Quantum and Classical Query Complexities of Local Search are Polynomially Related

Miklos Santha* Mario Szegedy†

Abstract

Let f be an integer valued function on a finite set V. We call an undirected graph $G(V, E)$ a *neighborhood structure* for f. The problem of finding a local minimum for f can be phrased as: for a fixed neighborhood structure $G(V, E)$ find a vertex $x \in V$ such that $f(x)$ is not bigger than any value that f takes on some neighbor of x. The complexity of the algorithm is measured by the number of questions of the form "what is the value of f on x?" We show that the deterministic, randomized and quantum query complexities of the problem are polynomially related. This generalizes earlier results of Aldous [4] and Aaronson [1] and solves the main open problem in [1].

1 Introduction

It follows from the theory of NP-hardness that many optimization problems, such as MAX-2SAT, are difficult. In the MAX-2SAT problem one looks for a truth assignment which maximizes the number of satisfied clauses. In the black box version of the problem we assume that $f : \{0,1\}^k \to \mathbb{Z}$ is an arbitrary function, unknown to us, and we want to find a global optimum for f. It is easy to see that to optimize f classically requires to make $\Omega(n)$ questions to the black box where $n = 2^k$. Using Grover's surprisingly efficient database search algorithm [15], Dürr and Høyer have shown [13] that the quantum version of the same oracle problem has query complexity only $O(\sqrt{n})$. This bound is tight [5, 8, 9, 29], and although much better than the deterministic or randomized complexities, it is still exponential.

A way of relaxing the minimum/maximum finding problem is to look for a solution which is optimal only in some neighborhood structure. For example, in the (weighted) 2SAT-FLIP problem we wish to find a truth assignment $x \in \{0,1\}^k$ such that the sum of the weights of the clauses it satisfies is not less than the same for any other truth assignment of Hamming distance one from x. The number of assignments is exponential (in k), but given any assignment, one can find out in polynomial time if it is locally optimal or not, and in the negative case produce a neighboring solution with a better value. Studying these new problems has lead Johnson, Papadimitriou and Yannakakis[18] to introduce the class PLS (Polynomial Local Search). Several important problems, such as 2SAT-FLIP are known to be complete in this class. Even though PLS is not harder than $\mathrm{NP} \cap \mathrm{coNP}$, it is still conjectured to be computationally intractable, at least classically.

What about the black box version of the local optimization problem, when f is a black box function and we pay only for the queries? Llewellyn, Tovey and Trick [21] and Llewellyn and Tovey [20] treat the deterministic case quite exhaustively. Aldous [4] has shown that for any graph of size n and of maximum degree Δ, the randomized query complexity is $O((n\Delta)^{1/2})$. Aaronson [1] has established an analogous $O(n^{1/3}\Delta^{1/6})$ quantum upper bound.

Two classes of specific neigborhood structures have been studied extensively: the k-dimensional hypercube where $n = 2^k$, and the d-dimensional grid graph, where $n = k^d$ for some fixed $d \geq 2$. For the case of

*LRI, Université Paris-Sud, CNRS, F-91405 Orsay, France; email: santha@lri.fr. Supported by the European Commission IST projects RESQ 37559 and QAP 015848, and by the ANR Blanc AlgoQP grant of the French Research Ministry.

†Rutgers University, email: szegedy@cs.rutgers.edu. Supported by NSF grant 0105692 and the European Commission IST project RESQ 37559. The research was done while the author was visiting LRI, Université Paris–Sud, CNRS.

the hypercube, Llewellyn, Tovey and Trick have shown [21] that to find a local optimum takes deterministically $\Theta(2^k/k^{1/2})$, that is only slightly less than n queries. The randomized and the quantum cases are much harder to analyze. In the randomized case, Aldous [4] proved that any algorithm must use $\Omega(2^{k/2-o(k)})$ queries. Aaronson [1] has improved the lower bound to $\Omega(2^{k/2}/k^2)$, and finally Zhang [30] has closed the gap by showing a matching $\Omega(2^{k/2}k^{1/2})$ bound to Aldous' result. In the quantum case the exact complexity is $\Theta(2^{k/3}k^{1/6})$, where the lower bound is due again to Zhang [30].

The query complexity of the grid graphs is known exactly except in small dimensions. Indeed, the lower bound results of Zhang [30] match the respective upper bounds of Aldous and Aaronson, implying that the randomized complexity is $\Theta(k^{d/2})$ when $d \geq 4$, and the quantum complexity is $\Theta(k^{d/3})$ when $d \geq 6$. The randomized lower bound of $\Omega(k^{3/2}/(\log k)^{1/2})$ of Zhang for $d = 3$ is tight up to a logarithmic factor, as well as his quantum lower bound of $\Omega(k^{5/3}/(\log k)^{1/3})$ for $d = 5$. For $d = 4$, the best quantum lower bound of $\Omega(k^{6/5})$, also due to Zhang, leaves a polynomial gap to the $O(n^{4/3})$ upper bound. The best quantum upper bound of $O(k^{1/2} \log\log k)$, for $d = 2$, was obtained by Verhoeven [28]. Recently, Sun and Yao [27] gave almost optimal lower bounds for the very small dimensional cases. More precisely, in the case $d = 2$, they have shown a randomized lower bound $\Omega(k^{1-\delta})$, and a quantum lower bound $\Omega(k^{1/2-\delta})$, for any constant $\delta > 0$. Their result for the quantum complexity, when $d = 3$, is $\Omega(k^{1-\delta})$, again for any constant $\delta > 0$.

A central question, explicitly raised by Aaronson in [1], has remained unanswered until now: are the deterministic, randomized and quantum query complexities polynomially related for arbitrary neighborhood structures? In this article we give an affirmative answer to this question.

The class PLS is a subset of TFNP, the family of total function problems from NP, introduced by Meggido and Papadimitriou [22]. Informally, TFNP consists of those NP search problems for which a solution is guaranteed to exist. Since factorization is a prominent member of TFNP, one can consider this class as a potential source of problems which might admit efficient quantum algorithms. Other important subclasses of TFNP are PPP (Polynomial Pigeonhole Principle), and PPAD, the directed version of PPA (Polynomial Parity Argument). The elements of PPP are problems which by their combinatorial nature obey the pigeonhole principle and therefore have a solution. The natural complete problem in PPP is PIGEONHOLE CIRCUIT, where, given the description of a boolean circuit with n input and n output bits, one looks for either a pre-image of 0^n or a collision, that is two distinct input strings yielding the same output. The elements of PPAD are search problems, where the existence of a solution is guaranteed by the fact that in a directed graph where all vertices have in-degree and out-degree at most 1, if there is a source then there is another vertex whose in-degree and out-degree adds up to exactly 1. A complete problem in PPAD is for example the 2-dimensional Sperner's problem 2D-SPERNER [11].

In the black box setting, polynomial relationships between the deterministic and quantum complexities of some complete problems have been obtained, in each of the classes PPP, PPAD and PLS. In particular, the collision lower bound of Aaronson and Shi [2] implies that for the black box analog of PIGEONHOLE CIRCUIT, and the quantum lower bound of Friedl et al. [14] for the query complexity of 2D-SPERNER. The underlying graph of 2SAT-FLIP is the hypercube, thus the quantum query lower bounds of Aaronson [1] and Zhang [30] imply a polynomial relationship for the query complexities of this problem. Our result can be interpreted as proving an analogous result for every problem in PLS. As a consequence, if an efficient quantum algorithm is ever to be found for a problem in PLS, it must exploit its specific structure.

2 Results

For a positive integer n we denote $\{1, \ldots, n\}$ by $[n]$. An *oracle problem* is a relation $\mathcal{R} \subseteq S \times T$ where T is a finite set and $S \subseteq \Sigma^{[n]}$ for some finite set Σ. The input is a function $f \in S$, hidden by an oracle, such that $f(x)$, where $x \in [n]$, can be accessed via a query parameterized by x. The output is some $t \in T$ such that $(f, t) \in \mathcal{R}$. A special case is the *functional oracle problem* when the relation is given by a function $A : S \to T$, the (unique) output is then $A(f)$. We say that A is *total* if $S = \Sigma^{[n]}$.

