On the Problem of Approximating the Eigenvalues of Undirected Graphs in Probabilistic Logspace

Dean Doron(✉) and Amnon Ta-Shma

The Blavatnik School of Computer Science,
Tel-Aviv University, 69978 Tel Aviv, Israel
deandoron@mail.tau.ac.il, amnon@tau.ac.il

Abstract. We introduce the problem of *approximating* the eigenvalues of a given stochastic/symmetric matrix in the context of classical space-bounded computation.

The problem can be *exactly* solved in $\mathsf{DET} \subseteq \mathsf{NC}^2$. Recently, it has been shown that the approximation problem can be solved by a *quantum* logspace algorithm. We show a BPL algorithm that approximates any eigenvalue with a *constant* accuracy. The result we obtain falls short of achieving the polynomially-small accuracy that the quantum algorithm achieves. Thus, at our current state of knowledge, we can achieve polynomially-small accuracy with quantum logspace algorithms, constant accuracy with probabilistic logspace algorithms, and no non-trivial result is known for deterministic logspace algorithms. The quantum algorithm also has the advantage of working over arbitrary, possibly non-stochastic Hermitian operators.

Our work raises several challenges. First, a derandomization challenge, trying to achieve a deterministic algorithm approximating eigenvalues with some non-trivial accuracy. Second, a de-quantumization challenge, trying to decide whether the quantum logspace model is strictly stronger than the classical probabilistic one or not. It also casts the deterministic, probabilistic and quantum space-bounded models as problems in linear algebra with differences between symmetric, stochastic and arbitrary operators. We therefore believe the problem of approximating the eigenvalues of a graph is not only natural and important by itself, but also important for understanding the relative power of deterministic, probabilistic and quantum logspace computation.

1 Introduction

A graph G can be associated with a linear operator A that describes a random walk on G. The operator A takes an especially simple form when G is undirected:

D. Doron—Supported by the Israel science Foundation grant no. 994/14, by the United States – Israel Binational Science Foundation grant no. 2010120 and by the Blavatnik Fund.

A. Ta-Shma—Supported by the Israel science Foundation grant no. 994/14 and by the United States – Israel Binational Science Foundation grant no. 2010120.

M.M. Halldórsson et al. (Eds.): ICALP 2015, Part I, LNCS 9134, pp. 419–431, 2015.
DOI: 10.1007/978-3-662-47672-7_34

If G is regular and undirected then A is symmetric and has a complete basis of orthonormal eigenvectors with real eigenvalues. In other words, there exists a unitary basis under which A is diagonal with real eigenvalues λ_i on the diagonal.

If G is undirected but not necessarily regular, then A is s diagonalizable with real eigenvalues, i.e., the picture is the same as before except that the basis is not necessarily unitary.[1]

If G is directed then A does not necessarily have a full basis of eigenvectors. In this case A (like any other linear operator) can be brought to its canonical Jordan Normal Form, where there exists a basis under which A is block-diagonal and each block has an eigenvalue λ on the main diagonal and 1 on the diagonal above it.

In this paper we raise the following natural questions:

- How difficult is it to approximate the largest eigenvalues of a general (not necessarily stochastic or non-negative) operator?
- How difficult is it to approximate all the spectrum of an operator?
- Does the problem become easy and belong to L when the graph is undirected?
- How about approximating the singular values of a graph?

1.1 The Bigger Picture

Derandomization is a major challenge of theoretical computer science. In the space-bounded model, Nisan [1] constructed a pseudo-random generator (PRG) against logarithmic space-bounded non-uniform algorithms that uses seed length $O(\log^2 n)$. Using that he showed BPL is contained in the class having simultaneously polynomial time and $O(\log^2 n)$ space. Saks and Zhou [2] showed BPL is contained in $\mathsf{DSPACE}(\log^{1.5} n)$. Reingold [3] showed that undirected st-connectivity (which was shown to be in RL by [4]) already belongs to L. These results seem to indicate that randomness does not add additional power to the model and many conjecture that in fact $\mathsf{BPL} = \mathsf{L}$. Yet, we currently do not know a PRG with seed length $o(\log^2 n)$, nor a general derandomization result that simultaneously uses $o(\log^2 n)$ space and polynomial time.

