SIAM J. COMPUT.
Vol. 39, No. 3, pp. 948–977

TIGHT BOUNDS FOR RANDOMIZED AND QUANTUM LOCAL SEARCH*

SHENGYU ZHANG†

Abstract. The problem Local Search, 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 both the randomized and the 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: $RLS(\{0,1\}^n) = \Theta(2^{n/2}n^{1/2})$, $QLS(\{0,1\}^n) = \Theta(2^{n/3}n^{1/6})$, $RLS([n]^d) = \Theta(n^{d/2})$ for $d \geq 4$, $QLS([n]^d) = \Theta(n^{d/3})$ for $d \geq 6$. 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 [in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, 2004, pp. 465–474], Ambainis (unpublished), and Santha and Szegedy [in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, 2004, pp. 494–501]. 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 previously best known upper bound of $O(n^{2/3})$ (see Aaronson [in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, 2004, pp. 465–474]), and implies that Local Search on grids exhibits different properties in low dimensions.

Key words. optimization, local search, quantum algorithm, quantum query complexity, randomized algorithm, randomized decision tree complexity

AMS subject classifications. 68Q17, 68W20, 68Q10, 68Q25

DOI. 10.1137/06066775X

1. Introduction.

Many important combinatorial optimization problems arising in both theory and practice are **NP**-hard, which forces one to resort to heuristic searches in practice. One popular approach is the local search, by which one first defines a *neighborhood structure* and then finds a solution that is locally optimal with respect to this neighborhood structure. In the past two decades, the local search approach has been extensively developed and "has reinforced its position as a standard approach in combinatorial optimization" in practice [1]. We use Local Search to denote the problem of finding a locally optimal point. Besides the practical applications, the problem also has many connections to 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 study of Local Search. Local Search is also related to

*Received by the editors August 21, 2006; accepted for publication (in revised form) June 2, 2009; published electronically September 2, 2009. This research was mainly done when the author was at Princeton University supported in part by NSF grants CCR-0310466 and CCF-0426582. A preliminary version of this paper appeared in *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, 2006, pp. 634–643.

http://www.siam.org/journals/sicomp/39-3/66775.html

†Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong (syzhang@cse.cuhk.edu.hk).

[1]Polynomial Local Search, introduced by Johnson, Papadimitriou, and Yannakakis [14].

[2]The family of total function problems, introduced by Megiddo and Papadimitriou [18].

physical systems including folding proteins and to 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 works 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 theoretical studies, the query model has drawn much attention [2, 4, 5, 16, 17, 20]. In this model, f is given by an oracle, which answers the query of the form "$f(v) =?$" We care only about the number of queries made; all other computations are free. If we are allowed to toss coins to decide the next query and also allow a small constant error probability, then we have a randomized query algorithm. If we are allowed the use of quantum mechanics to query all the positions (and get corresponding answers) in superposition, then we have a quantum query algorithm. (More precise definitions of the query models and complexities are given in section 2.) The deterministic, randomized, and quantum query complexities are the minimum numbers of queries needed to compute the function by a deterministic, randomized, and quantum query algorithm, respectively. We use $RLS(G)$ and $QLS(G)$ to denote the randomized and quantum query complexities of Local Search on graph G, respectively.

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 is a lower bound question, and the second 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 $[n]^d$:

$$(1) \qquad RLS(\{0,1\}^n) = \Omega(2^{n/2}/n^2), \qquad QLS(\{0,1\}^n) = \Omega(2^{n/4}/n);$$

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

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, the question

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

remains an open problem, explicitly stated 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 question 3, except for a few low-dimensional grids $[n]^d$, in which cases our new bounds also significantly improve the old ones.

1.1. Lower bounds.

Main ideas. Our proof of the quantum lower bounds uses the quantum adversary method, which was originally proposed by Ambainis [7] and later generalized in different ways [6, 8, 15, 24]. Recently Špalek and Szegedy made the picture clear by showing that all these generalizations are equivalent in power [21]. On the other hand, in proving lower bounds for a particular problem, some of the methods might be easier to apply than the others. In our case, the technique in [24], which generalizes the one in [6], 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 on input pairs. Previous proofs [2, 20] define the input sets and weight functions by using random walks on graphs. It turns out that the probability that a random path *passes* a vertex v *within* T steps, which we will refer to as the *passing probability*, plays an important role in the resulting lower bounds: The smaller the passing probability is, the better the lower bound is. Unfortunately in previously constructed random walks [2, 20], the passing probability is not small enough to obtain tight lower bounds for Local Search on $\{0,1\}^n$ and $[n]^d$.

Observe that both graphs $\{0,1\}^n$ and $[n]^d$ can be naturally decomposed as the product of two smaller graphs: $\{0,1\}^n = \{0,1\}^m \otimes \{0,1\}^{n-m}$ and $[n]^d = [n]^m \otimes [n]^{d-m}$. Here 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

$$E = \{(v_1 \otimes v_2, v_1' \otimes v_2) : (v_1, v_1') \in E_1, v_2 \in V_2\} \cup \{(v_1 \otimes v_2, v_1 \otimes v_2') : (v_2, v_2') \in E_2, v_1 \in V_1\}. \tag{3}$$

A key idea in our proof that yields better lower bounds is a different construction of random walk product graphs $G_1 \times G_2$. Specifically, we first fix a long and self-avoiding path $P = (v_0, \cdots, v_L)$ in G_2, then design a random walk in $G_1 \times G_2$ as follows. The particle starts at (u, v_0), where u is a vertex in G_1 and v_0 is the first vertex on P in G_2. It first goes to (u', v_0), where u' is a random neighbor of u, and it then goes to (u', v_1). That is, one step of the random walk in $G_1 \times G_2$ is formed by one step of the random walk in G_1 followed by one step of the walk along the path P in G_2. The particle repeats this process of alternatively performing the walks in G_1 and G_2.

Since the walk in G_2 is one-way, one can view it as a "clock" to record the number of steps taken by the random walk in G_1. A big advantage of adding this clock is that the passing probability in $G_1 \times G_2$, the probability that the random path in $G_1 \times G_2$ passes a vertex (u, v_t) within T steps, is now the *hitting probability* of the random walk in G_1, the probability that a random walk (in G_1) *hits* u *in exactly* t steps. Naturally, the hitting probability is much smaller than the passing probability, therefore eventually yielding better lower bounds for us. Another advantage of the clock is that since the walk in G_2 is self-avoiding, the resulting random path in $G_1 \times G_2$ is self-avoiding as well, which makes part of the analysis easier.

Results. We first describe a lower bound for general product graphs. Given a graph $G = (V, E)$, a random walk is a mapping $W : V \to 2^V$, where $W(u) \subseteq \{u\} \cup \{v : (u, v) \in E\}$. 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 $|W(u)|$ does not depend on u. 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 3.3) in section 3.

THEOREM 1.1. *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 on G_1, we have*

$$RLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} p_t}\right), \quad QLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} \sqrt{p_t}}\right). \tag{4}$$

When applying the theorem to $\{0,1\}^n$ and $[n]^d$, we need to decompose the graph carefully. Take $\{0,1\}^n = \{0,1\}^m \times \{0,1\}^{n-m}$ for instance. On one hand, we hope $\{0,1\}^m$ is large since the hitting probability in a larger graph is smaller. On the other hand, we also hope that $\{0,1\}^{n-m}$ is large so that it contains a path which is long enough to serve as the clock. A tradeoff thus exists, and an optimization over m is needed. For $[n]^d$, unfortunately, the optimal m is not an integer in general, and section 4.2.2 shows how to get around this difficulty.

It is also worth noting that to apply Theorem 1.1, we need to know not only the mixing time of the random walk in G_1, but also *its behavior before mixing* (because the summations in (4) are over the entire $t = 1, \ldots, T$). So the applications are not simply using standard upper bounds on the mixing times, but involving heavy analysis (Lemma 4.1 and Proposition 4.2) on the entire mixing processes.

The results for $\{0,1\}^n$ and $[n]^d$ improve previous ones and show tight bounds on both *RLS* and *QLS* except for a few cases in the low-dimensional grids.

THEOREM 1.2.

$$RLS(\{0,1\}^n) = \Theta(2^{n/2} n^{1/2}), \qquad QLS(\{0,1\}^n) = \Theta(2^{n/3} n^{1/6}). \tag{5}$$

THEOREM 1.3.

$$RLS([n]^d) = \Theta(n^{d/2}) \quad \text{if } d \geq 4, \qquad QLS([n]^d) = \Theta(n^{d/3}) \quad \text{if } d \geq 6. \tag{6}$$

The bounds for the low-dimensional cases are summarized by the following table, where the second and third rows are for $RLS([n]^d)$ and $QLS([n]^d)$, respectively.

d	2	3	4	5
R	$\Omega(n^{\frac{2}{3}})$, $O(n)$	$\Omega(n^{\frac{3}{2}}/\log^{\frac{1}{2}} n)$, $O(n^{\frac{3}{2}})$	-	-
Q	$\Omega(n^{\frac{2}{5}})$, $O(n^{\frac{2}{3}})$	$\Omega(n^{\frac{3}{4}})$, $O(n)$	$\Omega(n^{\frac{6}{5}})$, $O(n^{\frac{4}{3}})$	$\Omega(n^{\frac{5}{3}}/\log^{\frac{1}{3}} n)$, $O(n^{\frac{5}{3}})$

Note that the upper bounds are the ones given in [2, 4]. We include them here for comparison to the lower bounds. See the next section for a better upper bound for $[n]^2$.

1.2. Upper bounds. 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 (vertex) expansion plays a key role. For a graph $G = (V, E)$, the distance $l(u,v)$ 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 $c(k) = \max_{v \in V} |\{u : l(u,v) \leq k\}|$. Clearly, the smaller $c(k)$ is, the more slowly the graph expands. (Actually $c(k)$ is an upper

bound of the standard definition of the expansion.) We say a graph is of polynomial growth if $c(k) = O(k^\alpha)$ for some constant $\alpha \geq 1$. As a special case of Theorem 5.1 in section 5, the following upper bounds for the graphs of polynomial growth hold.

THEOREM 1.4. *If* $c(k) = O(k^\alpha)$ *for some constant* $\alpha \geq 1$, *then*

$$RLS(G) = \begin{cases} O\left(d^{\alpha-1} \log\log d\right) & \text{if } \alpha > 1, \\ O(\log d \log\log d) & \text{if } \alpha = 1, \end{cases} \tag{7}$$

$$QLS(G) = \begin{cases} O\left(d^{\frac{\alpha-1}{2}} (\log\log d)^{1.5}\right) & \text{if } \alpha > 1, \\ O(\log d \log\log d) & \text{if } \alpha = 1, \end{cases} \tag{8}$$

where d *is the diameter of the graph* G.

Note that by Theorem 1.3, it is tempting to conjecture that $\Theta(n^{d/3})$ is the correct answer for $QLS([n]^d)$ for all d's. The following corollary of the above Theorem, however, implies that Local Search on grids exhibits different properties in low dimensions. And we now see that one explanation of the difference is that the expansion is smaller as d decreases.

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

Other related results. Before this paper, it was mentioned in [2] that Ambainis showed $QLS(\{0,1\}^n) = \Omega(2^{n/3}/n^{O(1)})$ (unpublished).[3]

There has been subsequent work: After the preliminary version of this paper appeared, Verhoeven independently showed an upper bound in terms of the genus of the graph [23], giving an $O(\sqrt{n} \log\log n)$ quantum algorithm for $[n]^2$. Very recently, by applying a different random walk in the graphs, Sun and Yao [22] showed that $RLS([n]^2) = \Omega(n^{1-\delta})$, $QLS([n]^2) = \Omega(n^{1/2-\delta})$, and $QLS([n]^3) = \Omega(n^{1-\delta})$, which (almost) closes three of the four gaps ($RLS([n]^2)$, $QLS([n]^2)$, $QLS([n]^3)$, $QLS([n]^4)$) left in this paper.