In the query model of computation each query adds one to the complexity of an algorithm, but all other computations are free. The state of the computation is represented by three registers, the query register

$x \in [n]$, the answer register $a \in \Sigma$, and the work register z. The computation takes place in the vector space spanned by all basis states $|x\rangle|a\rangle|z\rangle$. In the *quantum query model* introduced by Beals, Buhrman, Cleve, Mosca and de Wolf [8] the state of the computation is a complex combination of all basis states which has unit length in the norm l_2. In the randomized model it is a non-negative real combination of unit length in the norm l_1, and in the deterministic model it is always one of the basis states.

The query operation O_f maps the basis state $|x\rangle|a\rangle|z\rangle$ into the state $|x\rangle|(a + f(x)) \bmod |\Sigma|\rangle|z\rangle$, where we identify Σ with residue classes $\bmod |\Sigma|$. Non-query operations do not depend on f. A *k-query algorithm* is a sequence of $(k+1)$ operations $(U_0, U_1, \ldots, U_k)$ where U_i is unitary in the quantum and stochastic in the randomized model, and it is a permutation in the deterministic case. Initially the state of the computation is set to some fixed value $|0\rangle|0\rangle|0\rangle$, and then the sequence of operations $U_0, O_f, U_1, O_f, \ldots, U_{k-1}, O_f, U_k$ is applied. A quantum or randomized algorithm ϵ-computes $\mathcal{R}$ for some fixed constant $\epsilon < 1/2$ if the observation of the appropriate last bits of the work register yields some $t \in T$ such that $(f, t) \in \mathcal{R}$ with probability at least $1 - \epsilon$. Then $Q_\epsilon(\mathcal{R})$ (resp. $R_\epsilon(\mathcal{R})$) is the smallest k for which there exists a k-query quantum (resp. randomized) algorithm which ϵ-computes $\mathcal{R}$. In the case of deterministic algorithms exact computation is required, and the deterministic query complexity $D(\mathcal{R})$ is defined then analogously. We have $D(\mathcal{R}) \geq R_\epsilon(\mathcal{R}) \geq Q_\epsilon(\mathcal{R})$.

Beals et al. [8] have shown that in the case of total functional oracle problems the deterministic and quantum complexities are polynomially related, and the gap is at most a degree 6 polynomial. No such relation is known for relations or for partial functional problems, in fact for several partial functional problems exponential quantum speedups are known [12, 24].

Throughout the paper $G(V, E)$ will be an undirected graph with vertex set V and edge set E, and we assume that $V = [n]$. In this paper we are concerned with the following oracle problem:

Local Search for G (in notation, $\mathrm{LS}_G**):**
Oracle Input: A function $f : V \to \mathbb{Z}$.
Output: An $x \in V$ such that for every $y \in V$ with $\{x, y\} \in E$ we have $f(x) \leq f(y)$.

We have chosen $\mathbb{Z}$ as the range of the input functions only for presentational convenience, it could be restricted to $[n]$ without loss of generality. Also note that graph G is known to us, and we treat it as a parameter. The problem as specified is a relation. However, an important special case is functional local search, when only inputs with a guaranteed unique local (and therefore global) minimum are considered. We call such inputs *unimodal*, they will play an important role in our considerations.

Our main result is analogous to the statement of Beals et al. [8] for local search: the quantum and deterministic query complexities are polynomially related.

Theorem 1 (Main Theorem). *For every graph G, we have $D(\mathrm{LS}_G) \in O(Q_\epsilon(\mathrm{LS}_G)^{19})$.*

We will prove our main theorem by showing that both the classical and the quantum query complexities of a graph G are polynomially related to three parameters: its maximum degree, the logarithm of its size and its separation number. In section 3 we present a deterministic algorithm for LS_G whose complexity is polynomial in these parameters (**Theorem 2**). The degree lower bound for the quantum complexity which follows trivially by a reduction from Grover's search (**Fact 1**), and the log-size lower bound (**Theorem 3**) are presented in section 4. The crucial lower bound involving the separation number (**Theorem 4**) is the subject of section 5, where we also prove our main theorem.

3 Local Search and Graph Parameters

Let $G(V, E)$ be a graph with vertex set V and edge set E. If $G'(V', E')$ is a subgraph of G then we write $G' \leq G$. We denote the subgraph of G spanned by a vertex set $V' \subseteq V$ with $G(V')$. The degree of a vertex in G is $\deg_G x$. We denote the maximum degree of G with $\Delta(G)$ or simply Δ if G is clear from the context. The vertex boundary of a set $H \subseteq V$ in G is

$$\partial_G H = \{x \in V \setminus H \mid (\exists y \in H)\ \{x, y\} \in E\}.$$

If $H = \{x\}$ then we denote $\partial_G H$ by $\partial_G x$. We define now our most important graph parameter.

Definition 1 (Separation number). *The* separation number *of a graph $G(V, E)$ is*

$$s(G) = \max_{H \subseteq V} \min_{\substack{S \subseteq H, \\ |H|/4 \leq |S| \leq 3|H|/4}} |\partial_{G(H)} S| \tag{1}$$

The separation number is one of the many graph parameters that measure expansion and connectivity of a graph, and it is the same as the treewidth within a constant factor.

We now give a deterministic algorithm which is a refinement of the divide and conquer procedure of Llewellyn, Tovey and Trick [21] which iteratively splits the graph into pieces by querying a separator set. The refinement consists of the specific choice of the separator set which makes explicit the connection between the complexity of the problem and the parameters we are interested in. For the sake of completeness we prove here the correctness of the algorithm and analyze its complexity in the three parameters. For every $H \subseteq V$ let H^* be the subset of H which minimizes $|\partial_{G(H)} S|$ when S is a subset of H of size between $|H|/4$ and $3|H|/4$.

Algorithm 1 Deterministic algorithm for LS_G

$H := V$, $S := \emptyset$, $u :=$ any vertex of G.
while output not found **do**
  Query $\{u\} \cup \partial_{G(H)} H^*$ to find the vertex m which minimizes f in this set, query $\partial_{G(H)} m$
  if $f(m) \leq f(v)$ for all $v \in \partial_{G(H)} m$ **then**
    output $:= m$
  else
    $S := S \cup \partial_{G(H)} H^*$, let w be the vertex which minimizes f in $\partial_{G(H)} m$
    $u := w$
    Let H be the connected component of $H \setminus \partial_{G(H)} H^*$ which contains u
  end if
end while

Theorem 2. Algorithm 1 *correctly solves* LS_G *and has query complexity* $O(\log n(\Delta(G) + s(G)))$.

Proof. By induction one sees that at the beginning of each iteration,

$$\partial_G v \subseteq S \cup \partial_{G(H)} v \text{ for all } v \in H, \tag{2}$$

$$f(u) \leq f(s) \text{ for all } s \in S. \tag{3}$$

The algorithm always terminates since in each iteration $|H|$ decreases, and if $H = \{u\}$ then it outputs u. When the output m is produced then by definition $f(m) \leq f(v)$ for all $v \in \partial_H m$. Also $f(m) \leq f(u)$, and $f(m) \leq f(v)$ for all $v \in S$ by (3). Therefore m is a local minimum of f in G by (2). The number of iterations is $O(\log n)$ since $|H|$ always decreases by a factor of at least 4/3, and inside each loop the number of queries is at most $\Delta(G) + s(G) + 1$. □

4 Log-size lower bound

The maximum degree and the $\log n$ lower bounds are obtained by reduction: the first one, very simple, from Grover's search, the second one, more sophisticated, from ordered search.