One can look up and ask which upper bounds we know on BPL. We then know the following:

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{RL} \subseteq \mathsf{NL} \subseteq \mathsf{DET} \subseteq \mathsf{NC}^2 \subseteq \mathsf{DSPACE}(O(\log^2 n)),$$

where DET is the class of languages that are NC^1 Turing-reducible to the problem *intdet* of computing the determinant of an integer matrix (see [5] for a definition

[1] If G is undirected and irregular, then the adjacency matrix $\tilde{A}$ is symmetric but the transition matrix $A = D^{-1}\tilde{A}$, where D is the diagonal degrees matrix, is not symmetric. Yet, consider the matrix $L = D^{-1/2}\tilde{A}D^{-1/2}$. L is symmetric and thus has an eigenvector basis with real eigenvalues. $A = D^{-1/2}LD^{1/2}$ is conjugate to L and thus is diagonalizable and has the same eigenvalues. As A is stochastic its eigenvalues are in the range $[-1, 1]$.

of DET). As it turns out, many important problems in linear algebra, such as inverting a matrix, or equivalently, solving a set of linear equations are in DET, and often complete for it (see, e.g., [5]). The fact that NL $\subseteq$ DET is due to [5] who showed that the directed connectivity problem, STCON is reducible to *intdet*. DET $\subseteq$ NC^2 follows from Csansky's algorithm [6] for the parallel computation of the determinant. In addition to the above we also know that BPL $\subseteq$ DET (e.g., using the fact that matrix powering is DET complete).

While matrix powering is complete for DET, *approximating* matrix powering of *stochastic* matrices is in BPL. To see that, assume A represents a stochastic matrix. Then one can approximate $A^k[s,t]$ by estimating the probability a random walk over A starting at s reaches t after k steps.[2] Conversely, it is possible to convert a BPL machine to a stochastic operator A such that the probability the machine moves from s to t in k steps is $A^k[s,t]$.[3] Thus, in a sense, approximating matrix-powering of stochastic operators is complete for BPL.

We now deviate from the classical picture we had so far and consider a quantum space-bounded model. In 1999, Watrous [7] defined the model of quantum logspace computation, and proved several facts on it. The definition was modified several times, see, [8]. Roughly speaking, a language is in BQL if there exists an L–uniform family of quantum circuits solving the language with only $O(\log n)$ qubits. The quantum circuits are over some universal basis of gates (e.g., CNOT, HAD, T) plus intermediate measurements (that in particular may simulate a stream of random coins). For details we refer the reader to [8,9]. The works of Watrous, van Melkebeek and Watson showed that BQL is also contained in NC^2.

Recently, it was shown in [9], building on an earlier work by [10], that it is possible to *approximate* the singular value decomposition (SVD) of a given linear operator in BQL. This also implies that it is possible to approximately invert a matrix in BQL. A natural question left open by this work is:

Open Problem: Is it possible to approximate the SVD of an arbitrary linear operator in BPL? The problem is also open for Hermitian operators, where singular values and eigenvalues coincide (up to their sign).

In fact, this question is open also when the operator is the transition matrix arising from a walk on a regular, undirected graph.

Thus, somewhat surprisingly, we see that the deterministic / probabilistic / quantum space-bounded classes and the class DET are capable of doing some sort of linear algebra on corresponding operators. Namely,

– In DET we can compute exactly the determinant which is the product of all eigenvalues as well as the product of all singular values. We can also solve matrix powering. Both problems are *complete* in DET. With that we can approximately invert an operator or perform the SVD decomposition.