2. Preliminaries and notation. We use $[M]$ to denote the set $\{1, 2, \ldots, M\}$. For an n-bit binary string $x = x_0 \cdots x_{n-1} \in \{0,1\}^n$, let $x^{(i)} = x_0 \cdots x_{i-1}(1 - x_i)x_{i+1} \cdots x_{n-1}$ be the string obtained by flipping 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 $E_1 \subseteq E_2$. Clearly, 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 let the variables $v_1 \otimes v_2$ 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 $(v_1, \cdots, v_l)$ 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

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

denote the set of distinct vertices on path X. A path is self-avoiding if $v_1, \cdots, v_l$ are all distinct. The length of a path $(v_1, \cdots, v_l)$ is $l-1$. For two vertices $u, v \in V$, the distance $l_G(u,v)$ 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 $E = \{(u,v) : \exists i \in \{0, \cdots, l-1\}$, such that $|u_i - v_i| = 1$ and $u_j = v_j$ for all $j \neq i\}$. Sometimes we abuse the notation by using $[k]^l$ to denote $G_{k,l}$. Note that both the Boolean hypercube and the constant-dimensional grid are special hypercubes.[4]

In an N-vertex graph $G = (V,E)$, a Hamilton path is a path $X = (v_1, \cdots, v_N)$ 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 $(1, \cdots, k)$. 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), we let $HamPath_{k,l} = (v_1, \cdots, v_N)$ 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 $i \in [N-1]$.

Consider a function $F : I^n \to [M]$, where $I = \{0, 1, \ldots, K-1\}$ is the alphabet for each variable, As mentioned in section 1, a deterministic query algorithm for F accesses the input $x \in I^n$ only by making queries in the form of "$x_i =?$." Each query has cost 1, and all the other computations between queries are free. A randomized query algorithm is the same except that the algorithm can toss coins to decide the next variable x_i to ask. The quantum query model, formally introduced in [9], has a working state in the form of $\sum_{i,a,z} \alpha_{i,a,z} |i,a,z\rangle$, where i ranges over $[n]$, a ranges over I, and z is the content in the working space. A quantum query on the input x corresponds to an oracle O_x, a unitary operation defined by

$$O_x \left(\sum_{i,a,z} \alpha_{i,a,z} |i,a,z\rangle \right) = \sum_{i,a,z} \alpha_{i,a,z} |i, a \oplus x_i, z\rangle, \tag{9}$$

where $a \oplus x_i$ is the bitwise XOR of a and x_i as two $\lceil \log K \rceil$-bit strings. A T-query quantum query algorithm works as a sequence of operations

$$U_0 \to O_x \to U_1 \to O_x \to \cdots \to U_{T-1} \to O_x \to U_T. \tag{10}$$

Here O_x is as defined above, and each U_t does not depend on the input x. In both randomized and quantum query models, we can allow a double-sided small constant error probability. The deterministic, randomized, and quantum query complexities, denoted by $D(F)$, $R_2(F)$, and $Q_2(F)$, are the minimum numbers of queries we need to make in order to compute the function by a deterministic, randomized, and quantum query algorithm, respectively. For more details on the query models and the corresponding query complexities, we refer to [10] as an excellent survey.

In the setting of Local Search on graph $G = (V,E)$, the input $f : V \to [M]$ is given by the oracle. Classically the oracle answers the query "$f(x) =?$," and quantumly the oracle O_f maps $|v,a,z\rangle$ to $|v, a \oplus f(v), z\rangle$. We use $RLS(G)$ and $QLS(G)$ to denote the randomized and quantum query complexities of Local Search on G, respectively.

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

2.1. The quantum and relational adversary methods. The quantum adversary method is one of the two powerful tools for proving lower bounds on quantum query complexity; see [13] for a comprehensive survey of this research area. In this paper, we will use the quantum adversary method proposed in [24]. The definition and theorem given here are a little more general than the original ones, but the proof remains unchanged.

DEFINITION 2.1. *Let $F : I^N \to [M]$ be an N-variate function. Let $R \subseteq I^N \times I^N$ be a relation such that $F(x) \neq F(y)$ 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*

$$u(x,y,i)v(x,y,i) \geq w^2(x,y) \tag{11}$$

for all $(x, y) \in R$ and $i \in [N]$ with $x_i \neq y_i$. We further put

$$w(x) = \sum_{y':(x,y')\in R} w(x,y'), \qquad w(y) = \sum_{x':(x',y)\in R} w(x',y), \tag{12}$$

$$u(x,i) = \sum_{y':(x,y')\in R, x_i\neq y'_i} u(x,y',i), \qquad v(y,i) = \sum_{x':(x',y)\in R, x'_i\neq y_i} v(x',y,i). \tag{13}$$

THEOREM 2.2 (see Zhang [24]). *For any F, R and any weight scheme w, u, v as in Definition* 2.1*, we have*

$$Q_2(F) = \Omega\left(\min_{(x,y)\in R, i\in[N]:\ x_i\neq y_i} \sqrt{\frac{w(x)w(y)}{u(x,i)v(y,i)}}\right). \tag{14}$$

In [2], Aaronson gives a nice technique for getting a lower bound for randomized query complexity. We restate it using language similar to that of Theorem 2.2.

THEOREM 2.3 (see 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*

$$R_2(F) = \Omega\left(\min_{(x,y)\in R, i\in[N], x_i\neq y_i} \max\left\{\frac{w(x)}{w(x,i)}, \frac{w(y)}{w(y,i)}\right\}\right), \tag{15}$$

where

$$w(x,i) = \sum_{y':(x,y')\in R, x_i\neq y'_i} w(x,y'), \qquad w(y,i) = \sum_{x':(x',y)\in R, x'_i\neq y_i} w(x',y). \tag{16}$$

Note that we can think of Theorem 2.3 as having a weight scheme, too, but requiring that $u(x, y, i) = v(x, y, i) = w(x, y)$. This simple observation is used in the proof of Theorems 1.2 and 1.3.

3. Lower bounds for Local Search on product graphs. In this section we prove a theorem which is stronger than Theorem 1.1 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 $W : V \times \mathbb{N} \to 2^V$ such that 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. (Note that $W(u, t)$ depends on t in general, and actually we will use this dependence when proving lower bounds for $[n]^d$.) 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* $(v_0, v_1, \cdots, v_T)$ *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.

DEFINITION 3.1. *The time-t hitting probability is*

$$p_t = \max\{p(u, t_1, v, t_2) : u, v \in V,\ t_1, t_2 \in \mathbb{Z},\ t_2 - t_1 = t\}. \tag{17}$$

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

Proof. By considering the two cases of the step $t_1 + 1$ (going to u' or not), we have

$$p(u, t_1, v, t_2) = \frac{1}{|W(u, t_1 + 1)|} p(u', t_1{+}1, v, t_2) + \left(1 - \frac{1}{|W(u, t_1 + 1)|}\right) q(u, u', t_1, v, t_2). \tag{18}$$

Thus

$$q(u, u', t_1, v, t_2) \le p(u, t_1, v, t_2) / \left(1 - \frac{1}{|W(u, t_1 + 1)|}\right) \le 2p(u, t_1, v, t_2). \tag{19}$$ ∎

THEOREM 3.3. *Suppose G contains $G_1 \times G_2$ (for two arbitrary graphs G_1 and 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 random walk (G_1, v_0, W) on G_1, we have*

$$RLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} p_t}\right), \qquad QLS(G) = \Omega\left(\frac{T}{\sum_{t=1}^{T} \sqrt{p_t}}\right), \tag{20}$$

where p_t is as defined in Definition 3.1 *for the walk (G_1, v_0, W).*

Proof. Without loss of generality, we assume $G = G_1 \times G_2$, as Local Search on a subgraph is no harder than Local Search on the original graph. Throughout the proof, we will use x, y to denote vertices in G_1, and z for vertices in G_2.

We shall construct a random walk on G from the random walk (G_1, v_0, W) on G_1 and a simple one-way walk on G_2. Starting from some fixed vertex in G, the walk proceeds by one step of walk in G_1 followed by two steps of walk in G_2. (We perform *two* steps of walk in G_2 for technical reasons, and this is where the factor of 2 in definition $T = \lfloor L/2 \rfloor$ comes from.) Precisely, fix a self-avoiding path $(z_{0,0}, z_{1,0}, z_{1,1}, z_{2,1}, z_{2,2}, \cdots, z_{T,T-1}, z_{T,T})$ of length $2T$ in G_2. Let the set P contain all the paths X in G in the form

$$X = (x_0 \otimes z_{0,0}, x_1 \otimes z_{0,0}, x_1 \otimes z_{1,0}, x_1 \otimes z_{1,1}, \cdots, x_T \otimes z_{T-1,T-1}, x_T \otimes z_{T,T-1}, x_T \otimes z_{T,T}), \tag{21}$$

where $x_0 = v_0$ and $(x_0, x_1, \cdots, x_T)$ is a path generated by the random walk (G_1, v_0, W). Define a problem PATH_P: Given a path $X \in P$, find the end point $x_T \otimes z_{T,T}$. The path X is accessed by an oracle, which takes a vertex v in G as input and outputs 1 or 0

depending on whether $v \in set(X)$.[5] The following claim says that the PATH$_P$ problem is not much harder than the problem Local Search in terms of query complexities.

CLAIM 1. $R_2(\text{PATH}_P) \leq 2RLS(G)$, $Q_2(\text{PATH}_P) \leq 4QLS(G)$.

Proof. Suppose we have a 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 Aaronson did in [2]: For each vertex $v \in G$, let

$$f_X(v) = \begin{cases} l_G(v, x_0 \otimes z_{0,0}) + 3T & \text{if } v \notin set(X), \\ 3(T-k) & \text{if } v = x_k \otimes z_{k,k}, \\ 3(T-k)-1 & \text{if } v = x_{k+1} \otimes z_{k,k} \neq x_k \otimes z_{k,k}, \\ 3(T-k)-2 & \text{if } v = x_{k+1} \otimes z_{k+1,k}. \end{cases} \tag{22}$$

It is easy to verify that the only local minimum is $x_T \otimes z_{T,T}$.

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 $f_X(v) = l_G(v, x_0 \otimes z_{0,0}) + 3T$; otherwise, $v = x \otimes z_{k+1,k}$ or $v = x \otimes z_{k,k}$ for some $x \in V$ and k. Note that k is known for any given vertex v (since the path in G_2 is self-avoiding and fixed). So if $v = x \otimes z_{k+1,k}$, then $x = x_{k+1}$ and thus $f_X(v) = 3(T-k)-2$. Now consider the case that $v = x \otimes z_{k,k}$. If $k = 0$, then let $f_X(v) = 3T$ if $v = x_0 \otimes z_{0,0}$ and $f_X(v) = 3T - 1$ otherwise. If $k \geq 1$, then $\mathcal{B}$ asks O whether $x \otimes z_{k,k-1} \in set(X)$. If yes, then $v = x_k \otimes z_{k,k}$ and thus $f_X(v) = 3(T-k)$; if no, then $v = x_{k+1} \otimes z_{k,k} \neq x_k \otimes z_{k,k}$ 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 the randomized model. The extra factor of 2 in the quantum case comes from the standard cancellation process of the quantum queries. This proves the claim. ∎

(Continuation of the proof of Theorem 3.3.) By Claim 1, it is sufficient to prove lower bounds for PATH$_P$. We define a relation R_P as follows:

$$R_P = \{(X, Y) : X \in P,\ Y \in P,\ X \text{ and } Y \text{ have different end points}\}. \tag{23}$$

For any pair $(X, Y) \in R_P$, where

$$X = (x_0 \otimes z_{0,0}, x_1 \otimes z_{0,0}, x_1 \otimes z_{1,0}, x_1 \otimes z_{1,1}, \cdots, x_T \otimes z_{T-1,T-1}, x_T \otimes z_{T,T-1}, x_T \otimes z_{T,T})$$

and

$$Y = (y_0 \otimes z_{0,0}, y_1 \otimes z_{0,0}, y_1 \otimes z_{1,0}, y_1 \otimes z_{1,1}, \cdots, y_T \otimes z_{T-1,T-1}, y_T \otimes z_{T,T-1}, y_T \otimes z_{T,T}),$$

we write $X \wedge Y = k$ if $x_0 = y_0, \cdots, x_{k-1} = y_{k-1}$ but $x_k \neq y_k$. Intuitively, $X \wedge Y = k$ if k is the place where the paths X and Y diverge for the first time. Note that if $X \wedge Y = k$, then $x_k, y_k \in W(x_{k-1}, k)$ and thus $|W(x_{k-1}, k)| \geq 2$. By Lemma 3.2, this implies that $q(x_{k-1}, x_k, k-1, v, j) \leq 2p_{j-k+1}$.

