $c(k) = O(k^\alpha)$ for some $\alpha \ge 1$ and $k = 1, ..., d$, the expected number of queries that the randomized algorithm uses is $O\left(\frac{d^{\alpha-1}-1}{1-2^{1-\alpha}} \log\log d\right)$ if $\alpha > 1$ and $O(\log d \log\log d)$ if $\alpha = 1$. The expected number of queries that the quantum algorithm use is $O\left(\frac{d^{\frac{\alpha-1}{2}}-1}{1-2^{\frac{1-\alpha}{2}}}(\log\log d)^{1.5}\right)$ if $\alpha > 1$ and $O(\log d \log logd)$ if $\alpha = 1$.

Several comments before proving the theorem:

1. $\lim_{\alpha\to 1} \frac{d^{\alpha-1}-1}{1-2^{1-\alpha}} = \lim_{\alpha\to 1} \frac{d^{\frac{\alpha-1}{2}}-1}{1-2^{\frac{1-\alpha}{2}}} = \log_2 d$

2. If $\alpha - 1 \ge \epsilon$ for some constant $\epsilon > 0$, then $\frac{d^{\alpha-1}-1}{1-2^{1-\alpha}} = \Theta(d^{\alpha-1})$ and $\frac{d^{\frac{\alpha-1}{2}}-1}{1-2^{\frac{1-\alpha}{2}}} = \Theta(d^{(\alpha-1)/2})$.

   If further the bound $c(k) = O(k^\alpha)$ is tight in the sense that $N = c(d) = \Theta(d^\alpha)$, then $RLS(G) = O\left(\frac{N}{d}\log\log d\right)$ and $QLS(G) = O(\sqrt{\frac{N}{d}}(\log\log d)^{1.5})$.

3. For 2-dimensional grid, $d = \Theta(n)$ and $\alpha = 2$. Thus Theorem 5 follows immediately.

PROOF. We shall prove the theorem for the quantum algorithm. The analysis of the randomized algorithm is almost the same (and actually simpler). We say W_i is *good* if $f(u_{i+1}) \le f(W_i)$. We shall first prove the following claim, then the theorem follows easily.

CLAIM 5. *For each $i = 0, 1, ..., I-1$, the following three statements hold.*

1. *$n(u_{i+1}, U_{i+1}) \le n(u_{i+1}, U_i) \le m_i/8 \le m_{i+1}$ with probability $1 - \epsilon_1 - \epsilon_2$.*

2. *If $n(u_{i+1}, U_i) \le m_i/8$, then W_i is good with probability $1 - \epsilon_3 J_i$, and $\mathbf{E}[J_i] \le 2$.*

3. *If $W_0, ..., W_i$ are all good, then $f(u_{i+1}) \le f(B(U_{i+1}))$, and $u_{i+1} \notin B(U_{i+1})$.*

PROOF. 1: In Step 3a - 3c, denote by S the set of the $\lceil \frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1} \rceil$ sampled vertices in Step 3a. Let $a = \min_{u\in S} f(u)$, then $|\{v \in U_i : f(v) < a\}| \le m_i/8$ with probability at least $1-\epsilon_1$. The v_i found in Step 3b achieves the minimum in the definition of a with probability at least $1 - \epsilon_2$. Put the two things together, we have $n(v_i, U_i) \le m_i/8$ with probability at least $1 - \epsilon_1 - \epsilon_2$. Since $f(u_{i+1}) \le f(v_i)$ (by Step 3c), $U_{i+1} \subseteq U_i$ (by Step 3e) and $m_{i+1} \ge m_i/8$ (by Step 3(d)i), we have $n(u_{i+1}, U_{i+1}) \le n(u_{i+1}, U_i) \le n(v_i, U_i) \le m_i/8 \le m_{i+1}$ with probability at least $1 - \epsilon_1 - \epsilon_2$.