[2] For completeness we include a proof of this in Appendix A. We also extend the class for which this works to matrices with non-negative or complex entries as long as their infinity norm is at most 1.

[3] This reduction is standard and appears in many papers, e.g., already in [1].

In a sense, DET is an *exact* computation of the spectrum (e.g., in terms of the characteristic polynomial) of an *arbitrary* linear operator.
- BQL is capable of approximating the whole singular value decomposition of any operator. This is somewhat equivalent to saying that BQL is capable of *approximating* the eigenvalues of *Hermitian* operators.
- BPL is capable of *approximating* matrix powering. In this paper we will show BPL can approximate any eigenvalue of an *undirected* graph with constant accuracy. We do not know yet whether we can do the same for *directed* graphs or whether we can approximate the *whole spectrum* of undirected graphs.
- In L we do not know how to do any of the above, but Reingold showed L is capable of solving USTCON, i.e., connectivity on undirected graphs. Notice that undirected graphs roughly correspond to the intersection of stochastic and Hermitian operators.

1.2 On the Problem of Approximating Arbitrary Eigenvalues of Undirected Graphs in BPL

We define the following promise problem:

Definition 1. *($EV_{\alpha,\beta}$) The input is a stochastic, Hermitian matrix A, $\lambda \in [-1,1]$ and $\alpha < \beta$.*

Yes instances *: There is an eigenvalue λ_i of A such that $|\lambda_i - \lambda| \leq \alpha$.*
No instances *: All eigenvalues of A are β–far from λ.*

One way to design a BPL algorithm for the problem is by "de-quantumizing" the quantum algorithm.[4] The BQL algorithm solves the above problem for any Hermitian operator A whose eigenvalues are τ–separated, for, say, $\tau = n^{-c}$, $\alpha = \frac{\tau}{4}$ and $\beta = 2\alpha$. That is, the quantum algorithm can handle any polynomially small accuracy. With such accuracy one can turn the solution of the promise problem to a procedure approximating the whole spectrum.

We develop a BPL algorithm that follows the main idea of the quantum algorithm, and in that sense we de-quantumize the quantum algorithm, but we achieve much worse parameters. Specifically, we prove that the promise problem $EV_{\alpha,\beta}$ belongs to BPL, for *constant* parameters $\alpha < \beta$. On the one hand the result is disappointing because the quantum algorithm does so much better and can handle polynomially small gaps. On the other hand, we remark that we do not know how to achieve even constant approximation with a deterministic logspace algorithm. We are not aware of many natural promise problems in BPL that are not known to be in L. This paper shows $EV_{\alpha,\beta}$ is such a promise problem.

[4] We remark that Ben-Or and Eldar [11] recently de-quantumized the SVD quantum algorithm and obtained a classical probabilistic algorithm for inverting matrices that achieves the state of the art running time, using a completely new approach that is derived from the quantum algorithm. We would like to do the same in the space-bounded model.

1.3 Our Technique

The usual way of describing the quantum algorithm is that it applies quantum phase estimation on the completely mixed state. The completely mixed state is a uniform mixture of the pure states that are formed from the eigenvectors of A, and on each such eigenvector, the quantum phase estimation estimates the corresponding eigenvalue. Thus, if the procedure can be run in (quantum) logarithmic space, we essentially sample a random eigenvector/eigenvalue pair, and from that we can approximately get the SVD decomposition of A.

Another (less standard) way of viewing the quantum algorithm is that it manipulates the eigenvalues of an input matrix A without knowing the decomposition of A to eigenvectors and eigenvalues. This can be done using the simple fact that if $\lambda_1, \ldots, \lambda_n$ are the roots of the characteristic polynomial of A, and if p is an arbitrary univariate polynomial, then $p(\lambda_1), \ldots, p(\lambda_n)$ are the roots of the characteristic polynomial of the matrix $p(A)$. The probability the algorithm measures λ is proportional to Tr $(p(A))$, where p is a shift of the Fejér kernel by λ (see, e.g., [12, Chapter2]). Applying p on A amplifies the eigenvalues that are close to λ to a value close to 1, and damps eigenvalues far from λ close to 0. Thus, Tr $(p(A))$ approximately counts the number of eigenvalues close to λ.