[5] Note that it is actually an oracle for the function $g : \{0,1\}^n \to \{0,1\}$, with $g(x) = 1$ if and only if $x \in set(X)$. 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 check that for any $X, Y \in P$, it holds that $X = Y \Leftrightarrow set(X) = set(Y)$. Indeed, if $X \neq Y$, suppose the first diverging place is k; i.e., $x_{k-1} = y_{k-1}$, but $x_k \neq y_k$. Then Y will never pass $x_k \otimes z_{k,k-1}$ because the clock immediately ticks and the time always advances forward. (Or more rigorously, the only point that is on Y and has the second component $z_{k,k-1}$ is $y_k \otimes z_{k,k-1}$. Since $y_k \neq x_k$, $x_k \otimes z_{k,k-1} \notin set(Y)$.)

We choose the weight functions in Theorem 2.2 by letting

$$w(X,Y) = 1/|\{Y' \in P : Y' \wedge X = k\}| \tag{24}$$

$$= 1/|\{X' \in P : X' \wedge Y = k\}| \tag{25}$$

$$= 1/[(c_k - 1)c_{k+1} \cdots c_T], \tag{26}$$

where $c_t = |W(x_{t-1}, t)|$ in random walk (G_1, x_0, W).

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':

$$w(X) = \sum_{k'=1}^{T} \sum_{\substack{Y':(X,Y')\in R_P \\ X\wedge Y' = k'}} w(X,Y') \tag{27}$$

$$= \sum_{k'=1}^{T} \sum_{\substack{Y':(X,Y')\in R_P \\ X\wedge Y' = k'}} \frac{1}{|\{Y' \in P : Y' \wedge X = k'\}|} \tag{28}$$

$$= \sum_{k'=1}^{T} \mathbf{Pr}_{Y'}[(X,Y') \in R_P | Y' \wedge X = k'] \tag{29}$$

$$= \sum_{k'=1}^{T} \mathbf{Pr}_{Y'}[(y')_T \neq x_T | Y' \wedge X = k']. \tag{30}$$

Here equality (29) holds because all paths diverging from X for the first time at k' have the same probability $1/[(c_{k'} - 1)c_{k'} \cdots c_T]$. Also note that the probability in the last equality is nothing but $1 - q(x_{k'-1}, x_{k'}, k' - 1, x_T, T)$, which is at least $1 - 2p_{T-k'+1}$. So we have

$$w(X) \geq T - 2\sum_{k'=1}^{T} p_{T-k'+1} = T - 2\sum_{t=1}^{T} p_t. \tag{31}$$

Similarly, we have $w(Y) \geq T - 2\sum_{t=1}^{T} p_t$, too.

Now we describe $u(X,Y,i)$ and $v(X,Y,i)$, where i is a point $x_{j+r} \otimes z_{j+s,j} \in set(X) - set(Y)$ or $y_{j+r} \otimes z_{j+s,j} \in set(Y) - set(X)$. Here $(r,s) \in \{(0,0),(1,0),(1,1)\}$, and $0 \leq j \leq j + r \leq T$. Let

$$u(X,Y,x_{j+r} \otimes z_{j+s,j}) = a_{k,j,r,s} w(X,Y), \quad u(X,Y,y_{j+r} \otimes z_{j+s,j}) = b_{k,j,r,s} w(X,Y), \tag{32}$$

$$v(X,Y,x_{j+r} \otimes z_{j+s,j}) = b_{k,j,r,s} w(X,Y), \quad v(X,Y,y_{j+r} \otimes z_{j+s,j}) = a_{k,j,r,s} w(X,Y), \tag{33}$$

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

To apply Theorems 2.2 and 2.3, let us calculate the quantities $u(x,i)$ and $v(y,i)$

in Definition 2.1:

$$u(X, x_{j+r} \otimes z_{j+s,j}) = \sum_{k'=1}^{j+r} \sum_{\substack{Y':(X,Y')\in R_P, X\wedge Y'=k' \\ x_{j+r}\otimes z_{j+s,j}\notin set(Y')}} a_{k',j,r,s} w(X, Y') \tag{34}$$

$$\leq \sum_{k'=1}^{j+r} \sum_{Y':X\wedge Y'=k'} a_{k',j,r,s} w(X, Y') \tag{35}$$

$$= \sum_{k'=1}^{j+r} a_{k',j,r,s}. \tag{36}$$

The computation for $v(Y, x_{j+r} \otimes z_{j+s,j})$ is a little more complicated. By definition,

$$v_{Y,x_{j+r}\otimes z_{j+s,j}} = \sum_{k'=1}^{j+r} \sum_{\substack{X':(X',Y)\in R_P,\ X'\wedge Y=k', \\ x_{j+r}\otimes z_{j+s,j}\in set(X')}} b_{k',j,r,s} w(X', Y) \tag{37}$$

$$\leq \sum_{k'=1}^{j+r} \sum_{\substack{X':X'\wedge Y=k', \\ x_{j+r}\otimes z_{j+s,j}\in set(X')}} b_{k',j,r,s} w(X', Y) \tag{38}$$

$$= \sum_{k'=1}^{j+r} b_{k',j,r,s} \mathbf{Pr}_{X'}[x_{j+r} \otimes z_{j+s,j} \in set(X') | X' \wedge Y = k']. \tag{39}$$

By adding the clock, the passing probability $\mathbf{Pr}_{X'}[x_{j+r} \otimes z_{j+s,j} \in set(X')|X' \wedge Y = k']$ is roughly the hitting probability $q(y_{k'-1}, y_{k'}, k'-1, x_{j+r}, j) + q(y_{k'-1}, y_{k'}, k'-1, x_{j+r}, j+1)$ except for some boundary cases. To be more precise, define

$$\begin{aligned} Bound_{k',j,r,s} = {} & 2p_{j-k'+2} \cdot \lambda[s = 1 \text{ OR } j < T] \\ & + 2p_{j-k'+1} \cdot \lambda[s = 0 \text{ AND } (k' \leq j \text{ OR } r = 0)], \end{aligned} \tag{40}$$

where the Boolean function $\lambda[\phi] = 1$ if ϕ is true and 0 otherwise. Then we have the following claim.

CLAIM 2. $\mathbf{Pr}_{X'}[x_{j+r} \otimes z_{j+s,j} \in set(X')|X' \wedge Y = k'] \leq Bound_{k',j,r,s}$.

Proof. We study the probability $\mathbf{Pr}_{X'}[x_{j+r}\otimes z_{j+s,j} \in set(X')|X'\wedge Y = k']$ case by case. If $s = 1$, then $r = 1$, and $x_{j+1}\otimes z_{j+1,j} \in set(X')$ if and only if $x_{j+1} = (x')_{j+1}$. So

$$\mathbf{Pr}_{X'}[x_{j+r}\otimes z_{j+s,j} \in set(X')|X'\wedge Y = k'] = q(y_{k'-1}, y_{k'}, k'-1, x_{j+1}, j+1) \leq 2p_{j-k'+2} \tag{41}$$

by Lemma 3.2. If $s = 0$, then $x_{j+r} \otimes z_{j,j} \in set(X')$ if and only if "$x_{j+r} = (x')_j$ or $x_{j+r} = (x')_{j+1}$." Also note that

$$\mathbf{Pr}_{X'}[x_{j+r} = (x')_j | X' \wedge Y = k'] = q(y_{k'-1}, y_{k'}, k'-1, x_{j+r}, j) \tag{42}$$

unless $k' = j+1$ and $r = 1$, in which case $\mathbf{Pr}_{X'}[x_{j+r} = (x')_j|X'\wedge Y = k'] = 0$ because $x_{j+1} \otimes z_{j,j} \notin set(Y)$, but $(x')_j \otimes z_{j,j} = y_j \otimes z_{j,j} \in set(Y)$. The other probability

$$\mathbf{Pr}_{X'}[x_{j+r} = (x')_{j+1}|X' \wedge Y = k'] = \begin{cases} q(y_{k'-1}, y_{k'}, k'-1, x_{j+r}, j+1) & \text{if } j \leq T-1, \\ 0 & \text{if } j = T. \end{cases} \tag{43}$$

Putting all cases together, we get the desired result. This proves the claim. □

(Continuation of the proof of Theorem 3.3.) Claim 2 implies that

$$v(Y, x_{j+r} \otimes z_{j+s,j}) \le \sum_{k'=1}^{j+r} b_{k',j,r,s} Bound_{k',j,r,s}. \tag{44}$$

The symmetric case of $u(X, Y, i)$ and $v(X, Y, i)$ where i is a point $y_{j+r} \otimes z_{j+s,j} \in set(Y) - set(X)$ can be dealt with in the same way, yielding $u(X, y_{j+r} \otimes z_{j+s,j}) \le \sum_{k'=1}^{j+r} b_{k',j,r,s} Bound_{k',j,r,s}$ and $v(Y, y_{j+r} \otimes z_{j+s,j}) \le \sum_{k'=1}^{j+r} a_{k',j,r,s}$.

By the definition of $Bound_{k',j,r,s}$, it holds for any (j, r, s) that

$$\sum_{k'=1}^{j+r} Bound_{k',j,r,s} \le 4 \sum_{t=1}^{T} p_t \qquad \text{and} \qquad \sum_{k'=1}^{j+r} \sqrt{Bound_{k',j,r,s}} \le 4 \sum_{t=1}^{T} \sqrt{p_t}. \tag{45}$$

Now for the randomized lower bound, let $a_{k',j,r,s} = b_{k',j,r,s} = 1$; then

$$RLS(G) = \Omega\left(\min_{j,r,s} \max\left\{ \frac{T - 2\sum_{t=1}^{T} p_t}{j+r}, \frac{T - 2\sum_{t=1}^{T} p_t}{\sum_{k'=1}^{j+r} Bound_{k',j,r,s}} \right\} \right) = \Omega\left(\frac{T}{\sum_{t=1}^{T} p_t} \right). \tag{46}$$

For the quantum lower bound, pick

$$a_{k',j,r,s} = \sqrt{Bound_{k',j,r,s}} \quad \text{and} \quad b_{k',j,r,s} = 1/\sqrt{Bound_{k',j,r,s}}. \tag{47}$$

Then

$$QLS(G) = \Omega\left(\min_{j,r,s} \sqrt{ \frac{\left(T - 2\sum_{t=1}^{T} p_t\right)\left(T - 2\sum_{t=1}^{T} p_t\right)}{\left(\sum_{k'=1}^{j+r} \sqrt{Bound_{k',j,r,s}}\right)\left(\sum_{k'=1}^{j+r} \sqrt{Bound_{k',j,r,s}}\right)} } \right) \tag{48}$$

$$= \Omega\left(\frac{T}{\sum_{t=1}^{T} \sqrt{p_t}} \right). \tag{49}$$

This completes the proof of Theorem 3.3. □

4. Applications to the two special graphs. In this section, we will apply Theorem 3.3 to the two special graphs. Note that in both cases, the probability p_t is not easy to upper bound. Also note that we need to pick not only the random walk, but also the way to decompose the graph.