Fact 1 (Aaronson [1]). *Let $G(V, E)$ be connected with maximum degree Δ. Then $Q_\epsilon(\mathrm{LS}_G) \in \Omega(\sqrt{\Delta})$.*

In the ordered search reduction we will use unimodal functions, like Aaronson [1], which arise from paths rooted at a fixed node of G. We define now the generic construction of functions generated by paths. A *directed path* P in G is a sequence of vertices $(p_0, p_1, \ldots, p_l)$ (with possible repetitions) such that for every $0 \leq i \leq l-1$ node p_i is connected to node p_{i+1} in G. In this article we do not consider undirected paths. Let G be a connected graph, and let us denote by $\mathrm{dist}_G(x, y)$ the length of the shortest path in G between vertices x and y.

Definition 2 (Path function). *The* path function $f_P : V \to \mathbb{Z}$ *is defined by*

$$f_P(x) = \begin{cases} -j & \text{if } p_j = x \text{ and } p_k \neq x \text{ for } k > j \\ \mathrm{dist}_G(p_0, x), & \text{if } \forall j,\ p_j \neq x. \end{cases}$$

Observe that path functions are always unimodal, and that they have their minimum in the last vertex of the path.

Reductions between black box problems are regularly used to prove quantum lower bound. In these reductions usually a single query to the oracle of the reduced problem simulates a query of the problem to which the reduction is made. However, in more sophisticated cases, several queries can be permitted for the simulation. We make this precise in the following definition and lemma.

Definition 3. *A functional oracle problem $A : S_1 \to T$ with $S_1 \subseteq \Sigma_1^{[n]}$ is* k-query reducible *to a functional oracle problem $B : S_2 \to T$ with $S_2 \subseteq \Sigma_2^{[m]}$ if the following two conditions are satisfied:*

$$\begin{gathered} \exists \alpha : S_1 \to S_2 \text{ such that } \forall f \in S_1,\ A(f) = B(\alpha(f)); \\ \exists \gamma_1, \ldots, \gamma_k : [m] \to [n] \text{ and } \gamma : [m] \times \Sigma_1^k \to \Sigma_2 \text{ such that} \\ \forall f \in S_1,\ x \in [m], (\alpha(f))(x) = \gamma(x, f(\gamma_1(x)), \ldots, f(\gamma_k(x))) \end{gathered}$$

Lemma 1. *If A is k-query reducible to B then $Q_\epsilon(B) \geq Q_\epsilon(A)/2k$.*

Proof. Let $\mathcal{B}$ be a query algorithm which solves B with c queries. We show that there exists a query algorithm $\mathcal{A}$ which solves A with $2kc$ queries. The algorithm $\mathcal{A}$ simulates $\mathcal{B}$, and in its non-query steps it is identical to it. For simulating the query steps $\mathcal{A}$ will use $2k$ additional registers. When $\mathcal{B}$ makes a query step involving the basis state $|x\rangle$ then $\mathcal{A}$ computes $\gamma_1(x), \ldots, \gamma_k(x)$ in the first k additional registers, queries the oracle k-times to get $f(\gamma_1(x)), \ldots, f(\gamma_k(x))$ in the next k registers, computes $\alpha(f)(x)$ in the oracle answer register, and finally erases the contents of the additional registers. □

In fact, Lemma 1 remains also valid with weaker conditions on the reduction. For example, one could authorize that the queries to $\alpha(f)$ are computed by a quantum query algorithm which can make k queries to f. The reduction of Buhrman and de Wolf [10] of the parity problem to ordered search indeed uses this more general notion. However, for our purposes the more restricted reduction is sufficient.

Theorem 3. *Let $G(V, E)$ be a connected graph on n vertices. Then $Q_\epsilon(\mathrm{LS}_G) \in \Omega(\log n)$.*

Proof. Let $S_1 = \{f_y : y \in [n]\}$, where $f_y : [n] \to \{0, 1\}$ is defined by

$$f_y(x) = \begin{cases} 0 & \text{if } x \leq y \\ 1 & \text{otherwise.} \end{cases}$$

The ordered search problem is defined by $A(f_y) = y$, and Høyer, Neerbek and Shi [16] have shown that $Q(A) = \Omega(\log n)$. We will show that A is 2-query reducible to LS_G, and then the result follows from Lemma 1. Let $V = [n]$, and consider any depth first search traversal of G. Let dfn be the depth-first numbering of the vertices (that is the vertices are numbered in preorder). To simplify the description, without loss of generality we will suppose that $\mathrm{dfn}(x) = x$ for every vertex x. This means for example that vertex 1 is the root of the depth first search traversal. Let $\mathrm{desc}(x)$ be the number of descendants of x in the

search. By definition we assume that x is an ancestor and descendant of itself. It is well known [3] that for every vertex y, vertex x is an ancestor of y if and only if

$$x \leq y < x + \mathrm{desc}(x). \tag{4}$$

For $y \in [n]$, let P_y be the path from vertex 1 to vertex y in the above dfs tree of G. By the definition of the path function,

$$f_{P_y}(x) = \begin{cases} -\mathrm{dist}_G(1,x) & \text{if } x \text{ is on the path from 1 to } y \\ \mathrm{dist}_G(1,x) & \text{otherwise.} \end{cases}$$

The unique local minimum of f_{P_y} is in the vertex y, therefore $\mathrm{LS}_G(f_{P_y}) = y$, and for the purpose of the reduction we can set $S_2 = \{f_{P_y} : y \in [n]\}$ and $\alpha(f_y) = f_{P_y}$. To determine the answer of the oracle $\alpha(f_y)$ on query x, using the oracle f_y, it has to be decided if x is an ancestor of y. By (4) this happens exactly when $f_y(x) = 0$ and $f_y(x + \mathrm{desc}(x)) = 1$. Thus if we set $\gamma_1(x) = x$, $\gamma_2(x) = x + \mathrm{desc}(x)$ and

$$\gamma(x, b_1, b_2) = \begin{cases} -\mathrm{dist}_G(1,x) & \text{if } b_1 = 0 \text{ and } b_2 = 1 \\ \mathrm{dist}_G(1,x) & \text{otherwise,} \end{cases}$$

then $f_{P_y}(x) = \gamma(x, f_y(\gamma_1(x)), f_y(\gamma_2(x)))$, and the reduction is completed.

□

5 Quantum Query Complexity and Graph Expansion

The purpose of this section is to prove the following lower bound.

Theorem 4. *Let $G = (V, E)$ be a connected graph on n vertices with maximum degree Δ and separation number s. Then $Q_\epsilon(\mathrm{LS}_G) \in \Omega((s/\Delta)^{1/8}(\log n)^{-1})$.*

Proof. The statement is a direct consequence of Theorems 6 and 7 below. □

Beside the reduction method, lower bound proofs in the quantum query model fall into two main categories: they are obtained either by the degree argument (the polynomial method) of Beals at al. [8] or by the quantum adversary method of Ambainis [5]. Aaronson [1] uses the latter, and our paper closely follows his proof scheme. Like in [1], all our sample functions will be path functions generated by paths rooted at a fixed node of G (Aaronson calls these paths snakes). Since we need to present an argument that works for general graphs we invented some useful lemmas about the structure of expander graphs. Another place where we improve on [1] is that we almost entirely eliminate the problem with self intersecting paths. To do so we use a more general version of the quantum adversary method explained in the next section.

5.1 Quantum Adversary Method

The quantum adversary method of Ambainis [5] has several known extensions. These include the spectral method of Barnum, Saks and Szegedy [7], the weighted adversary method of Ambainis [6], the strong weighted adversary method of Zhang [31], and the Kolmogorov complexity method of Laplante and Magniez [19]. All these methods were proven to be equivalent by Špalek and Szegedy [26]. Recently, an even more powerful method using negative weights has been proposed by Høyer, Lee and Špalek [17].

The spectral method, that expresses the bound in terms of ratios of eigenvalues of matrices, was derived from the characterization of quantum query complexity as a semidefinite feasibility problem [7]. Though the spectral method has been stated only for Boolean function, its generalization to the non-Boolean case is quite straightforward, as pointed out in [25] and [26]. We will need the more general form of the method, that we state here. For an entry-wise non-negative symmetric matrix M, let us denote its largest eigenvalue by $\lambda(M)$.