2: We say an m_{ij} is good if the corresponding W_{ij} is good, *i.e.* $f(u_{i+1}) \le f(W_{ij})$. Note that for any $m_{ij} \in [m_i]$, we have $W_{ij} \subseteq U_i$, and also have $W_{ij} \cap W_{ij'} = \emptyset$ if $m_{ij} \ne m_{ij'}$. Therefore, if $n(u_{i+1}, U_i) \le m_i/8$, then at most $m_i/8$ distinct m_{ij}'s in $[m_i]$ are *not* good. Also note that the number of distinct m_{ij}'s *s.t.* $|W(m_{ij})| > 10|U_i|/m_i$ is less than $m_i/10$. Therefore, $|M_i| \ge (\frac{3}{8} - \frac{1}{10})m_i > m_i/4$. So if $n(u_{i+1}, U_i) \le m_i/8$, a random m_{ij} in M_i is good with probability at least 1/2, and thus $\mathbf{E}[J_i] \le 2$. Also the probability that all the Grover searches in Step 3(d)ii are correct is at least $1 - J_i\epsilon_3$.

3: We shall first prove $B(U_{i+1}) \subseteq B(U_i) \cup W_i$. In fact, any $s \in B(U_{i+1})$ satisfies that $s \in U_{i+1}$ and that $\exists t \in V - U_{i+1}$ *s.t.* $|s - t| = 1$. Recall that $U_{i+1} \subseteq U_i$, so if $t \in V - U_i$, then $s \in B(U_i)$ by definition. Otherwise $t \in U_i - U_{i+1}$, and thus $t \in U_i$ and $|t - u_{i+1}| > m_{i+1}$ by the definition of U_{i+1}. Noting that $|s - u_{i+1}| \le m_{i+1}$ since $s \in U_{i+1}$, and that $|s - t| = 1$, we have $|s - u_{i+1}| = m_{i+1}$, which means $s \in W_i$. Thus for all $s \in B(U_{i+1})$, either $s \in B(U_i)$ or $s \in W_i$ holds, which implies $B(U_{i+1}) \subseteq B(U_i) \cup W_i$.

Applying the result recursively, we have $B(U_{i+1}) \subseteq B(U_0) \cup W_0 \cup ... \cup W_i = W_0 \cup ... \cup W_i$. Since we have $f(u_{i+1}) \le f(u_i) \le ... \le f(u_1)$ (by Step 3c) and $f(u_{k+1}) \le f(W_k)$ (for $k = 0, ..., i$) by the assumption that all W_k's are good, we know that $f(u_{i+1}) \le f(W_0 \cup ... \cup W_i)$, which implies $f(u_{i+1}) \le f(B(U_{i+1}))$.

For the other goal $u_{i+1} \notin B(U_{i+1})$, it is sufficient to prove $u_{i+1} \notin B(U_i)$ and $u_{i+1} \notin W_i$. The latter is easy to see by the definition of W_i. For the former, we can actually prove $u_{k+1} \notin B(U_k)$ for all $k = 0, ..., i$ by induction on k. The base case of $k = 0$ is trivial because $B(U_0) = \emptyset$. Now suppose $u_k \notin B(U_{k-1})$. There are two cases of u_{k+1} by Step 3c. If $f(u_k) \le f(v_k)$, then $u_{k+1} = u_k \notin B(U_{k-1})$ by induction. Again by the definition of W_{k-1} we know that $u_k \notin W_{k-1}$ and thus $u_{k+1} = u_k \notin B(U_k)$. The other case is $f(u_k) > f(v_k)$, then $u_{k+1} = v_k$, and therefore $f(u_{k+1}) = f(v_k) < f(u_k) \le f(B(U_k))$ (by the first part in 3), which implies that $u_{k+1} \notin B(U_k)$. □