4.1. Lower bounds for Local Search on the Boolean hypercube. To apply Theorem 3.3 to $\{0,1\}^n$, we decompose the whole graph into the two parts $\{0,1\}^m$ and $\{0,1\}^{n-m}$, where $m = \lfloor (n + \log_2 n)/2 \rfloor$ in the proof of the randomized lower bound and $m = \lfloor (2n + \log_2 n)/3 \rfloor$ in the proof of the quantum lower bound. In both the randomized and quantum cases, pick the random walk $(\{0,1\}^m, v_0, W)$, where $v_0 = 0^m \in \{0,1\}^m$ and $W(x,t) = \{x^{(i)} : i \in \{0, \cdots, m-1\}\}$ for each vertex $x = x_0 \cdots x_{m-1} \in \{0,1\}^m$ and each $t \in \mathbb{N}$. (Recall that $x^{(i)} = x_0 \cdots x_{i-1}(1 - x_i)x_{i+1} \cdots x_{m-1}$.) 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 4.1. *For any $t \in \mathbb{N}$, we have*

$$p_t = \begin{cases} O(m^{-\lceil t/2 \rceil}) & \text{if } t \le 10, \\ O(m^{-5}) & \text{if } 10 < t \le m^2, \\ O(2^{-m}) & \text{if } t > m^2. \end{cases} \tag{50}$$

Once we prove the lemma, Theorem 1.2 follows immediately. Actually, for the randomized lower bound, $T = \Theta(2^{n/2}/n^{1/2})$ and $\sum_{t=1}^{T} p_t = O(1/n)$. Thus $RLS(\{0,1\}^n) = \Omega(\sqrt{n}2^{n/2})$. For the quantum lower bound, let $T = \Theta(2^{n/3}/n^{1/3})$ and $\sum_{t=1}^{T} \sqrt{p_t} = O(1/\sqrt{n})$. Thus $QLS(\{0,1\}^n) = \Omega(2^{n/3}n^{1/6})$. Next we prove the lemma.

Proof of Lemma 4.1. Consider that we put t balls randomly into m bins one by one. The jth ball goes into the i_jth bin. Denote by n_i the total number of balls in the ith bin. We write $n_i \equiv b_i$ if $b_i = n_i \bmod 2$. We say that $(i_1, \cdots, i_t)$ *generates the parity sequence* $(b_1, \cdots, b_m)$, or simply that $(i_1, \cdots, i_t)$ *generates* $(b_1, \cdots, b_m)$, if $n_i \equiv b_i$ for all $i \in [m]$. For $b_1 \cdots b_m \in \{0,1\}^m$, denote by $p^{(t)}[b_1, \cdots, b_m]$ the probability that $n_i \equiv b_i$ for all $i \in [m]$. Let $p^{(t)} = \max_{b_1,\cdots,b_m} p^{(t)}[b_1, \cdots, b_m]$. Since flipping each coordinate twice amounts to no flipping at all, it is easy to see that $p^{(t)} = p_t$ in Lemma 4.1, so it is enough to prove the same bounds in Lemma 4.1 for $p^{(t)}$.

We start with several simple observations. First, we assume that t and $\sum_{i=1}^{m} b_i$ have the same parity, because otherwise the probability is 0 and the lemma holds trivially. Second, by the symmetry, any permutation of $b_1, \cdots, b_m$ does not change $p^{(t)}[(b_1, \cdots, b_m)]$. Third, $p^{(t)}[(b_1, \cdots, b_m)]$ decreases if we replace two 1's in $b_1, \cdots, b_m$ by two 0's. Precisely, if we have two b_i's being 1, say $b_1 = b_2 = 1$, then $p^{(t)}[(b_1, \cdots, b_m)] < p^{(t)}[(0, 0, b_3, \cdots, b_m)]$. In fact, we note that

$$\begin{aligned} & p^{(t)}[(b_1, \cdots, b_m)] \\ =& \frac{1}{m^t} \sum_{\substack{n_1+\cdots+n_m=t \\ n_i \equiv b_i, i\in[m]}} \frac{t!}{n_1! \cdots n_m!} \qquad (51) \\ =& \frac{1}{m^t} \sum_{\substack{n_3+\cdots+n_m \le t \\ n_i \equiv b_i, i=3,\ldots,m}} \left(\frac{t!}{(n_1+n_2)! n_3! \cdots n_m!} \sum_{\substack{n_1+n_2=t-n_3-\cdots-n_m \\ n_i \equiv b_i, i=1,2}} \frac{(n_1+n_2)!}{n_1! n_2!} \right), \qquad (52) \end{aligned}$$

where, as usual, we let $0! = 1$. If $n_3 + \cdots + n_m < t$, then

$$\sum_{\substack{n_1+n_2=t-n_3-\cdots-n_m \\ n_i \equiv 1, i=1,2}} \frac{(n_1+n_2)!}{n_1! n_2!} = \sum_{\substack{n_1+n_2=t-n_3-\cdots-n_m \\ n_i \equiv 0, i=1,2}} \frac{(n_1+n_2)!}{n_1! n_2!}. \tag{53}$$

If $n_3 + \cdots + n_m = t$, then the only possible (n_1, n_2) is $(0,0)$, so

$$\sum_{\substack{n_1+n_2=t-n_3-\cdots-n_m \\ n_i \equiv 1, i=1,2}} \frac{(n_1+n_2)!}{n_1! n_2!} = 0, \qquad \sum_{\substack{n_1+n_2=t-n_3-\cdots-n_m \\ n_i \equiv 0, i=1,2}} \frac{(n_1+n_2)!}{n_1! n_2!} = 1. \tag{54}$$

Thus $p^{(t)}[(1,1,b_3,\cdots,b_m)] < p^{(t)}[(0,0,b_3,\cdots,b_m)]$.

By these observations, it is sufficient to prove the lemma for the case $p^{(t)}[(0,\cdots,0)]$ if t is even, and for the case $p^{(t)}[(1,0,\cdots,0)]$ if t is odd. Note that if t is even, then

$$p^{(t)}[(0,\cdots,0)] = \sum_{i=1}^{m} \mathbf{Pr}[i_1 = i] \mathbf{Pr}[(i_2, \cdots, i_t) \text{ generates } (e_i)], \tag{55}$$

where e_i is the standard m-bit basis vector with only coordinate i being 1 and all other coordinates being 0. By symmetry, $p^{(t-1)}[e_1] = \cdots = p^{(t-1)}[e_m]$; thus $p^{(t)}[(0,\cdots,0)] = p^{(t-1)}[e_1] = p^{(t-1)}[1,0,\cdots,0]$. Therefore, it is enough to show the lemma for even t.

We now express $p^{(t)}[0,\cdots,0]$ in two ways. One is to prove the first case ($t \le 10$) in the lemma, and the other is for the second case ($10 < t \le m^2$) and the third case ($t > m^2$) in the lemma.

To avoid confusion, we write m explicitly as a subscript: $p_m^{(t)}[b_1,\cdots,b_m]$. We consider which bin(s) the first two balls are put into:

$$p_m^{(t)}[0,\cdots,0] = \mathbf{Pr}[i_1 = i_2]p_m^{(t-2)}[0,\cdots,0] + \mathbf{Pr}[i_1 \neq i_2]p_m^{(t-2)}[1,1,0,\cdots,0] \tag{56}$$

$$= \frac{1}{m}p_m^{(t-2)}[0,\cdots,0] + \frac{m-1}{m}p_m^{(t-2)}[1,1,0,\cdots,0]. \tag{57}$$

To compute $p_m^{(t-2)}[1,1,0,\cdots,0]$, we consider how to put $(t-2)$ balls in m bins. By the analysis of the third observation above, we know that

$$p_m^{(t-2)}[0,\cdots,0] - p_m^{(t-2)}[1,1,0,\cdots,0] \tag{58}$$

$$= \mathbf{Pr}[n_1 = n_2 = 0, n_3 \equiv 0, \cdots, n_m \equiv 0] \tag{59}$$

$$= \mathbf{Pr}[n_1 = n_2 = 0]\mathbf{Pr}[n_3 \equiv 0, \cdots, n_m \equiv 0 | n_1 = n_2 = 0] \tag{60}$$

$$= \left(\frac{m-2}{m}\right)^{t-2} p_{m-2}^{(t-2)}[0,\cdots,0]. \tag{61}$$

Therefore,

$$p_m^{(t)}[0,\cdots,0] = \frac{1}{m}p_m^{(t-2)}[0,\cdots,0] - \frac{m-1}{m}\left(\frac{m-2}{m}\right)^{t-2} p_{m-2}^{(t-2)}[0,\cdots,0]. \tag{62}$$

Now using the above recursive formula and the base case $p_m^{(2)}[0,\cdots,0] = 1/m$, it is easy (but tedious) to prove by calculations that $p_m^{(t)}[0,\cdots,0] = ((t-1)!!/m^{\frac{t}{2}})(1-o(1))$ for even $t \le 10$. This proves the first case in the lemma.

For the remaining two cases, we shall use a generating function and a technique inspired by Fourier analysis. Consider the generating function

$$(x_1 + \cdots + x_m)^t = \sum_{n_1+\cdots+n_m=t} \binom{t}{n_1,\cdots,n_m} x_1^{n_1}\cdots x_m^{n_m}. \tag{63}$$

If $x_i \in \{-1,1\}$, then $(x_1 + \cdots + x_m)^t = \sum_{n_1+\cdots+n_m=t} \binom{t}{n_1,\cdots,n_m}(-1)^{|\{i:x_i=-1,n_i\equiv 1\}|}$. We sum it over all $x_1\cdots x_m \in \{-1,1\}^m$. Note that for those $(n_1,\cdots,n_m)$ that have some $n_{i_0} \equiv 1$, it holds due to the cancellation that $\sum_{x_1,\cdots,x_m\in\{-1,1\}}(-1)^{|\{i:x_i=-1,n_i\equiv 1\}|} = 0$. On the other hand, if all n_i's are even, then $\sum_{x_1,\cdots,x_m\in\{-1,1\}}(-1)^{|\{i:x_i=-1,n_i\equiv 1\}|} = 2^m$. Thus we have

$$\sum_{x_1,\cdots,x_m\in\{-1,1\}} (x_1 + \cdots + x_m)^t = 2^m \sum_{\substack{n_1+\cdots+n_m=t \\ n_i\equiv 0, i\in[m]}} \binom{t}{n_1,\cdots,n_m}. \tag{64}$$

And therefore,

$$p^{(t)}[0,\cdots,0] = \frac{1}{m^t} \sum_{\substack{n_1+\cdots+n_m=t \\ n_i\equiv 0, i\in[m]}} \binom{t}{n_1,\cdots,n_m} \tag{65}$$

$$= \frac{1}{2^m m^t} \sum_{x_1,\cdots,x_m\in\{-1,1\}} (x_1+\cdots+x_m)^t \tag{66}$$

$$= \frac{1}{2^m m^t} \sum_{i=0}^{m} \binom{m}{i} (m-2i)^t \tag{67}$$

$$= \frac{1}{2^m} \sum_{i=0}^{m} \binom{m}{i} \left(1-\frac{2i}{m}\right)^t. \tag{68}$$

Note that t is even, so $p^{(t)}[0,\cdots,0]$ decreases if t increases by 2, and this proves the second case of the lemma with the help of the first case. And if $t > m^2/2$, then

$$p^{(t)}[0,\cdots,0] \le \frac{1}{2^m}\left(2+\left(1-\frac{2}{m}\right)^t \sum_{i=1}^{m-1}\binom{m}{i}\right) < 2/2^m + e^{-m} = O(1/2^m). \tag{69}$$

This proves the third case of the lemma. □

4.2. Lower bounds for Local Search on the constant-dimensional grid. In this section we shall decompose the graph $[n]^d$ into $[n]^m \otimes [n]^{d-m}$ and apply Theorem 3.3 in section 4.2.1. This is not enough to prove Theorem 1.3 since the optimal m turns out to be a real number instead of an integer. In section 4.2.2, we show how to implement a real-number dimension, which enables us to achieve the results in Theorem 1.3 except for $[n]^2$. Finally, we use a different random walk to obtain the lower bound of $\Omega(n^{2/5})$ for $[n]^2$ in section 4.2.3, which finishes the proof of Theorem 1.3.

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 $x = x_0\cdots x_{m-1} \in [n]^m$ and each $i \in \{0,\cdots,m-1\}$, define

$$x^{(i),-} = x_0\cdots x_{i-1}\max\{x_i-1,1\}x_{i+1}\cdots x_{m-1}, \tag{70}$$

$$x^{(i),+} = x_0\cdots x_{i-1}\min\{x_i+1,n\}x_{i+1}\cdots x_{m-1}. \tag{71}$$

We perform the random walk $([n]^m, v_0, W)$, where $v_0 = 00\cdots 0 \in [n]^m$ and

$$W(x,t) = \{x^{((t-1) \bmod m),+}, x^{((t-1) \bmod m),-}\}. \tag{72}$$

To analyze the probability p_t in Theorem 3.3, we first consider the following simpler "line walk." Suppose a particle is initially put at point $i \in \{1,\cdots,n\}$, 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 stops at point j after exactly t steps of the walk. For $t \ge 1$, the following proposition gives a tight estimate on $\max_{ij} p_{ij}^{(t)}$.