We would like to follow the same approach but with a probabilistic algorithm rather than a quantum one. We say a matrix A is *simulatable* if a probabilistic logspace algorithm can approximate $A^k[s,t]$ for any k polynomial in n and with polynomially-small accuracy (see Definition 2 for the exact details). From the discussion above it is clear that if A is the transition matrix of a (directed or undirected) graph then A is simulatable (see Lemma 1). We remark that in the appendix we show that even non-stochastic matrices A with negative or complex entries are simulatable as long as A has infinity norm at most 1, namely, those matrices A for which all rows $i \in [n]$ have ℓ_1 norm at most 1, $\sum_j |A[i,j]| \leq 1$.

If A is simulatable and the coefficients of $p(x) = \sum_i c_i x^i$ are not too large (i.e., only polynomially large in n), then we can approximate in BPL the matrix $p(A) = \sum_i c_i A^i$. In particular, we can also approximate Tr $(p(A))$. By taking p to be a threshold polynomial with degree logarithmic in n (that guarantees the size of the coefficients c_i is polynomial in n) and a threshold around λ, we can solve $EV_{\alpha,\beta}(A)$ for constants $\alpha < \beta$ (see Section 3).

There are many other possible candidate functions for a threshold polynomial p. However, we prove in Theorem 2 that no polynomial can do significantly better than a threshold polynomial. The reason the quantum algorithm works better is because it is able to take p up to some polynomial degree (rather than logarithmic degree) not worrying about the (quite large) size of the coefficients, thus leading to much better accuracy. The quantum algorithm also has the advantage that it works for any normal operator A, not necessarily stochastic or simulatable.

Thus, the algorithm we give for $EV_{\alpha,\beta}$ is simple: Approximate Tr $(p(A))$ to a simple logarithmic degree polynomial p. Nevertheless, we believe it features a new component that has not been used before by probabilistic space-bounded algorithms. An algorithm that takes a random walk on a graph and takes a decision based on the walk length and connectivity properties of the graph (as,

e.g., [4]) works with some power of the input matrix A. More generally, such an algorithm can work with a convex combination of powers of the input matrix (by probabilistically choosing which power to take). The algorithm we present utilizes *arbitrary* (positive or *negative*) combinations of matrix powers and we believe it is a crucial feature of the solution. We are not aware of previous BPL algorithms using such a feature.

The approach above does not work for approximating the eigenvalues of a *directed* graph G. It is still true that the resulting operator A is stochastic and therefore simulatable. Also, it remains true that if λ is an eigenvalue of A (i.e., a root of the characteristic polynomial) then $p(\lambda)$ is a root of $p(A)$. However, since A is not Hermitian, the eigenvalues λ of A may be complex and we do not know how to control $p(\lambda)$ when p may have both negative and positive coefficients. We believe it should be possible to approximate in BPL an arbitrary eigenvalue of any stochastic operator (not necessarily Hermitian) to within constant accuracy, but we have not been able to show it so far.

1.4 A Short Discussion

We believe the problem of approximating the eigenvalues of an undirected graph is natural and important. Also, at our current state of knowledge, it simultaneously separates deterministic, probabilistic and quantum complexity: In BQL we can solve it with polynomially-small accuracy, in BPL with constant accuracy and in L we do not know how to solve it at all. Thus it poses several challenges:

- First, there is the natural question of whether one can approximate eigenvalues in BPL with better accuracy. A positive answer would imply BPL approximations to many important linear algebra problems that are currently only known to be in NC^2. A negative answer would imply a separation between BQL and BPL.
- Second, it raises the natural question of derandomization. Can one design a *deterministic* algorithm approximating eigenvalues to constant accuracy?