Theorem 5 (Spectral Method). *Let $A : S \to T$ be a functional oracle problem where $S \subseteq \Sigma^{[n]}$. Let Γ be an arbitrary $S \times S$ nonnegative symmetric matrix that satisfies $\Gamma[f,g] = 0$ whenever $A(f) = A(g)$. For $x \in [n]$ let Γ_x be the matrix:*

$$\Gamma_x[f,g] = \begin{cases} 0 & \text{if } f(x) = g(x) \\ \Gamma[f,g] & \text{if } f(x) \neq g(x). \end{cases}$$

Then:

$$Q_\epsilon(A) \in \Omega\left(\frac{\lambda(\Gamma)}{\max_{1\leq x\leq n}\lambda(\Gamma_x)}\right).$$

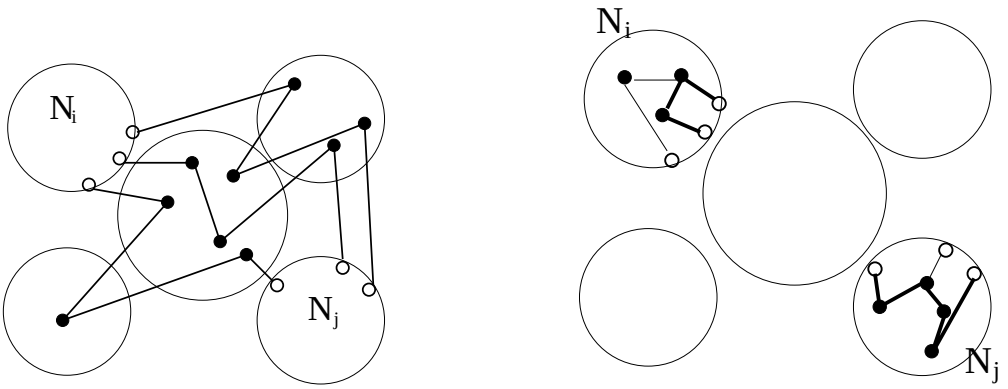
5.2 Path Arrangement

Let $G(V,E)$ be a connected graph. For a directed path $P = (p_0, p_1, \ldots, p_l)$ we set

$$\mathrm{start}(P) \stackrel{def}{=} p_0; \qquad \mathrm{end}(P) \stackrel{def}{=} p_l; \qquad \mathrm{length}(P) \stackrel{def}{=} l.$$

We call p_0 the starting point of P and p_l the end point. All paths other than the empty path have a starting and an end point (which may coincide). For a path P we denote by $P^\ominus$ the path that we obtain by deleting both the starting and end points from P. Formally, $P^\ominus = (p_1, \ldots, p_{l-1})$. Note that if P is just an edge (p_0, p_1) then $P^\ominus = \epsilon$, the empty path. For paths $P = (p_0, \ldots, p_l)$ and $Q = (q_0, \ldots, q_{l'})$ with $q_0 = p_l$ their *join* is $PQ = (p_0, \ldots, p_l, q_1 \ldots, q_{l'})$. A path is *simple* if no vertex occurs more than once in it.

We often view a path P as a multi-set. For a path P and $x \in V$ let $\mathrm{mult}(x,P)$=$|\{j \mid p_j = x\}|$, that is the number of times P goes through x, and let $\mathrm{mult}(P) = \max_{x\in V}\mathrm{mult}(x,P)$.



(a) **(b)**

Figure 1: Figure (a) shows a set of three inter-cluster paths between N_i and N_j. Figure (b) shows trees T_i and T_j. With thicker lines it also shows the edges of an intra-cluster path within T_i and another one within T_j.

Definition 4 (Path arrangement and Inter/Intra-cluster path). *A* path arrangement *for G with parameter m is a set of connected, disjoint subsets $N_1, \ldots, N_m \subseteq V$ with some fixed spanning trees T_i of $G(N_i)$ together with a set of directed paths $\{E_i(j,k)\}_{1\leq i,j\neq k\leq m}$ ($1 \leq i, j \neq k \leq m$ means $1 \leq i, j, k \leq m$ with the additional condition that $j \neq k$) such that*

1. $E_i(j,k) \cap N_j = \{\mathrm{start}(E_i(j,k))\}$ *for* $1 \leq i, j \neq k \leq m$;
2. $E_i(j,k) \cap N_k = \{\mathrm{end}(E_i(j,k))\}$ *for* $1 \leq i, j \neq k \leq m$;
3. $E_i(j,k)^\ominus \cap E_{i'}(j,k)^\ominus = \emptyset$ *for* $1 \leq i \neq i', j \neq k \leq m$;

4. *$E_i(j,k)$ is simple for $1 \leq i, j \neq k \leq m$.*

In the path arrangement the sets N_i are called the clusters *for $1 \leq i \leq m$. The paths $E_i(j,k)$ are called* inter-cluster paths *and i is the* enumerator number *of $E_i(j,k)$. For $x, y \in N_i$, the* intra-cluster path *$R_i(x,y)$ is defined as the shortest path from x to y in T_i.*

Definition 5 (Path arrangement parameter). *The* path arrangement parameter *$m(G)$ of G is the maximal m such that there is path arrangement for G with parameter m.*

To bring an example to the above definitions consider the two dimensional $\sqrt{n}$ by $\sqrt{n}$ grid, $G_{2,n}$ with vertex set $V = I \times J$ (we suppose that n is a square number, I and J are paths of length $\sqrt{n}$). Two vertices in V are connected if either their I coordinates are the same and their J coordinates are connected or their J coordinates are the same and their I coordinates are connected. We create a path arrangement. The clusters will be the sets $\{i\} \times J$ for all $i \in I$. Clearly, every cluster induces a connected graph in $G_{2,n}$. For two clusters, $\{i\} \times J$ and $\{i'\} \times J$, define $P_{i,i'}$ as the unique shortest path connecting i and i' in I, and consider the collection of paths

$$E_j(i,i') = P_{i,i'} \times \{j\},$$

where j runs through all elements of J. These paths will serve as inter-cluster paths in between the two clusters. It is easy to check that all conditions of Definition 4 are satisfied and therefore $m(G_{2,n}) \geq \sqrt{n}$. From Fact 2 below it follows that $m(G_{2,n}) \leq 2\sqrt{n}+1$, and therefore we conclude that $m(G_{2,n}) \in \Theta(\sqrt{n})$.

Another example is the complete graph, K_n on the vertex set V with $|V| = n$. Here the individual nodes can be defined as the clusters and the inter-cluster paths in between any two nodes $v, w \in V$ are $E_i(v,w) = (v,w)$, the same for all $1 \leq i \leq n$. The disjointness property for these paths holds, since $(v,w)^{\ominus} = \emptyset$. We conclude that $m(K_n) = n$.

Fact 2. *$m(G) \leq \sqrt{n\Delta} + 1$ for all graphs G.*

Proof. Let a be the size of the smallest cluster: $a = min_{1 \leq j \leq m} |N_j|$ with $a = |N_{j_0}|$. We show that $a\Delta \geq m - 1$. Indeed, notice that $a\Delta$ is an upper bound on the size of the edge-boundary of N_{j_0}. Now, either there is another cluster, N_{j_1}, to which there is no direct edge from N_{j_0}, and so the m disjoint inter cluster paths from N_{j_0} to N_{j_1} require m outgoing edges from N_{j0}, or N_{j0} has a direct edge to each other cluster, which requires at least $m-1$ outgoing edges. Also, trivially $a(m-1) \leq am \leq n$ (this is the only place where we use that N_{j_0} is the smallest). In summary, $m - 1 \leq a\Delta$ and $m - 1 \leq n/a$. We get now $m - 1 \leq \sqrt{n\Delta}$ by multiplying the above two inequalities, and taking the square root of both sides. □

5.3 Separation Number versus Path Arrangement Parameter

We will show that the path arrangement parameter, $m(G)$ is in $\Omega(\sqrt{s/\Delta})$. We start with some easy lemmas.