PROPOSITION 4.2. *For any $t \ge 1$,*

$$\max_{i,j} p_{ij}^{(t)} = \begin{cases} O(1/\sqrt{t}) & \text{if} \quad t \le n^2, \\ O(1/n) & \text{if} \quad t > n^2. \end{cases} \tag{73}$$

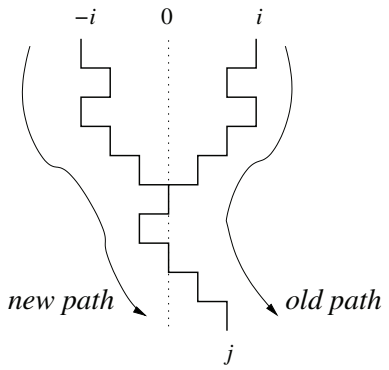

FIG. 1. *The proof of the reflecting rule.*

Before the formal proof, let us briefly discuss the main difficulty and the idea for getting around it. First note that since we care about the whole mixing process (i.e., before and after mixing), the standard eigenvalue gap does not immediately apply. Second, 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. Fortunately, there is a basic *reflecting rule* as follows.

Reflecting rule. In the line walk without barriers, the number of paths from $i > 0$ to $j > 0$ in exactly t steps touching or crossing the point 0 is equal to the number of paths from $-i$ to j in exactly t steps.

The proof of this rule is very easy. Suppose a random path touches the point 0 at t' for the first time where $t' \le t$; then do a reflection of the first t' steps of the path with respect to point 0. See Figure 1 for an illustration. It is not hard to see that this gives a 1-1 correspondence between the following two sets: (1) the set of paths from i to j after exactly t steps touching or crossing the point 0, and (2) the set of paths from $-i$ to j.

Now let us consider the barrier setting. Note that a path may try to cross the two barriers in some pattern; for example, they may try to cross the left barrier (i.e., point 1) a times and then try to cross the right barrier (i.e., point n) b times. Imagine that we now remove the two barriers; then the path will touch (from right) but not cross the point $1-a$ and will touch (from left) but not cross the point $n+b-a$. To use the reflecting rule, we just need to further note the following simple fact:

$$\begin{aligned}&\{\text{paths touching but not crossing point } 1-a\}\\ &\quad=\{\text{paths touching or crossing point } 1-a\}\\ &\qquad-\{\text{paths touching or crossing point } -a\}.\end{aligned}$$

Following this idea, we will construct a series of 1-1 correspondences to reduce the problem step by step to the no-barrier case. The precise proof is as follows.

Proof. We consider two settings. One is the line walk on n points $0, \cdots, n-1$ with the two barriers 0 and $n-1$.[6] Another is the same except that the barriers are removed, and we have an infinite number of points in a line. For each t-bit binary

[6]Here we let the n points be $0, \cdots, n-1$ instead of $1, \cdots, n$ just to make the later calculation cleaner.

string $x = x_1 \cdots x_t$, we use P_i^x and Q_i^x to denote the two paths that start at i and walk according to x in the two settings. Precisely, at step s, Q_i^x goes left if $x_s = 0$ and goes right if $x_s = 1$. P_i^x goes in the same way except that it will stand still if the point is currently at the left (or right) end and it still wants to go left (or right). If the end point of P_i^x is j, then we write $i \to_t^{P,x} j$, referring to the event that the walk P_i^x hits j after exactly t steps. Let $X_{ij}^{(t),P}$ be the set of $x \in \{0,1\}^t$ such that $i \to_t^{P,x} j$, and put $n_{ij}^{(t),P} = |X_{ij}^{(t),P}|$. Then by definition, $p_{ij}^{(t)} = n_{ij}^{(t),P}/2^t$. The notations $i \to_t^{Q,x} j$, $X_{ij}^{(t),Q}$, and $n_{ij}^{(t),Q}$ are similarly defined, with the corresponding P changed to Q. Note that $n_{ij}^{(t),Q} = \binom{t}{t/2+(j-i)/2}$ if $j-i$ and t have the same parity, and 0 otherwise. We now want to upper bound $n_{ij}^{(t),P}$ in terms of $n_{ij}^{(t),Q}$.

For a path P_i^x, if at some step it is at point 0 and wants to go left, we say it *attempts to pass the left barrier*. Similarly for the right barrier. We say a path is in the $\{a_s, b_s\}_{s=1}^l$ category if it first attempts to pass the left barrier a_1 times and then attempts to pass the right barrier b_1 times, and so on. We call each round a stage s, which begins at the time that P_i^x attempts to pass the left barrier for the $(a_1 + \cdots + a_{s-1} + 1)$th time, and ends right before the time that P_i^x attempts to pass the left barrier for the $(a_1 + \cdots + a_s + 1)$th time. We also split each stage s into two halves, cutting at the time right before the path attempts to pass the right barrier for the $(b_1 + \cdots + b_{s-1} + 1)$th time. Note that a_1 may be 0, which means that the path first attempts to pass the right barrier. Also b_l may be 0, which means that the last barrier the path attempts to pass is the left one. But all other a_i, b_i's are positive. Also note that in the case of $l = 0$, the path never attempts to pass either barrier. Now for any fixed $l > 0$, we consider those categories with $a_1 > 0$ and $b_l > 0$. Other cases can be handled similarly. Partition $X_{ij}^{(t),P}$ as

$$X_{ij}^{(t),P} = \bigcup_{l,\ \{a_s,b_s\}_{s=1}^l} X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l], \tag{74}$$

where $X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]$ contains those $x \in \{0,1\}^t$ such that P_i^x is in the category $\{a_s, b_s\}_{s=1}^l$. We put $n_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l] = |X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]|$, and then we have $n_{ij}^{(t),P} = \sum_l \sum_{\{a_s,b_s\}_{s=1}^l} n_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]$.

Now consider the corresponding paths in $X_{ij}^{(t),Q}$. The following observation relates P_i^x and Q_i^x.

OBSERVATION 1. *For each $x \in X_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l]$, the following three properties hold for any s.*

1. *In the first half of stage s, the path Q_i^x touches (from right) but does not cross the point $\alpha_s = \sum_{r=1}^{s-1}(b_r - a_r) - a_s$.*
2. *In the second half of stage s, the path Q_i^x touches (from left) but does not cross the point $\beta_s = n - 1 + \sum_{r=1}^{s}(b_r - a_r)$.*
3. *The path Q_i^x ends at $\gamma = j + \sum_{s=1}^{l}(b_s - a_s)$.*

We let $Y_{i\gamma}^{(t),Q}[\{\alpha_s, \beta_s\}_{s=1}^l]$ contain those $x \in \{0,1\}^t$ satisfying the three properties in the above observation, and denote by $m_{i\gamma}^{(t),Q}[\{\alpha_s, \beta_s\}_{s=1}^l]$ the size of the set $Y_{i\gamma}^{(t),Q}[\{\alpha_s, \beta_s\}_{s=1}^l]$. Thus, according to the observation, $X_{ij}^{(t),P}[\{\alpha_s, \beta_s\}_{s=1}^l] \subseteq Y_{ij}^{(t),Q}[\{\alpha_s, \beta_s\}_{s=1}^l]$, and therefore we have $n_{ij}^{(t),P}[\{a_s, b_s\}_{s=1}^l] \le m_{i\gamma}^{(t),Q}[\{\alpha_s, \beta_s\}_{s=1}^l]$. So it is enough to upper bound $m_{i\gamma}^{(t),Q}[\{\alpha_s, \beta_s\}_{s=1}^l]$.

Now for each $x \in Y_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l]$, if we change property 1 in stage $s=1$ by allowing the path to cross the point α_1 (and keep properties 2 and 3 unchanged), and let $Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l]$ be the new set satisfying the properties, then $m_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l] = |Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l]| - |Z_{i\gamma}^{(t),Q}[\alpha_1-1,\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l]|$. In other words, the set of paths that touch from right but do not cross α_1 is the set of paths that touch or cross α_1 minus the set of paths that touch or cross α_1-1.

Now we calculate $|Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l]|$ by the so-called reflection rule. Suppose the first time that Q_i^x touches α_1 is t_1. We reflect the first t_1 part of the path Q_i^x with respect to the point α_1. Precisely, let $y=(1-x_1)\cdots(1-x_{t_1})x_{t_1+1}\cdots x_t$; then the paths Q_i^x and $Q_{2\alpha_1-i}^y$ merge at time t_1. And it is easy to check that there is a 1-1 correspondence between $Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l]$ and $Y_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l]$. Here $Y_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l]$ is the set of paths starting at $2\alpha_1-i$, satisfying (a) property 2 at the first stage, (b) both properties 1 and 2 at the remaining $l-1$ stages, and (c) property 3. So

$$|Z_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l]| = |Y_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l]| = m_{2\alpha_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l] \tag{75}$$

$$= m_{-2a_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l] \tag{76}$$

$$= m_{-a_1-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l], \tag{77}$$

where (76) is due to the fact that $\alpha_1=-a_1$, and (77) holds because the number of the paths does not change if we move all the paths right by a_1. Similarly, we have

$$|Z_{i\gamma}^{(t),Q}[\alpha_1-1,\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l]| = m_{2\alpha_1-2-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l] \tag{78}$$

$$= m_{-a_1-2-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l]. \tag{79}$$

Therefore,

$$n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^l] \le m_{i\gamma}^{(t),Q}[\{\alpha_s,\beta_s\}_{s=1}^l] \tag{80}$$

$$= m_{-2a_1-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l] - m_{-2a_1-2-i,\gamma}^{(t),Q}[\beta_1,\{\alpha_s,\beta_s\}_{s=2}^l] \tag{81}$$

$$= m_{-a_1-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l] \tag{82}$$

$$- m_{-a_1-2-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l]. \tag{83}$$

Note that $\alpha_s+a_1 = b_1+\sum_{r=2}^{s-1}(b_r-a_r)-a_s$, $\beta_s+a_1 = n-1+b_1+\sum_{r=2}^{s}(b_r-a_r)$, and $\gamma+a_1 = j+b_1+\sum_{r=2}^{s}(b_r-a_r)$ are all functions of $(b_1,a_2,b_2,\cdots,a_l,b_l)$, not of a_1 any longer. Therefore,

$$\sum_{a_1,b_1,\cdots,a_l,b_l>0} n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^l] \tag{84}$$

$$\le \sum_{b_1,\cdots,a_l,b_l>0}\sum_{a_1>0}(m_{-a_1-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l] \tag{85}$$

$$- m_{-a_1-2-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l]) \tag{86}$$

$$= \sum_{b_1,\cdots,a_l,b_l>0}(m_{-1-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l] \tag{87}$$

$$+ m_{-2-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l]) \tag{88}$$

$$\le \sum_{b_1,\cdots,a_l,b_l>0} 2\max_{h=1,2}\{m_{-h-i,\gamma+a_1}^{(t),Q}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^l]\}. \tag{89}$$

Now using the similar methods, i.e., reflecting with respect to points $(n-1+b_1)$ and $(n+b_1)$, moving the paths left by b_1, and finally collapsing the telescope, we can get

$$\sum_{b_1,\cdots,a_l,b_l>0} m^{(t),Q}_{-h-i,\gamma+a_1}[\beta_1+a_1,\{\alpha_s+a_1,\beta_s+a_1\}_{s=2}^{l}] \tag{90}$$

$$\leq \sum_{a_2,b_2,\cdots,a_l,b_l>0} 2\max_{k=1,2}\{m^{(t),Q}_{2n+i+h-k+1,\gamma+a_1-b_1}[\{\alpha_s+a_1-b_1,\beta_s+a_1-b_1\}_{s=2}^{l}]\} \tag{91}$$

and thus

$$\sum_{a_1,b_1,\cdots,a_l,b_l>0} n^{(t),P}_{ij}[\{a_s,b_s\}_{s=1}^{l}] \tag{92}$$

$$\leq \sum_{a_2,b_2,\cdots,a_l,b_l>0} 4\max_{h=0,1,2}\{m^{(t),Q}_{2n+i+h,\gamma+a_1-b_1}[\{\alpha_s+a_1-b_1,\beta_s+a_1-b_1\}_{s=2}^{l}]\}. \tag{93}$$