We believe the solution of this problem is not only important by itself, but may also shed new light on the strengths and weaknesses of the space-bounded model, and the relative strengths of the deterministic, probabilistic and quantum models of space-bounded computation.

2 Preliminaries

Often we are interested in approximating a *value* (e.g., an entry in a matrix with integer values or the whole matrix) with a probabilistic machine. More precisely, assume there exists some value $u = u(x) \in \mathbb{R}$ that is determined by the input $x \in \{0,1\}^n$. We say a probabilistic TM $M(x,y)$ (ε,δ)–approximates $u(x)$ if:

$$\forall_{x\in\{0,1\}^n} \quad \Pr_y \left[|M(x,y) - u(x)| \geq \varepsilon\right] \leq \delta. \tag{1}$$

A random walk on a graph G (or its transition matrix A) can be simulated by a probabilistic logspace machine. As a consequence, a probabilistic logspace machine can approximate powers of A well. Here we try to extend this notion to arbitrary linear operators A, not necessarily stochastic. We say a matrix A is *simulatable* if any power of it can be approximated by a probabilistic algorithm running in small space. Formally:

Definition 2. *We say that a family of matrices $\mathcal{A}$ is* simulatable *if there exists a probabilistic algorithm that on input $A \in \mathcal{A}$ of dimension n with $\|A\| \leq poly(n)$, $k \in \mathbb{N}$, $s, t \in [n]$, runs in space $O(\log \frac{nk}{\varepsilon\delta})$ and (ε, δ)–approximates $A^k[s,t]$.*

In Appendix A we give for completeness a proof that:

Lemma 1. *The family of transition matrices of (directed or undirected) graphs is simulatable.*

We say $\|A\|_\infty \leq c$ if for every $i \in [n]$, $\sum_j |A[i,j]| \leq c$. In the same Appendix we also show:

Lemma 2. *The family of real matrices with infinity norm at most* 1 *is simulatable.*

3 Approximating Eigenvalues with Constant Accuracy

In this section we prove:

Theorem 1. *There exists a probabilistic algorithm that on input a stochastic matrix B with real eigenvalues in $[0,1]$, constants $\beta > \alpha > 0$ and $\lambda \in [0,1]$ such that:*

- *There are d eigenvalues λ_i satisfying $|\lambda - \lambda_i| \leq \alpha$,*
- *All other eigenvalues λ_i satisfy $|\lambda - \lambda_i| \geq \beta$,*

outputs d with probability at least $2/3$. Furthermore the algorithm runs in probabilistic space $O(\log n)$.

We remark that Theorem 1 covers the case of transition matrices of undirected graphs. As mentioned earlier, a transition matrix A of an undirected graph has an eigenvector basis with real eigenvalues in the range $[-1,1]$. Taking $B = \frac{1}{2}A + \frac{1}{2}I_{n\times n}$ we get a stochastic matrix with eigenvalues in the range $[0,1]$, and whose eigenvectors are in a natural one-to-one correspondence with A's eigenvalues.

Proof. (Of Theorem 1) The input to the algorithm is $n, B, \lambda, \alpha, \beta$. We assume a univariate polynomial $p(x) = \sum_{i=0}^{M} c_i x^i$ with the following properties:

- p has a sharp peak around λ, i.e., $p(x) \geq 1 - \eta$ for $x \in [\lambda - \alpha, \lambda + \alpha]$ and $p(x) \leq \eta$ for $x \in [0,1] \setminus (\lambda - \beta, \lambda + \beta)$, where $\eta = \eta(n) = n^{-2}$.

– p can be computed in L. Formally, $M = \deg(p)$ and $|c_i|$ are at most $\text{poly}(n)$ and for every i, c_i can be computed (exactly) by a deterministic Turing machine that uses $O(\log n)$ space.