(Continue the proof of Theorem 12) Now by the claim, we know that with probability at least $1 - I(\epsilon_1 + \epsilon_2) - \sum_{i=0}^{I-1} J_i\epsilon_3$, we will have that

$$n(u_I, U_I) \le m_I, \qquad f(u_I) \le f(B(U_I)), \qquad u_I \notin B(U_I).$$

Note that the correctness of the algorithms follows from these three items. Actually, by the last two items, we know that any decreasing path from u_I is contained in U_I. Otherwise suppose $(u_I^0, u_I^1, ..., u_I^T)$ is a decreasing path from u_I (so $u_I^0 = u_I$), and the first vertex out of U_I is u_I^t, then $u_I^{t-1} \in B(U_I)$. Since $u_I^0 \notin B(U_I)$, we have $t - 1 > 0$ and thus $f(u_I^{t-1}) < f(u_I)$, contradicting to $f(u_I) \le f(B(U_I))$. Now together with the first item, we know that any decreasing path from u_I is no more than m_I long. Thus Step 5 will find a local minimum by following a decreasing path.

The error probability of the algorithm is $I(\epsilon_1 + \epsilon_2) + J\epsilon_3 + 10/100$, where $J = \sum_{i=0}^{I-1} J_i$. Since $\mathbf{E}[J] = 2I$, we know by Markov inequality that with $J < 20I$ with probability at least 9/10. Since $\epsilon_1 = \epsilon_2 = 1/(10\log_2 d)$ and

$\epsilon_3 = 1/(200 \log_2 d)$, and note that $I \le \log_2 d$ because $m_0 = d$ and $m_{i+1} \le \lceil m_i/2 \rceil$. So the total error probability is less than 1/2.

We now consider the number of queries used in the i-th iteration. Note from Step 1 and Step 3e that $|U_i| \le c(m_i)$ for $i = 0, 1, ..., I-1$. So Step 3b uses

$$O\left(\sqrt{\frac{8|U_i|}{m_i}\log\log d}\log\log d\right) = O\left(\sqrt{\frac{c(m_i)}{m_i}}(\log\log d)^{1.5}\right)$$

queries. Also note from Step 3(d)i that $|W_{ij}| \le 10|U_i|/m_i$, so Step 3d uses $O(\sum_{j=1}^{J_i}\sqrt{c(m_i)/m_i}\log\log d)$ queries, which has the expectation of $O(\sqrt{c(m_i)/m_i}\log\log d)$. Finally, Step 5 uses $O(\sqrt{\delta})$ queries. Note that $\delta = c(1) = O(c(m_I)/m_I)$ where m_I is a constant integer in the range $[6, 10]$. Altogether, the total expected number of queries used is

$$O\left(\left(\sum_{i=0}^{\log_2 d-1}\sqrt{c(m_i)/m_i}\right)(\log\log d)^{1.5}\right).$$

If $c(k) = O(k^\alpha)$ for some $\alpha \ge 1$ and $k = 1, ..., d$, then

$$\sum_{i=0}^{\log_2 d-1}\sqrt{\frac{c(m_i)}{m_i}} = \sum_{i=0}^{\log_2 d-1} m_i^{(\alpha-1)/2}$$
$$= \sum_{i=0}^{\log_2 d-1}(d/2^i)^{(\alpha-1)/2} = \frac{d^\beta - 1}{1-2^{-\beta}}$$

where $\beta = (\alpha-1)/2$. This completes the proof for the quantum algorithm, except that in the case of $\alpha = 1$, where we have a quantum upper bound of $O(\log d(\log\log d)^{1.5})$. But this is worse than the randomized algorithm, which uses $O(\log d \log\log d)$ queries. So when $\alpha = 1$, the quantum algorithm just uses the randomized one. □

Acknowledgement

The author thanks Scott Aaronson, Xiaoming Sun and Andy Yao very much for many valuable discussions. Thanks also to Yves Verhoeven and Dirk Winkler for each pointing out an error in a preliminary version of the paper.

6. REFERENCES

[1] K. Aardal, S. Hoesel, J.K. Lenstra, L. Stougie. A Decade of Combinatorial Optimization. CWI Tracts 122, pp. 5-14, 1997.