We continue this process, and finally we get

$$\sum_{a_1,b_1,\cdots,a_l,b_l>0} n^{(t),P}_{ij}[\{a_s,b_s\}_{s=1}^{l}] \leq 2^{2l}\max_{h=0,1,\cdots,2l} n^{(t),Q}_{2ln+i+h,\gamma+\sum_{s=1}^{l}(a_s-b_s)} \tag{94}$$

$$= 2^{2l}\max_{h=0,1,\cdots,2l} n^{(t),Q}_{2ln+i+h,j} \tag{95}$$

$$= 2^{2l} n^{(t),Q}_{2ln+i,j} \tag{96}$$

$$\leq 2^{2l}\binom{t}{\frac{t}{2}+\frac{j-i-2ln}{2}}. \tag{97}$$

Thus

$$\sum_{l>0}\sum_{a_1,b_1,\cdots,a_l,b_l>0} n^{(t),P}_{ij}[\{a_s,b_s\}_{s=1}^{l}] \tag{98}$$

$$\leq \sum_{l\geq 0} 2^{2(l+1)}\binom{t}{t/2+ln} \tag{99}$$

$$= 4\binom{t}{t/2}+\sum_{l\geq 1} 2^{2(l+1)}\binom{t}{t/2+ln} \tag{100}$$

$$\leq 4\binom{t}{t/2}+\frac{1}{n}\sum_{l\geq 1} 2^{2(l+1)}\left(\binom{t}{t/2+ln}+\binom{t}{t/2+ln-1}+\cdots+\binom{t}{t/2+ln-n+1}\right) \tag{101}$$

$$\leq 4\binom{t}{t/2}+\frac{1}{n}\sum_{l\geq 1} 2^{2(l+1)}\left(\binom{t}{t}+\binom{t}{t-1}+\cdots+\binom{t}{t/2+ln-n+1}\right) \tag{102}$$

$$\leq 4\binom{t}{t/2}+\frac{1}{n}\sum_{l\geq 1} 2^{2(l+1)}2^t e^{-\frac{2(l-1)^2n^2}{3t}}, \tag{103}$$

where $\binom{t}{t'} = 0$ if $t' > t$. Here the first two inequalities are by the monotonicity of binomial coefficients, and the last inequality is by Chernoff's bound. Now if $t \le n^2$, then $\sum_{l\ge 1} 2^{2(l+1)} e^{-\frac{2(l-1)^2 n^2}{3t}} \le \sum_{l\ge 1} 2^{2(l+1)} e^{-\frac{2(l-1)^2}{3}} = O(1)$. Thus it holds that $\sum_{l>0} \sum_{a_1,b_1,\cdots,a_l,b_l>0} n_{ij}^{(t),P}[\{a_s,b_s\}_{s=1}^{l}] \le O(\binom{t}{t/2} + 2^t/n) = O(2^t/\sqrt{t})$. For other categories of $a_1 = 0$ or $b_l = 0$, the same result can be proved similarly, and the $l = 0$ is easy since $n_{ij}^{(t),Q} = O(2^t/\sqrt{t})$. Putting all of these things together, we see that $p_{ij}^{(t)} = O(1/\sqrt{t})$ if $t \le n^2$. The other part, i.e., $p_{ij}^{(t)} = O(1/n)$ when $t > n^2$, can be easily derived from this and the fact that $\max_{ij} p_{ij}^{(t)}$ decreases as t increases. This completes our proof. □

Now we use Proposition 4.2 to prove the weaker lower bounds for grids. Note that the random walk $([n]^m, v_0, W)$ is just a product of m line walks, i.e., cyclicly performing the line walk in the cyclic order of dimension $0, 1, \cdots, m-1$ (see (72)). Therefore, the p_t in the random walk $([n]^m, v_0, W)$ satisfies

$$p_t = \begin{cases} O(1/\sqrt{t^m}) & \text{if } t \le n^2, \\ O(1/n^m) & \text{if } t > n^2. \end{cases} \tag{104}$$

Now for the randomized lower bounds, when $d > 4$ we pick $m = \lceil d/2 \rceil > 2$ and we get

$$RLS([n]^d) = \Omega\left(\frac{n^{d-m}}{O(1) + n^{d-m}/n^m}\right) = \Omega(n^{\lfloor d/2 \rfloor}) = \begin{cases} \Omega(n^{\frac{d}{2}}) & \text{if } d \text{ is odd}, \\ \Omega(n^{\frac{d}{2}-\frac{1}{2}}) & \text{if } d \text{ is even}. \end{cases} \tag{105}$$

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

$$QLS([n]^d) = \Omega\left(\frac{n^{d-m}}{O(1) + n^{d-m}/n^{m/2}}\right) = \begin{cases} \Omega(N^{\frac{1}{3}}) & \text{if } d = 3d', \\ \Omega(N^{\frac{1}{3}-\frac{1}{3d}}) & \text{if } d = 3d'+1, \\ \Omega(N^{\frac{1}{3}-\frac{1}{6d}}) & \text{if } d = 3d'+2. \end{cases} \tag{106}$$

For $d = 6$, we let $m = 4$ and we have $QLS([n]^6) = \Omega(n^2/\log n)$. For $d = 5, 4, 3$, we let $m = d-2$, and 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. Getting around the integer constraint for dimension. 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 graphs $[n]$. 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, there is just one step of

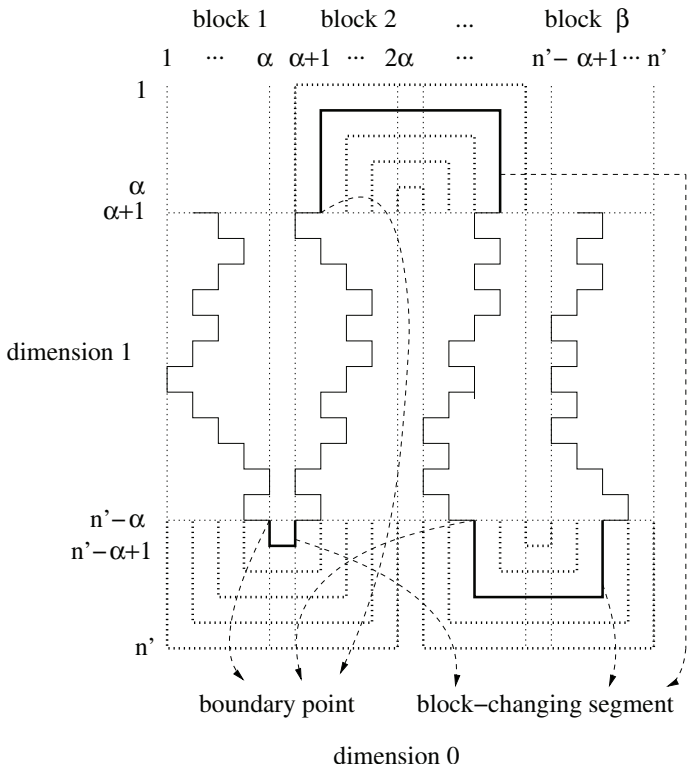

FIG. 2. *Illustration for changing a block in the* 2*-dimensional grid.*

random walk in $[n^r]^{d-1}$ followed by two steps of one-way walk in the last dimension space $[n]$. When the clock $[n]$ is exhausted, 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 an $[n^r]^{d-1} \times [L]$ 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.) Figure 2 is an illustration for the case of $d = 2$.

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

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 $[n']^{d-1} = \{x_0 \cdots x_{d-2} : x_i \in [n']\}$. We cut V_1 into β^{d-1} parts $\{x_0 \cdots x_{d-2} : (k_i - 1)\alpha < x_i \leq k_i\alpha\}_{k_0 \cdots k_{d-2} \in [\beta]^{d-1}}$, each of which is a small grid isomorphic to $[\alpha]^{d-1}$. We then refer to the set $\{x_0 \cdots x_{d-2}x_{d-1} : (k_i - 1)\alpha < x_i \leq k_i\ \alpha,\ i = 0, \ldots, d-2,\ \alpha < x_{d-1} \leq n' - \alpha\}$ as the "*block* $(k_0, \cdots, k_{d-2})$." Note that $(k_0, \cdots, k_{d-2})$ also can be 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 $(k'_0, \cdots, k'_{d-2})$ *the next block* of the block $(k_0, \cdots, k_{d-2})$ if $(k'_0, \cdots, k'_{d-2})$, viewed as the point in $[\beta]^{d-1}$, is the next point of $(k_0, \cdots, k_{d-2})$ in $HamPath_{\beta,d-1}$. Note that by our definition of $HamPath_{\beta,d-1}$, we know that $\exists i \in \{0, \cdots, d-2\}$ such that $k'_i \in \{k_i + 1, k_i - 1\}$ and for all other $j \neq i$, $k'_j = k_j$. That is, adjacent blocks have

only one different coordinate, and the difference is 1. We call the block $(k_0, \cdots, k_{d-2})$ *the last block* if $(k_0, \cdots, k_{d-2})$ 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 circularly next dimension), followed by two steps of walk in the clock space. If the clock is exhausted, we say we reach *a boundary point* at the current block, and we move to the next block via a path segment called a *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 $x_0 \cdots x_{d-1}$ 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 $(k_0, \cdots, k_{d-2})$ with the offsets $(y_0, \cdots, y_{d-1})$. Thus the instruction $x_0 = x_0 + 1$, for example, means that the particle moves from $x_0 \cdots x_{d-1}$ to $(x_0 + 1)x_1 \cdots x_{d-1}$.

1. Initially $x_0 = \cdots = x_{d-2} = 0$, $x_{d-1} = \alpha + 1$, $k_0 = \cdots = k_{d-2} = 1$.
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. Also note that we do not let the clock tick when we are moving from one block to another. Thus the block-changing segments thread all the blocks to form an $[\alpha]^{d-1} \times [L]$ grid, where $L = (n' - 2\alpha)\beta^{d-1}$. Actually, for our lower bound purpose, we can think of the random walk as performed in the product graph $[\alpha]^{d-1} \times [L]$. We will make this clearer below.

What we care about is, as before, the probability that the random walk starting from a point $x = x_0 \cdots x_{d-1}$ passes another point $x' = x'_0 \cdots x'_{d-1}$. Note that for any point x (including those on the block-changing segments), there is only one time t when the walk may hit x, and this t is determined by x itself. Similarly we use t' to denote the time when the path passes x'. Denote the probability that the random walk starting from x passes x' by $\mathbf{Pr}[x \to x']$. As before suppose $x_i = (k_i - 1)\alpha + y_i$ and $x'_i = (k'_i - 1)\alpha + y'_i$ for $i \in \{0, \cdots, d-2\}$.

We first consider the case that one of the two points, say x', is on a block-changing segment. Since different block-changing segments never intersect, a path passes x' if and only if the path passes the boundary point x'' at the beginning of the block-changing segment that x' is in. Also note that the time when the path passes x'' is also t' because the time does not elapse on the block-changing segment. So we have

that $\mathbf{Pr}[x \to x'] = \mathbf{Pr}[x \to x'']$, and it is enough to consider the case that both x and x' are not in block-changing segments.