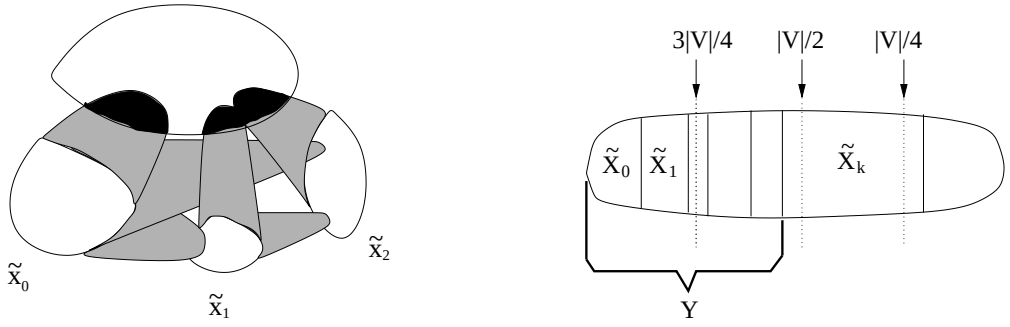
Definition 6. *A graph $G(V,E)$ is a (n_1, n_2, λ)-*expander *if for every subset $H \subseteq V(G)$ such that $n_1 < |H| \leq n_2$, we have that $|\partial_G H| \geq \lambda |H|$.*

Lemma 2. *Every connected graph G on n vertices which is an $(n/4, 3n/4, \lambda)$-expander has an induced subgraph G' with at least $n/2$ vertices which is a $(0, n/2, \lambda)$-expander.*

Proof. For $X \subseteq V$ we denote the subgraph of G induced on X by $G(X)$. Assume that the statement of the lemma is false. Then with every $X \subseteq V$ such that $|X| \geq n/2$ we can associate a non-empty $\tilde{X} \subseteq X$ with the property that $|\tilde{X}| \leq n/2$ and $|\partial_{G(X)} \tilde{X}| < \lambda |\tilde{X}|$.

Define $X_0 = V$ and $X_i = X_{i-1} \setminus \tilde{X}_{i-1}$ for $i \geq 1$. Since $|X_i|$ is strictly smaller than $|X_{i-1}|$ there has to be an index k such that $|X_k| \geq n/2$ but $|X_{k+1}| < n/2$. We define

$$Y = \begin{cases} V \setminus X_{k+1} & \text{if } |V \setminus X_{k+1}| \leq 3n/4, \\ V \setminus X_k & \text{otherwise.} \end{cases}$$



(a) **(b)**

Figure 2: Figure (a) shows that if G is not an expander with respact to sets smaller than $|V|/4$, we can iteratively split off the bad-behaving sets. In the figure the black region represents the vertex-boundary of the union. The figure helps to understand Equation (5), which in words says that the union of bad-behaving sets is also bad-behaving. Figure (b) helps to understand the choice for Y. We could have selected any index j such that the size of $Y = \tilde{X}_0 \dot{\cup} \ldots \dot{\cup} \tilde{X}_j$ is between $|V|/4$ and $3|V|/4$.

Since $|\tilde{X}_k| \leq n/2$, in both cases it holds that $n/4 < |Y| \leq 3n/4$. Therefore $|\partial_G Y| \geq \lambda|Y|$. By another argument we show that the boundary of Y is small, thus getting a contradiction. Suppose that $Y = V \setminus X_k$, the other case is similar. Notice that

$$Y = \tilde{X}_0 \dot{\cup} \tilde{X}_1 \dot{\cup} \cdots \dot{\cup} \tilde{X}_{k-1},$$

where $\dot{\cup}$ denotes the disjoint union. Therefore

$$\partial_G Y \;=\; (\partial_G Y) \cap X_k \;\subseteq\; \bigcup_{i=0}^{k-1} ((\partial_{G(X_i)} \tilde{X}_i) \cap X_k) \;\subseteq\; \bigcup_{i=0}^{k-1} \partial_{G(X_i)} \tilde{X}_i. \tag{5}$$

By our assumption the last set has size less than $\lambda|\tilde{X}_0| + \cdots + \lambda|\tilde{X}_{k-1}| = \lambda|Y|$, which is a contradiction. □

Lemma 3. *For every connected graph $G(V, E)$ and for every $1 \leq k \leq |V|$ we can find at least $\lfloor |V|/k \rfloor$ vertex disjoint connected subgraphs of G, each with at least k/Δ vertices.*

Proof. Set $N_0 = \emptyset$. Let l be the smallest integer such that $(l+1)k > |V|$, observe that $l = \lfloor |V|/k \rfloor$. It is enough to find $N \subseteq V$ such that $G(N)$ and $G(V \setminus N)$ are connected and $k/\Delta \leq |N| \leq k$. Indeed, then by induction we can pick N_i for $i = 1, \ldots, l$ such that N_i is connected, has size between k/Δ and k, and is a subset of $V \setminus \cup_{j=1}^{i-1} N_j$.

We now prove the existence of the above N. If $k = n$ then we just set $N = V$. Otherwise let T_0 be a spanning tree of G with root r_0. Let r_1 be the child of r_0 which roots the subtree T_1 of T_0 with the greatest number of nodes among all children of r_0. Similarly, let r_2 be the child of r_1 which roots the subtree T_2 of T_1 with the greatest number of nodes among all children of r_1, etc, until we get T_s, a tree with $|T_s| \leq k$. The size of T_{s-1} is strictly greater than k, therefore the size of T_s is at least $(|V(T_{s-1})| - 1)/\Delta \geq k/\Delta$. Thus $V(T_s)$ is a good choice for N. □

Lemma 4. *Let $G(V, E)$ be a $(0, |V|/2, \lambda)$-expander, and let $N_1, N_2 \subseteq V$ be disjoint subsets, both of size at least k. Then there are at least $m = \lambda k$ paths $E_1, \ldots, E_m$ such that*

1. *$E_i \cap N_1 = \{\mathrm{start}(E_i)\}$ for $1 \leq i \leq m$;*
2. *$E_i \cap N_2 = \{\mathrm{end}(E_i)\}$ for $1 \leq i \leq m$;*

3. $E_i^{\ominus} \cap E_{i'}^{\ominus} = \emptyset$ *for* $1 \leq i \neq i' \leq m$;

4. E_i *is simple for* $1 \leq i \leq m$.

Proof. If there is an edge (v, w) with $v \in N_1$, $w \in N_2$ then we can set $E_i = (v, w)$ for $1 \leq i \leq m$. Otherwise contract N_1 into a single node n_1 and N_2 into a single node n_2. Assume that there are no paths in G that satisfy 1-4. Then there are no paths satisfying 1-3 either. It follows that in the contracted graph, G', one cannot find m vertex-disjoint paths that connect n_1 and n_2. This, in turn, by Menger's theorem [23] implies that there is a vertex cut $C \subseteq V$ of size smaller than m which separates n_1 and n_2. Let C_1' be the component of G' that contains n_1 and C_2' be the component of G' that contains n_2, and let the expanded version of these components be C_1 and C_2. Without loss of generality we can assume that $k \leq |C_1| \leq n/2$, which leads to a contradiction, since $\partial_G C_1 \subseteq C$, $|C| < m = \lambda k$. □

Theorem 6. *Let G be a connected graph with maximum degree Δ, separator number s and path arrangement parameter m. Then we have $m \geq \max\{\lfloor\sqrt{s/2\Delta}\rfloor, 1\}$.*