[2] S. Aaronson. Lower Bounds for Local Search by Quantum Arguments. Proceedings of the thirty-sixth Annual ACM Symposium on Theory of Computing, pp. 465-474, 2004.

[3] E. Aarts and J. Lenstra, John Wiley & Sons, Inc. New York, NY, USA, 1997.

[4] D. Aldous. Minimization Algorithms and Random Walk on the d-Cube. Annals of Probability, 11(2), pp. 403-413, 1983.

[5] I. Althofer and K. Koschnich. On the Deterministic Complexity of Searching Local Maxima. Discrete Applied Mathematics 43, pp. 111-113, 1993.

[6] A. Ambainis. Polynomial Degree vs. Quantum Query Complexity. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 230-239, 2003.

[7] A. Ambainis. Quantum Lower Bounds by Quantum Arguments, Journal of Computer and System Sciences, 64, pp. 750-767, 2002.

[8] H. Barnum, M. Saks, M. Szegedy. Quantum Query Complexity and Semidefinite Programming. Proceedings of the 18th Annual IEEE Conference on Computational Complexity, pp. 179-193, 2003.

[9] R. Beals, H. Buhrman, R. Cleve, M.Mosca, R. de Wolf. Quantum Lower Bounds by Polynomials. Journal of ACM, 48, pp. 778-797, 2001.

[10] H. Buhrman and R. de Wolf. Complexity Measures and Decision Tree Complexity: a survey. Theoretical Computer Science, 288(1), pp. 21-43, 2002.

[11] C. Durr and P. Hoyer. A Quantum Algorithm for Finding the Minimum, quant-ph/9607014, 1996.

[12] L. Grover. A Fast Quantum Mechanical Algorithm for Database Search, Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, pp. 212-219, 1996.

[13] P. Hoyer and R. Spalek. Lower Bounds on Quantum Query Complexity, Bulletin of the European Association for Theoretical Computer Science 87, pp. 78-103, 2005.

[14] D. Johnson, C. Papadimitriou, and M. Yannakakis. How Easy is Local Search, Journal of Computer and System Sciences 37, pp. 429-448, 1988.

[15] S. Laplante and F. Magniez. Lower Bounds for Randomized and Quantum Query Complexity Using Kolmogorov Arguments. Proceedings of the 19th Annual IEEE Conference on Computational Complexity, pp. 294-304, 2004.

[16] D. Llewellyn and C. Tovey. Dividing and Conquering the Square. Discrete Applied Mathematics 43, pp. 131-153, 1993.

[17] D. Llewellyn, C. Tovey, M. Trick. Local Optimization on Graphs. Discrete Aplied Mathematics 23, pp. 157-178, 1989. Erratum: 46, pp. 93-94, 1993.

[18] N. Megiddo and C. Papadimitriou. On Total Functions, Existence Theorems, and Computational Complexity. Theoretical Computer Science 81, pp. 317-324, 1991.

[19] J. Orlin, A. Punnen, A. Schulz. Approximate Local Search in Combinatorial Optimization. SIAM Journal on Computing, 33(5), pp. 1201-1214, 2004.

[20] M. Santha and M. Szegedy. Quantum and Classical Query Complexities of Local Search Are Polynomially Related. Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pp.494-501, 2004.

[21] R. Spalek and M. Szegedy. All Quantum Adversary Methods Are Equivalent. Proceedings of 32nd International Colloquium on Automata, Languages and Programming, pp. 1299-1311, LNCS 3580, Lisboa, Portugal, 2005.

[22] Y. Verhoeven, Enhanced Algorithms for Local Search. Information Processing Letters 97, 171-176, 2006.

[23] S. Zhang. On the Power of Ambainis Lower Bounds, Theoretical Computer Science, 339(2-3), pp. 241-256, 2005.

[24] S. Zhang. New Upper and Lower Bounds for Randomized and Quantum Local Search, quant-ph/0603034, 2006.