Now suppose both x and x' are not in block-changing segments. In general, x and x' may be not in the same block , so going from x to x' needs to change blocks. Recall that to change from the block $(k_0, \cdots, k_{d-2})$ to the next one, only one k_i changes by increasing or decreasing by 1. Suppose that to go to x' from x, we change blocks for c times, by changing $k_{i_1}, k_{i_2}, \cdots, k_{i_c}$ in turn. Let $n_j = |\{s \in [c] : i_s = j\}|$. Note that to get to x' from x after $t' - t$ steps, the coordinate j needs to be x'_j after $t' - t$ steps for each coordinate $j \in \{0, \cdots, d-2\}$. It is not hard to see that if a block-changing needs to change k_j by increasing $b \in \{-1, 1\}$, then among all the offsets y_i's, only the y_j gets changed, and the change is a reflection within the block. That is, suppose x_j is $(k_j - 1)\alpha + y_j$ before the block-changing; then x_j changes to $(k_j + b - 1)\alpha + (\alpha + 1 - y_j)$ after the block-changing. So if $c = 1$, then $\mathbf{Pr}[x \to x']$ is equal to the probability that a random walk in $[\alpha]^{d-1}$ starting from $y_0 \cdots y_{d-2}$ hits $y''_0 \cdots y''_{d-2}$ after exactly $t' - t$ steps, where $y''_j = y'_j$ if $j \neq i_1$ and $y''_{i_1} = \alpha + 1 - y'_{i_1}$. For general c, $\mathbf{Pr}[x \to x']$ is equal to the probability that a random walk in $[\alpha]^{d-1}$ starting from $y_0 \cdots y_{d-2}$ hits $y''_0 \cdots y''_{d-2}$ after exactly $t' - t$ steps, where $y''_j = y'_j$ if n_j is even and $y''_j = \alpha + 1 - y'_j$ if n_j is odd. Note that this probability has nothing to do with the block-changing; it is just the same as if we had a clock space $[(n' - 2\alpha)\beta^{d-1}]$ to record the random walk on $[\alpha]^{d-1}$. Thus we can use Proposition 4.2 to upper bound this probability and just think of the graph as $[n^r]^{d-1} \times [L]$ and use Theorem 3.3, with $G_1 = [n^r]^{d-1}$ and $G_2 = [L]$.

Now we have $T = \lfloor L/2 \rfloor$ and $p_t = O(1/\sqrt{t^{d-1}})$ for $t \leq n^{2r}$ and $p_t = O(1/n^{r(d-1)})$ for $t > n^{2r}$. So for randomized lower bounds, if $d \geq 4$, then we let $r = d/(2d-2)$ and get

(107)
$$RLS([n]^d) = \Omega\left(n^{1+(1-r)(d-1)} / \left(\sum_{t=1}^{n^{d/(d-1)}} \frac{1}{\sqrt{t^{d-1}}} + \frac{n^{1+(1-r)(d-1)}}{n^{r(d-1)}}\right)\right) = \Omega\left(n^{d/2}\right).$$

If $d = 3$, we let $r = 3/4 - \log\log n/(4\log n)$ and get $RLS([n]^3) = \Omega((n^3/\log n)^{1/2})$. For $d = 2$, we let $r = 2/3$ and get $RLS([n]^2) = \Omega(n^{2/3})$.

For the quantum lower bounds, if $d \geq 6$, then we let $r = 2d/(3d-3)$ and get

(108)
$$QLS([n]^d) = \Omega\left(n^{1+(1-r)(d-1)} / \left(\sum_{t=1}^{n^{d/(d-1)}} \frac{1}{t^{(d-1)/4}} + \frac{n^{1+(1-r)(d-1)}}{n^{r(d-1)/2}}\right)\right) = \Omega(n^{d/3}).$$

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 $QLS([n]^d) = \Omega(n^{d/2 - d/(d+1)})$, which is $\Omega(n^{1/3})$, $\Omega(n^{3/4})$, $\Omega(n^{6/5})$ for $d = 2, 3, 4$, respectively.

4.2.3. Further improvement on 2-dimensional grid $[n]^2$. Some other random walks may be used to further improve the lower bound on low-dimensional grid cases. Here is one way to improve $QLS([n]^2)$ from $\Omega(n^{1/3})$ to $\Omega(n^{2/5})$. We cut the graph $[n]^2$ into $n^{2/5}$ smaller grids, each of size $n^{4/5} \times n^{4/5}$. Without loss of generality, assume both $n^{1/5}$ and $n^{4/5}$ are integers and further assume $n^{1/5} = 3 \bmod 4$; otherwise we can consider a slightly smaller grid by the simple trick as at the beginning of section 4.2.2. We shall use a random walk similar to Aaronson's in [2] in each block and change blocks after each step. Thus different blocks record different times.

For any time $t \in [n^{1/5}(n^{1/5} - 1)]$, suppose $t = 2rn^{1/5} + t'$, where $r \in \{0, 1, \cdots,$

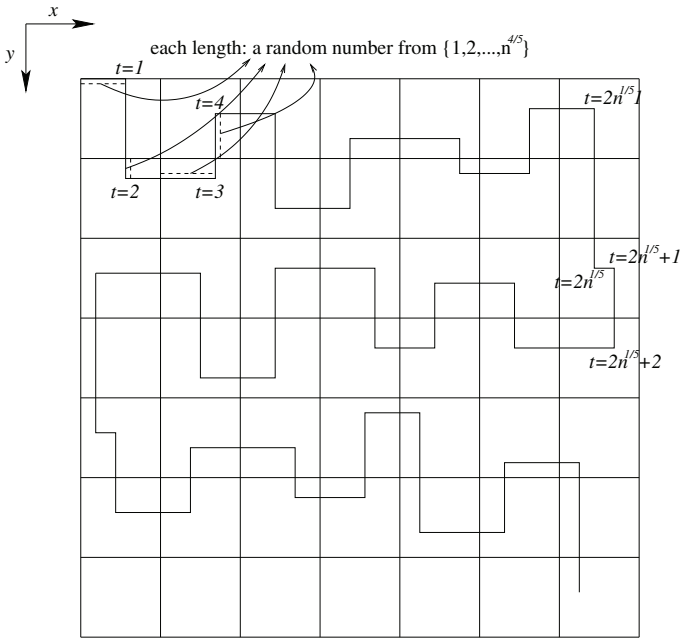

FIG. 3. *A different random walk in the* 2*-dimensional grid.*

$(n^{1/5}-3)/2\}$ and $t' \in \{1, 2, \cdots, 2n^{1/5}\}$. Let

$$u = \begin{cases} 0 & \text{if } t' \equiv 0, 1 \pmod 4, \\ n^{4/5} & \text{if } t' \equiv 2, 3 \pmod 4. \end{cases} \tag{109}$$

Let $block(t)$ be the small grid

$$\begin{cases} \{((\lceil t'/2 \rceil - 1)n^{4/5} + x', 2rn^{4/5} + u + y') : x', y' \in [n^{4/5}]\} & \text{if } r \text{ is even}, \\ \{((n^{1/5} - \lceil t'/2 \rceil)n^{4/5} + x', 2rn^{4/5} + u + y') : x', y' \in [n^{4/5}]\} & \text{if } r \text{ is odd}. \end{cases} \tag{110}$$

In either case, the (x', y') is called the *offset* of the corresponding element in the block. Now define the random walk as follows and as depicted in Figure 3.

Initially $(x, y) = (1, 1)$
for $t = 1, 2, \ldots, n^{1/5}(n^{1/5} - 1)$ (suppose the current point is (x, y) with offset (x', y'))
if t' is odd,
pick a random $x'' \in [n^{4/5}]$, move horizontally to the point in $block(t)$ with the offset (x'', y')
else
if $t' = 2n^{1/5}$, **then** c = 1 **else** c = 0
pick a random $y'' \in [n^{4/5}]$, move vertically to the point in $block(t + c)$ with the offset (x', y'').
We then follow the same track as in the proof of Theorem 3.3. To get a reduction

from Local Search on $[n]^2$ to the $Path_P$ problem, we define the function

$$f_X(v) = \begin{cases} l_{[n]^2}(v,(1,1)) & \text{if } v \notin set(X), \\ -2n^{4/5}(t-1) - (-1)^r x'_v + (-1)^{\lceil t'/2 \rceil} y'_v & \text{if } v \in set(X) \cap block(t). \end{cases} \tag{111}$$

Intuitively, the function value decreases along the path as before. But the decrement is not always by 1: each block has its fixed value setting. If, for example, the path passes through the block toward the right and down (as in the first block), then the value $-x' - y'$ is used within the block. In this way, we do not need to know the length of the path segment from top to v to calculate each $f_X(v)$.

What we care about is still, as in equality (39), the probability that the path X' passes another point x on X, under the condition that $X' \wedge Y = k'$. It is not hard to see that this probability is $\Theta(1)$ in general if x is in $block(k')$, and $\Theta(1/n^{4/5})$ otherwise (i.e., when x is in $block(t)$ for some $t > k'$). Thus by $L = \Theta(n^{2/5})$ we have

$$QLS([n]^2) = \Omega\left(n^{2/5} / \left(1 + n^{2/5}/\sqrt{n^{4/5}}\right)\right) = \Omega(n^{2/5}). \tag{112}$$

This completes the proof of Theorem 1.3.

Note that this random walk suffers from the fact that the "passing probability" is now $n^{4/5}$ times the "hitting probability." So for general d, we can get $RLS([n]^d) = \Omega(n^{d/(d+1)})$ and $QLS([n]^d) = \Omega(n^{d/(2d+1)})$, which only gives better results for QLS on the 2-dimensional grid.

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. Pick k random samplings over all the vertices, and find a vertex v in them with the minimum f-value.[7] Then roughly speaking, v is the N/k-minimum vertex over all the N vertices in G. Now 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. Since v is the N/k-minimum vertex, any decreasing path from it has length no more than N/k. Thus we need $k + \delta N/k$ queries in the randomized case and $\sqrt{k} + \sqrt{\delta} N/k$ queries in the quantum case, where δ is the maximum degree of the graph G. Optimizing k achieves the performance of the algorithms mentioned in section 1. 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 samples + steepest descent" method when the graph expands slowly. Here the idea is that after finding the minimum vertex v of the k sampled points, we know that v is (roughly) the N/k-minimum vertex. Therefore, there must be a local optimum within the smaller range $\{u : l_G(u,v) \le N/k\}$. So instead of following the decreasing path of v, we do the local search within this smaller range recursively.

While this idea sounds simple and effective, there is one caveat here: A local minimum u in the smaller range may be not a local minimum in the original larger graph G, because u may have more neighbors in G. To deal with this difficulty, we will actually solve a stronger version of the local search problem: On the graph G, given

[7] For the minimum f-value finding procedure, the randomized algorithm in [4] just queries all these vertices and finds the minimum, while the quantum algorithm in [2] uses the algorithm by Dürr and Høyer [11] based on Grover search [12] to get a quadratic speedup.

a function $f : V \to \mathbb{N}$ and a vertex v, find a local optimum u such that $f(u) \le f(v)$. Note that such u must exist; any decreasing path from v leads to a valid u. Also note that this problem is harder than the original local search problem; any algorithm for this new problem is also an algorithm for the original one. A key property of this new problem is that it allows a recursion: Given a small range S_{small} ($\subseteq S_{large} \subseteq V$) and a vertex v in it, suppose that any vertex w on the boundary[8] of S_{small} is worse than v (i.e., $f(w) > f(v)$); then any local optimum u in S_{small} satisfying $f(u) \le f(v)$ is also a local optimum in a larger range S_{large}.