Proof. If $s < 2\Delta$ then the statement of the theorem is trivially true. Otherwise, let H be the subset of V for which the right hand side of (1) is maximized. Then $s = s(G(H))$, which implies that $G(H)$ is an $(|H|/4, 3|H|/4, s/|H|)$-expander. By Lemma 2 there is an $H' \subseteq H$ such that $|H'| \geq |H|/2$ and $G(H')$ is an $(0, |H|/2, s/|H|)$-expander. We shall construct the required path arrangement inside $G(H')$. The graph $G(H')$ is connected since all of its components have to be $(0, |H|/2, s/|H|)$-expanders, and this is possible only if $G(H')$ has a single component. In order to use Lemma 3, we set $k = |H'|\sqrt{2\Delta/s}$. Observe that $k \leq |H'|$ since $s \geq 2\Delta$. Also, $k \geq 1$ since the size of the vertex boundary of a any subset of H' of size $|H'|/2$ is at least s. By Lemma 3 we find clusters $N_1, \ldots, N_{\lfloor\sqrt{s/2\Delta}\rfloor}$ such that $|N_i| \geq |H'|/\sqrt{s\Delta/2}$ and $G(N_i)$ is connected for $1 \leq i \leq \lfloor\sqrt{s/2\Delta}\rfloor$. Here we used that $\Delta(G(H')) \leq \Delta$. To finish the proof now all we need is to construct the set of intra-cluster paths. Observe that $G(H')$ is also an $(0, |H'|/2, s/|H|)$-expander since $|H'| \leq |H|$. Therefore by Lemma 4 between N_i and N_j we can find $(|H'|/\sqrt{s\Delta/2}) \times (s/|H|)$ external paths. Since $|H'| \geq |H|/2$, the previous expression is at least $\sqrt{s/2\Delta}$. We have thus constructed a path arrangement with parameter $\lfloor\sqrt{s/2\Delta}\rfloor$, and therefore, according to Definition 5 the path arrangement parameter of G is at least $\lfloor\sqrt{s/2\Delta}\rfloor$. □

5.4 The Adversary Matrix

Let $G(V, E)$ be a connected graph, we want to define our adversary matrix for LS_G. We fix a path arrangement with parameter $m = m(G)$ for which we use the notations of Definition 4. We set $a = \sqrt{m}/10$ for the rest of the section. To every $\overline{r} = r_0 \ldots r_{2a} \in [m]^{2a+1}$ with $r_0 = 1$, we define a directed path $P(\overline{r}) = P(\overline{r}, 0)P(\overline{r}, 1) \ldots P(\overline{r}, 2a-1)$ which will be the join of $2a$ paths. Even indices will name clusters, odd indices will name enumerator numbers of intra-cluster paths. We fix an arbitrary starting node $x_{\mathrm{start}} \in N_1$, and for $0 \leq i \leq 2a-1$ we define:

$$P(\overline{r}, i) = \begin{cases} R_1(x_{\mathrm{start}}, \mathrm{start}(P(\overline{r}, 1))) & \text{if } i = 0 \\ E_{r_i}(r_{i-1}, r_{i+1}) & \text{if } i \text{ is odd} \\ R_{r_i}(\mathrm{end}(P(\overline{r}, i-1))), \mathrm{start}(P(\overline{r}, i+1))) & \text{if } i \text{ is even and } 2 \leq i \leq 2a-2. \end{cases}$$

For $\overline{r}$ and $\overline{r}'$ let $\mathrm{div}(\overline{r}, \overline{r}')$ be the first i such that $r_i \neq r_i'$. We set

$$\Gamma'[\overline{r}, \overline{r}'] = \begin{cases} m^{\mathrm{div}(\overline{r}, \overline{r}')} & \text{if } \mathrm{div}(\overline{r}, \overline{r}') \text{ is odd and } r_{2a} \neq r_{2a}' \\ 0 & \text{otherwise.} \end{cases}$$

Observe that the rows (and columns) of Γ' are indexed by sequences and not functions on V as required. This is just a formal matter. With every sequence $\overline{r}$ we associate the path $P(\overline{r})$ and with every such path we associate the path function $f_{P(\overline{r})}$ as in Section 4. As an index, $\overline{r}$ really corresponds to the function $f_{P(\overline{r})}$. We have chosen a weight function with the property that if we pick a pair of paths with probability

that is proportional to the weight of the pair, then the uncertainty about where the members of this pair will separate from each other is the largest. This uncertainty is maximized when the probability that the separation happens at an (odd) index between 1s and $2a-1$ is uniformly $1/a$. The exponential increase of the weight function in $\mathrm{div}(\overline{r},\overline{r}')$ ensures exactly this.

The matrix Γ' is not yet the adversary matrix we need, we have to zero out some rows and columns that correspond to "bad sequences." Let us recall that $\mathrm{mult}(x,P(\overline{r}))$ is the number of times the path $P(\overline{r})$ goes through the vertex x. A sequence $\overline{r}$ is *repetitive* if there are indices $0 \le i \ne j \le 2a$ such that $r_i = r_j$, it is *criss-crossing* if there is an $x \in V$ such that $\mathrm{mult}(x,P(\overline{r})) > 100\log|V|$. Finally $\overline{r}$ is *bad* if it is either repetitive or criss-crossing otherwise it is *good*. We define our adversary matrix as:

$$\Gamma[\overline{r},\overline{r}'] = \begin{cases} \Gamma'[\overline{r},\overline{r}'] & \text{if both } \overline{r} \text{ and } \overline{r}' \text{ are good} \\ 0 & \text{otherwise.} \end{cases}$$

Theorem 7. *For every connected graph G on n vertices, and with path arrangement parameter m, we have $Q_\epsilon(\mathrm{LS}_G) \in \Omega(m^{1/4}/\log n)$.*

Proof. The statement is an immediate consequence of the spectral method (Theorem 5), and Lemmas 5 and 6 below. □

5.4.1 Eigenvalue Lower Bound

Recall that Γ is the adversary matrix, m is the path arrangement parameter of G, and $a = \sqrt{m}/10$.

Lemma 5. $\lambda(\Gamma) \in \Omega(am^{2a+1})$.

Proof. The matrix Γ has m^{2a} rows. In order to lower bound $\lambda(\Gamma)$ we simply use the entry-sum estimate, that is $\lambda(\Gamma) \ge \sum_{\overline{r},\overline{r}'} \Gamma[\overline{r},\overline{r}']/m^{2a}$. First we estimate the entry-sum of Γ'. Since all rows and columns of Γ' have the same sum, we will sum up the elements in an arbitrary row indexed with $\overline{r}$. In that row, we group the columns together according to $\mathrm{div}(\overline{r},\overline{r}')$, and get

$$\sum_{\overline{r}'} \Gamma'[\overline{r},\overline{r}'] = \sum_{l=0}^{a-1} \sum_{\overline{r}':\mathrm{div}(\overline{r},\overline{r}')=2l+1} \Gamma_1[\overline{r},\overline{r}'].$$

For the group of columns indexed with $\overline{r}'$ such that $\mathrm{div}(\overline{r},\overline{r}') = 2j+1$, the corresponding entries of our row sum up to $m^{2j+1}(m-1)m^{2a-2j-2}(m-1)$. Therefore we obtain

$$\sum_{\overline{r}':\mathrm{div}(\overline{r},\overline{r}')=2l+1} \Gamma'[\overline{r},\overline{r}'] = (1+o(1))m^{2a+1}. \tag{6}$$

Since $\mathrm{div}(\overline{r},\overline{r}')$ can have a different values, this implies the estimation

$$\sum_{\overline{r}'} \Gamma'[\overline{r},\overline{r}'] = (1+o(1))am^{2a+1}. \tag{7}$$

We now upper bound the number of bad rows and columns. The number of non-repetitive sequences $\overline{r}$ is $(m-1)\dots(m-2a) \ge (m-2a)^{2a}$. With our choice of $a = \sqrt{m}/10$, this is approximately $e^{-1/25}m^{2a}$ which is greater than $0.9m^{2a}$.

Pick now randomly a non-repetitive $\overline{r}$. We claim that for a fixed x, the probability that $\mathrm{mult}(x,P(\overline{r})) \ge 100\log n + 1$ is upper bounded by

$$\binom{a}{100\log n} \frac{1}{m(m-1)\dots(m-100\log n)}. \tag{8}$$

To see that, fix the even indices anyhow and condition on this fixing. There can be only at most one even index, say $2q$, such that $x \in P(\overline{r},2q)$. There has to be $100\log n$ odd indices $2i+1$, for which

$x \in P(\overline{r}, 2i+1)^{\ominus}$ since inter-cluster paths are simple. However, once the even indices are fixed, we get an independence between paths belonging to different odd indices, except that because of the non-repetitive nature of $\overline{r}$ their enumerator number should be different. (Note that for every $1 \leq j \neq k \leq m$, where $x \notin N_j \cup N_k$ there is at most one i such that $x \in E_i(j,k)$ by property 3 of Definition 4.) From this the above formula follows.

The expression in (8) is easily upper-bounded by $o(1/n)$. Applying the union bound over all $x \in V$, we get that for a random non-repetitive $\overline{r}$ the probability that $\mathrm{mult}(P(\overline{r})) \geq 100 \log n + 1$ is $o(1)$. We got that that the number of good rows (and columns) is at least $0.9m^{2a}(1-o(1))$. Since all rows and columns of Γ have the same sum, we conclude that taking out $0.1m^{2a}(1+o(1))$ rows and $0.1m^{2a}(1+o(1))$ columns decreases the entry sum of Γ by at most a factor of $0.2 + o(1)$. Thus

$$\lambda(\Gamma) \in \Omega(am^{2a+1}).$$

□

5.4.2 Eigenvalue Upper Bound

Recalling from Theorem 5 the definition of Γ_x, we have for $x \in V$,

$$\Gamma_x[\overline{r}, \overline{r}'] = \begin{cases} \Gamma[\overline{r}, \overline{r}'] & \text{if } f_{P(\overline{r})}(x) \neq f_{P(\overline{r}')}(x) \\ 0 & \text{otherwise.} \end{cases}$$

As always, m is the path arrangement parameter of G, and $a = \sqrt{m}/10$.

Lemma 6. *$\lambda(\Gamma_x) \in O(\sqrt{a}m^{2a+1} \log n)$ for all $x \in V$.*

Proof. In order to give an upper bound on $\lambda(\Gamma_x)$, we decompose Γ_x as $\Gamma_x = \sum_{j=0}^{100 \log n} \Gamma_x^{(j)}$ where

$$\Gamma_x^{(j)}[\overline{r}, \overline{r}'] = \begin{cases} \Gamma_x[\overline{r}, \overline{r}'] & \text{if } \min\{\mathrm{mult}(x, P(\overline{r})), \mathrm{mult}(x, P(\overline{r}'))\} = j \\ 0 & \text{otherwise.} \end{cases}$$

Such a decomposition is possible since the entries corresponding to criss-crossing sequences are zero in Γ, and therefore in Γ_x.

We will use two known properties of the λ function. Firstly, it is sub-additive, that is $\lambda(M_1 + M_2) \leq \lambda(M_1) + \lambda(M_2)$. Secondly, according to a generalization of Mathias' lemma due to Špalek and Szegedy (Lemma 2 in [26]), for any matrix M with non-negative coefficients,

$$\lambda(M) \leq \max_{p,q:M[p,q]\neq 0} \sqrt{\Big(\sum_s M[p,s]\Big)\Big(\sum_s M[s,q]\Big)}. \tag{9}$$

Because of sub-additivity, it is sufficient to show that $\lambda(\Gamma_x^{(j)}) \in O(\sqrt{a}m^{2a+1})$ for $0 \leq j \leq 100 \log n$. Fix $j \leq 100 \log n$. Using (9) we get

$$\lambda(\Gamma_x^{(j)}) \leq \max_{\overline{r},\overline{r}':\min\{\mathrm{mult}(x,P(\overline{r})),\mathrm{mult}(x,P(\overline{r}'))\}=j} \sqrt{\Big(\sum_{\overline{r}''} \Gamma_x^{(j)}[\overline{r}, \overline{r}'']\Big)\Big(\sum_{\overline{r}''} \Gamma_x^{(j)}[\overline{r}'', \overline{r}']\Big)}. \tag{10}$$

Fix good sequences $\overline{r}$ and $\overline{r}'$ such that $\mathrm{mult}(x, P(\overline{r})) = j$ and $\mathrm{mult}(x, P(\overline{r}')) \geq j$. We can bound $\sum_{\overline{r}''} \Gamma_x^{(j)}[\overline{r}'', \overline{r}']$ in (10) from above by $\sum_{\overline{r}''} \Gamma_1[\overline{r}'', \overline{r}']$ which is $O(am^{2a+1})$ from (7). We claim that

$$\sum_{\overline{r}''} \Gamma_x^{(j)}[\overline{r}, \overline{r}''] \in O(m^{2a+1}). \tag{11}$$

The statement of the lemma follows then immediately using (10).

We now turn to the proof of (11). For $0 \leq i \leq 2a$, we define the paths $B(\overline{r}, i) = P(\overline{r}, 0)P(\overline{r}, 1) \ldots P(\overline{r}, i-1)$ ($B(r, i) = \emptyset$, when $i = 0$) and $T(\overline{r}, i) = P(\overline{r}, i)P(\overline{r}, i+1) \ldots P(\overline{r}, 2a)$. Observe that $P(\overline{r})$ is $B(\overline{r}, i)T(\overline{r}, i)$ for every i. We set $u = \max\{i \mid \ x \in P(\overline{r}, i)\}$, when $j > 0$, and $u = -1$ when $j = 0$. For each $0 \leq l \leq a-1$, we define

$$S_l = \sum_{\overline{r}'':\mathrm{div}(\overline{r},\overline{r}'')=2l+1} \Gamma_x^{(j)}[\overline{r}, \overline{r}''].$$

Depending on l we estimate S_l. Let $\overline{r}''$ be a random sequence (selected from good) with the condition that $\mathrm{div}(\overline{r}, \overline{r}'') = 2l+1$. Observe that $B(\overline{r}, 2l+1)$ and $B(\overline{r}'', 2l+1)$ may differ only on the elements of $N_{r_{2l}}$. Also, $\Gamma_x^{(j)}[\overline{r}, \overline{r}''] = 0$ when $\mathrm{mult}(x, P(\overline{r}'')) < j$.

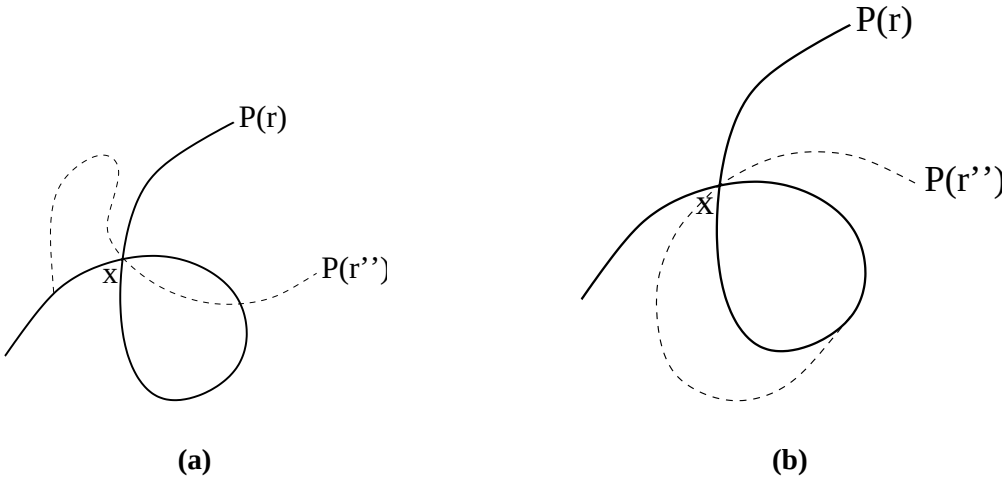


Figure 3: Figure (a) shows the situation when $2l < u$. Figure (b) shows the situation when $2l > u$. In both cases we win because the probability that $P(\overline{r}'')$ goes through x is small.