Now we describe the algorithm precisely, with some notation 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 $B(S) = \{u \in S : \exists v \in V - S$ such that $(u, v) \in E\}$. In particular, $B(V) = \emptyset$. A decreasing path from a vertex $v \in V$ is a sequence of vertices $v_0, v_1, \cdots, v_k$ 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 $i = 0, \ldots, k-1$. We write $f(u) \le f(S)$ if $f(u) \le f(v)$ for all $v \in S$. In particular, it always holds that $f(u) \le f(\emptyset)$. Suppose $d = \max_{u,v\in V} l(u, v)$ is the diameter of the graph, and $\delta = \max_{v\in V} |\{u : (u, v) \in E\}|$ is the maximum 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. $m_0 = d$, $U_0 = V(G)$;
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. — $O\Big(\frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1}\Big)$
   - Quantum algorithm: use Dürr and Høyer's algorithm [11] with the error probability at most $\epsilon_2 = 1/(10 \log_2 d)$. — $O\Big(\sqrt{\frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1}} \log \frac{1}{\epsilon_2}\Big)$

   (c) **if** $i = 0$, **then** $u_{i+1} = v_i$;
   else if $f(u_i) \le f(v_i)$, **then** $u_{i+1} = u_i$;
   else $u_{i+1} = v_i$;
   (d) **for** $j = 1, 2, \ldots$
   i. Randomly pick $m_{ij} \in M_i = \{m : m_i/8 \le m \le m_i/2,\ |W(m)| \le 10|U_i|/m_i\}$, where $W(m) = \{w \in U_i : l(w, u_{i+1}) = m\}$. Let $W_{ij} = W(m_{ij})$.
   ii. Test whether $f(u_{i+1}) \le f(W_{ij})$.
   - Randomized algorithm: query all vertices in W_{ij}. — $O(|W_{ij}|)$
   - Quantum algorithm: use Dürr and Høyer'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 3(e).
   (e) $J_i = j$, $m_{i+1} = m_{ij}$, $W_i = W_{ij}$, $U_{i+1} = \{u \in U_i : l(u, u_{i+1}) \le m_{i+1}\}$;
   (f) $i = i + 1$;

[8] The boundary is the set of those vertices in S_{small} that have neighbors in $S_{large} - S_{small}$.

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 Dürr and Høyer's algorithm with the error probability at most 1/100. — $O(\sqrt{\delta})$

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

THEOREM 5.1. *The algorithm outputs a local minimum with probability at least* 1/2. *The randomized algorithm uses* $O\big(\sum_{i=0}^{\lceil \log_2 d \rceil - 1} \frac{c(m_i)}{m_i} \log\log d\big)$ *queries in expectation, and the quantum algorithm uses* $O\big(\sum_{i=0}^{\lceil \log_2 d \rceil - 1} \sqrt{\frac{c(m_i)}{m_i}} (\log\log d)^{1.5}\big)$ *queries in expectation.*

In the case that $c(k) = O(k^\alpha)$ *(*α *may be a function of* n*) for some* $\alpha \ge 1$ *and* $k = 1, \dots, d$, *the expected number of queries that the randomized algorithm uses is* $O\big(\frac{d^{\alpha-1}-1}{1-2^{1-\alpha}} \log\log d\big)$ *if* $\alpha > 1$ *and* $O(\log d \log\log d)$ *if* $\alpha = 1$. *The expected number of queries that the quantum algorithm uses is* $O\big(\frac{d^{\frac{\alpha-1}{2}}-1}{1-2^{\frac{1-\alpha}{2}}} (\log\log d)^{1.5}\big)$ *if* $\alpha > 1$ *and* $O(\log d \log\log d)$ *if* $\alpha = 1$.

We make 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 $QLS(G) = O\big(\sqrt{\frac{N}{d}} (\log\log d)^{1.5}\big)$.
3. For a 2-dimensional grid, $d = \Theta(n)$ and $\alpha = 2$. Thus Corollary 1.5 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; the theorem then follows easily.

CLAIM 3. *For each* $i = 0, 1, \dots, I-1$, *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* $\mathbf{E}[J_i] \le 2$.[9]
3. *If* $W_0, \cdots, W_i$ *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 steps 3(a)–3(c), denote by S the set of the $\lceil \frac{8|U_i|}{m_i} \log \frac{1}{\epsilon_1} \rceil$ sampled vertices in step 3(a). 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 3(b) achieves the minimum in the definition of a with probability at least $1 - \epsilon_2$. Putting 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 3(c)), $U_{i+1} \subseteq U_i$ (by step 3(e)), 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 $1 - \epsilon_1 - \epsilon_2$.

[9] Since J_i is a random variable, the meaning of "W_i is good with probability $1 - \epsilon_3 J_i$" is that for each fixed $j = 1, 2, \dots$, we have that W_i is good with probability $1 - \epsilon_3 j$ under the condition that $J_i = j$. Similar language is also used in the latter part of the proof. Finally we will upper bound the probability of these random variables being large, and in the case that they are small, the error probability is small. This implies that the total error probability is small.

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 $W_{ij} \cap W_{ij'} = \emptyset$ if $m_{ij} \neq m_{ij'}$. 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 such that $|W(m_{ij})| > 10|U_i|/m_i$ 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 Dürr–Høyer 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 $\exists t \in V - U_{i+1}$ such that $l(s,t) = 1$. 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 $l(t, u_{i+1}) > m_{i+1}$ by the definition of U_{i+1}. Noting that $l(s, u_{i+1}) \le m_{i+1}$ since $s \in U_{i+1}$, and that $l(s,t) = 1$, we have $l(s, u_{i+1}) = m_{i+1}$, 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 $B(U_{i+1}) \subseteq B(U_0) \cup W_0 \cup \cdots \cup W_i = W_0 \cup \cdots \cup W_i$. Since we have $f(u_{i+1}) \le f(u_i) \le \cdots \le f(u_1)$ (by step 3(c)) and $f(u_{k+1}) \le f(W_k)$ (for $k = 0, \ldots, i$) by the assumption that all W_k's are good, we know that $f(u_{i+1}) \le f(W_0 \cup \cdots \cup W_i)$, 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 $k = 0, \ldots, i$ 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 3(c). 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 step 3), which implies that $u_{k+1} \notin B(U_k)$. □

(Continue the proof of Theorem 5.1.) Now by Claim 3, 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

$$n(u_I, U_I) \le m_I, \qquad f(u_I) \le f(B(U_I)), \qquad u_I \notin B(U_I). \tag{113}$$

Note that the correctness of the algorithm 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 $(u_I^0, u_I^1, \cdots, u_I^T)$ 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 $f(u_I) \le f(B(U_I))$. 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 $J = \sum_{i=0}^{I-1} J_i$. The last summand comes from the error of step 5; note that the decreasing path is of length at most 10. Since $\mathbf{E}[J] \le 2I$, we know by Markov inequality that $J < 20I$ 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 noting that $I \le \log_2 d$ because $m_0 = d$ and $m_{i+1} \le \lceil m_i/2 \rceil$, the total error probability is less than $1/2$.

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

$$O\left(\sqrt{\frac{8|U_i|}{m_i}\log\log d}\log\log d\right) = O\left(\sqrt{\frac{c(m_i)}{m_i}}(\log\log d)^{1.5}\right) \tag{114}$$

queries. Also note from step 3(d)i that $|W_{ij}| \le 10|U_i|/m_i$; therefore, step 3(d) uses $O(\sum_{j=1}^{J_i} \sqrt{c(m_i)/m_i} \log\log d)$ queries, which is $O(\sqrt{c(m_i)/m_i} \log\log d)$ in expectation. 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

$$O\left(\left(\sum_{i=0}^{\log_2 d-1} \sqrt{c(m_i)/m_i}\right) (\log\log d)^{1.5}\right). \tag{115}$$

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

$$\sum_{i=0}^{\log_2 d-1} \sqrt{\frac{c(m_i)}{m_i}} = \sum_{i=0}^{\log_2 d-1} m_i^{(\alpha-1)/2} = \sum_{i=0}^{\log_2 d-1} (d/2^i)^{(\alpha-1)/2} = \frac{d^\beta - 1}{1 - 2^{-\beta}}, \tag{116}$$

where $\beta = (\alpha - 1)/2$. Finally the cost of step 5 is $O(\sqrt{\delta})$ since the length of the path is at most 10. This completes the proof for the quantum algorithm, except that in the case of $\alpha = 1$ we have only a quantum upper bound of $O(\log d(\log\log d)^{1.5})$. But note that the randomized algorithm uses $O(\log d \log\log d)$ queries (because of the savings in error probability controls). So when $\alpha = 1$, the quantum algorithm uses just the randomized one. □

6. Open problems and future directions. An immediate open question is to close the remaining gaps for the low-dimensional grids. As mentioned in section 1, three of the four gaps were almost closed by Sun and Yao recently [22]; the only remaining one is $QLS([n]^4)$.

A probably more important question is the characterization of the deterministic, randomized, and quantum query complexities of Local Search on graph G. Are the query complexities determined by some simple characteristic of the graph G? If yes, is it the expansion?

Acknowledgments. The author thanks Scott Aaronson, Xiaoming Sun, and Andy Yao for many valuable discussions. Thanks also to Alexander Razborov and Nicholas Pippenger for carefully reading the manuscript and giving many detailed suggestions, and to Sean Hallgren, Martin Roetteler, and Pranab Sen for listening to a presentation of the work and offering useful comments. The author is also indebted to Yves Verhoeven and Dirk Winkler for each pointing out an error in a preliminary version of the paper.

REFERENCES

[1] K. Aardal, S. Hoesel, J.K. Lenstra, and L. Stougie, *A decade of combinatorial optimization*, CWI Tracts, 122 (1997), pp. 5–14.

[2] S. Aaronson, *Lower bounds for local search by quantum arguments*, in Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, 2004, pp. 465–474.

[3] E. Aarts and J. Lenstra, *Local Search in Combinatorial Optimization*, John Wiley & Sons, New York, 1997.

[4] D. Aldous, *Minimization algorithms and random walk on the d-cube*, Ann. Probab., 11 (1983), pp. 403–413.

[5] I. Althofer and K. Koschnich, *On the deterministic complexity of searching local maxima*, Discrete Appl. Math., 43 (1993), pp. 111–113.

[6] A. Ambainis, *Polynomial degree vs. quantum query complexity*, in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, 2003, pp. 230–239.

[7] A. Ambainis, *Quantum lower bounds by quantum arguments*, J. Comput. System Sci., 64 (2002), pp. 750–767.
[8] H. Barnum, M. Saks, and M. Szegedy, *Quantum query complexity and semidefinite programming*, in Proceedings of the 18th Annual IEEE Conference on Computational Complexity, 2003, pp. 179–193.
[9] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, *Quantum lower bounds by polynomials*, J. ACM, 48 (2001), pp. 778–797.
[10] H. Buhrman and R. de Wolf, *Complexity measures and decision tree complexity: A survey*, Theoret. Comput. Sci., 288 (2002), pp. 21–43.
[11] C. Dürr and P. Høyer, *A Quantum Algorithm for Finding the Minimum*, 1996; available from http://www.arxiv.org/abs/quant-ph/9607014.
[12] L. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, 1996, pp. 212–219.
[13] P. Høyer and R. Špalek, *Lower bounds on quantum query complexity*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS, 87 (2005), pp. 78–103.
[14] D. Johnson, C. Papadimitriou, and M. Yannakakis, *How easy is local search?*, J. Comput. System Sci., 37 (1988), pp. 79–100.
[15] S. Laplante and F. Magniez, *Lower bounds for randomized and quantum query complexity using Kolmogorov arguments*, in Proceedings of the 19th Annual IEEE Conference on Computational Complexity, 2004, pp. 294–304.
[16] D. Llewellyn and C. Tovey, *Dividing and conquering the square*, Discrete Appl. Math., 43 (1993), pp. 131–153.
[17] D. Llewellyn, C. Tovey, and M. Trick, *Local optimization on graphs*, Discrete Appl. Math., 23 (1989), pp. 157–178; Erratum, 46 (1993), pp. 93–94.
[18] N. Megiddo and C. Papadimitriou, *On total functions, existence theorems, and computational complexity*, Theoret. Comput. Sci., 81 (1991), pp. 317–324.
[19] J. B. Orlin, A. P. Punnen, and A. S. Schulz, *Approximate local search in combinatorial optimization*, SIAM J. Comput., 33 (2004), pp. 1201–1214.
[20] M. Santha and M. Szegedy, *Quantum and classical query complexities of local search are polynomially related*, in Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, 2004, pp. 494–501.
[21] R. Špalek and M. Szegedy, *All quantum adversary methods are equivalent*, Theory Comput., 2 (2006), pp. 1–18.
[22] X. Sun and A. Yao, *On the quantum query complexity of local search in two and three dimensions*, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 429–438.
[23] Y. Verhoeven, *Enhanced algorithms for local search*, Inform. Process. Lett., 97 (2006), pp. 171–176.
[24] S. Zhang, *On the power of Ambainis lower bounds*, Theoret. Comput. Sci., 339 (2005), pp. 241–256.