1. If $2l < u$ and $x \notin N_{r_{2l}}$ then $\mathrm{mult}(x, B(\overline{r}'', 2l+1)) = \mathrm{mult}(x, B(\overline{r}, 2l+1)) < \mathrm{mult}(x, P(\overline{r}))$. Therefore $\overline{r}''$ contributes to S_l only when $x \in T(\overline{r}'', 2l+1)$. Since x belongs to an inter-cluster path with probability at most $1/m$, the union bound implies that this happens with probability at most $2a/m$. Using (6) we conclude that $S_l \leq \frac{2a}{m} m^{2a+1}$.
2. If $2l > u$ and $x \notin N_{r_{2l}}$ then the only way that $f_{P(\overline{r})}(x) \neq f_{P(\overline{r}'')}(x)$ is when $x \in T(\overline{r}'', 2l+1)$. Therefore as above we conclude that $S_l \leq \frac{2a}{m} m^{2a+1}$.
3. If $2l = u$ or $x \in N_{r_{2l}}$ then we have only the trivial m^{2a+1} upper bound on S_l.

Because $\overline{r}$ is good there is only at most one l such that $x \in N_{r_{2l}}$. In summary, we can estimate the row-sum associated with $\overline{r}$ as

$$\sum_{0 \leq l \leq a-1} S_l \leq a\frac{2a}{m} m^{2a+1} + 2m^{2a+1} \in O(m^{2a+1}),$$

because by our choice $a \in \Theta(\sqrt{m})$. □

5.5 Putting things together

Proof of Theorem 1:

Proof. From Theorem 2 we know that $D(\mathrm{LS}_G) \in O(\log n(\Delta + s))$. We claim that

$$\log n(\Delta + s) \in O((\max\{\sqrt{\Delta}, \log n, (s/\Delta)^{1/8}/\log n\})^{19}).$$

Indeed, we have (trivially):

$$s \log n = (\sqrt{\Delta})^2 (\log n)^9 ((s/\Delta)^{1/8}/\log n)^8,$$

where the right hand side is clearly in $O((\max\{\sqrt{\Delta}, \log n, (s/\Delta)^{1/8}/\log n\})^{19})$. Since $\Delta \log n$ is obviously in $O((\max\{\sqrt{\Delta}, \log n, (s/\Delta)^{1/8}/\log n\})^{19})$, the claim follows.

Let us suppose first that G is connected. In that case $Q_\epsilon(\mathrm{LS}_G)$ is simultaneously in $\Omega(\sqrt{\Delta}), \Omega(\log n)$ and $\Omega((s/\Delta)^{1/8}/\log n)$ by Fact 1 and Theorems 3 and 4, and the statement follows.

Otherwise set $G_0 = G$ and let $G_1, \ldots, G_k$ be the connected components of G for some k. Set $d_i = D(\mathrm{LS}_{G_i})$ and $q_i = Q_\epsilon(\mathrm{LS}_{G_i})$ for $0 \leq i \leq k$. Let t be the index of the component for which $q_t = q_0$. Then $d_0 \leq d_t$ and $d_t \in O(q_t^{19})$ since G_t is connected. □

We conjecture that one can significantly improve on the exponent 19 in Theorem 1.

Acknowledgments

We would like to thank Katalin Friedl and Yves Verhoeven for helpful discussions.

References

[1] S. Aaronson. Lower bounds for local search by quantum arguments, *SIAM Journal on Computing* 35(4), pp. 804–824, 2006.

[2] S. Aaronson, and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems, *Journal of the ACM* 51(4), pp. 595–605, 2004.

[3] A. Aho, J. Hopcroft, and J Ullman. *Data Structures and Algorithms*, Addison-Wesley, 1983.

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

[5] A. Ambainis. Quantum lower bounds by quantum arguments, *Journal of Computer and System Sciences* 64, pp. 750–767, 2002.

[6] A. Ambainis. Polynomial degree vs. quantum query complexity, *Journal of Computer and System Sciences* 72(2), pp. 220–238, 2006.

[7] H. Barnum, M. Saks, and M. Szegedy. Quantum decision trees and semidefinite programming, in *Proc. of the 18th IEEE Conference on Computational Complexity*, pp. 179–193, 2003.

[8] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials, *Journal of the ACM* 48(4), pp. 778–797, 2001.

[9] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strength and weaknesses of quantum computing, *SIAM Journal on Computing* 26(5), pp. 1510–1523, 1997.

[10] H. Buhrman, and R. de Wolf. A lower bound for quantum search of an ordered list, *Information and Processing Letters* 70, pp. 205–209, 1999.

[11] X. Chen, and X. Deng. On the complexity of 2D discrete fixed point problem, in *Proc. of the 33rd International Colloquium on Automata, Languages and Programming*, pp. 489–500, 2006.

[12] D. Deutsch, and R. Jozsa. Rapid solution of problems by quantum computation, in *Proc. of the Royal Society A*, volume 439,1985.

[13] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems, *SIAM Journal on Computing* 35(6), pp. 1310–1328, 2006.

[14] K. Friedl, G. Ivanyos, M. Santha, and Y. Verhoeven. On the black-box complexity of Sperner's Lemma, in *Proc. of the 15th Fundamentals of Computation Theory*, LNCS 3623, pp. 245–257, 2005.

[15] L. Grover. A fast quantum mechanical algorithm for database search, in *Proc. of the 28th ACM Symposium on Theory of Computing*, pp. 212–219, 1996.

[16] P. Høyer, J. Neerbek, and Y. Shi. Quantum complexities of ordered searching, sorting, and element distinctness, *Algorithmica* 34(4), pp. 429–448, 2002.

[17] P. Høyer, T. Lee, and R. Špalek. Negative weights make adversaries stronger, in *Proc. of the 39th ACM Symposium on Theory of Computing*, pp. 526–535, 2007.

[18] D. Johnson, C. Papadimitriou, and M. Yannakakis. How easy is local search, *Journal of Computer and System Sciences* 37, pp. 429–448, 1988.

[19] S. Laplante, and F. Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments, in *Proc. of the 19th IEEE Conference on Computational Complexity*, pp. 294–304, 2004.

[20] D. Llewellyn, and C. Tovey. Dividing and conquering the square, *Discrete Applied Mathematics* 43, pp. 131–153, 1993.

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

[22] N. Megiddo, and C. Papadimitriou. On total functions, existence theorems, and computational complexity, *Theoretical Computer Science* 81, pp. 317–324, 1991.

[23] K. Menger. Zur allgemeinen Kurventhoerie, *Fundamenta Mathematicae* 10, pp. 96–115, 1927.

[24] D. Simon. On the power of quantum computation, *SIAM Journal on Computing* 26(5), pp. 1474–1783, 1997.

[25] R. Špalek. Quantum algorithms, lower bounds, and time-space tradeoffs, Ph. D. Thesis, ILLC Dissertation Series DS-2006-04, Universiteit van Amsterdam, 2006.

[26] R. Špalek, and M. Szegedy. All quantum adversary methods are equivalents, *Theory of Computing* 2, pp. 1–18, 2006.

[27] X. Sun, and A. Yao. On the quantum query complexity of local search in two and three dimensions, in *Proc. of the 47th IEEE Symposium on Foundations of Computer Science*, pp. 429–438, 2006.

[28] Y. Verhoeven. Enhanced algorithms for local search, *Information and Processing Letters* 97, pp. 171–176, 2006.

[29] C. Zalka. Grover's quantum searching algorithm is optimal, *Physical Review A* 60, pp. 2746–2751, 1999.

[30] S. Zhang. New upper and lower bounds for randomized and quantum local search, in *Proc. of the 38th ACM Symposium on Theory of Computing*, pp. 634–643, 2006.

[31] S. Zhang. On the power of Ambainis's lower bounds, *Theoretical Computer Science* 339, pp. 241–256, 2005.