In the next subsection we show how to obtain such a polynomial p with $M = 32(\beta-\alpha)^{-2}\log n$ and $|c_i| \leq 2^{O(M)}$.

Choose $\varepsilon = \frac{1}{n}$ and $\delta = \frac{1}{3}$. Set $\varepsilon' = \varepsilon \cdot 2^{-2M}$ and $\delta' = \delta \cdot 2^{-M}$. The output of the algorithm is the integer closest to

$$R = \sum_{i=0}^{M} c_i \cdot \text{TP}(B, n, i, \varepsilon', \delta')$$

where TP is the probabilistic algorithm guaranteed by Lemma 2 that (ε', δ')–approximates Tr (B^i).

It is easy to check that:

Claim. $\Pr[|R - \text{Tr}\ (p(B))| \geq \varepsilon] \leq \delta$.

As Tr $(p(B)) = \sum_{i=1}^{n} p(\lambda_i)$, $\Pr[|R - \sum_{i=1}^{n} p(\lambda_i)| \geq \varepsilon] \leq \delta$. However, $p(\lambda_i)$ is large when λ_i is α–close to λ and small when it is β–far from λ, and we are promised that *all* eigenvalues λ_i are either α–close or β–far from λ. Thus,

$$|\text{Tr}\ (p(B)) - d| \leq n\eta.$$

Altogether, except for probability δ, $|R-d| \leq \varepsilon + n\eta \leq \frac{1}{3}$, and the nearest integer closest to R is d. The correctness follows. It is also straightforward to check that the space complexity is $O(\log(n\varepsilon^{-1}\delta^{-1})) = O(\log n)$.

The constant accuracy we achieve is far from being satisfying. The matrix B has n eigenvalues in the range $[0, 1]$, so the average distance between two neighboring eigenvalues is $1/n$. Thus, the assumption that there is an interval of length $\beta - \alpha$ with no eigenvalue is often not true. The desired accuracy we would like to get is $o(1/n)$. Having such accuracy would enable outputting an approximation of the whole spectrum of B, using methods similar to those in [9], thus getting a true classical analogue to the quantum algorithm in [9]. However, we do not know how to achieve subconstant accuracy. The question whether better accuracy is possible in BPL is one of the main questions raised by this work.

3.1 Using the Symmetric Threshold Functions

There are several natural candidates for the function p above. In this subsection we use the threshold function to obtain such a function p. For $\lambda = \frac{k}{M}$ for some integers k and M, define:

$$p_\lambda(x) = \sum_{i=k}^{M} \binom{M}{i} x^i (1-x)^{M-i}.$$

p_λ approximates well the threshold function $\mathbf{Th}_\lambda(x) : [0,1] \to \{0,1\}$ that is one for $x \geq \lambda$ and zero otherwise. Specifically, using the Chernoff bound, we obtain:

Lemma 3. *Let $x \in [0,1]$. $p_\lambda(x)$ approximates $\mathbf{Th}_\lambda(x)$ over $[0,1]$ with accuracy $(\xi(\varepsilon))^{Mx}$, where $\varepsilon = \frac{\lambda - x}{x}$ and $\xi(\varepsilon) = \frac{e^\varepsilon}{(1+\varepsilon)^{1+\varepsilon}}$.*

As a polynomial in x, $p_\lambda(x) = \sum_{i=0}^{M} c_i x^i$ with $c_i = (-1)^i \sum_{j=\lambda M}^{i} \binom{M}{j}\binom{M-j}{i-j}(-1)^j$ and therefore $|c_i| \leq \sum_{j=\lambda M}^{i} \binom{M}{j}\binom{M-j}{i-j} \leq M\binom{M}{M/2}^2 = 2^{O(M)}$. Furthermore, c_i can be computed (exactly) by a deterministic Turing machine that uses $O(M)$ space by simply running through the loop over j, each time updating the current result by $(-1)^j \binom{M}{j}\binom{M-j}{i-j}$.

To obtain our polynomial p, define p as the difference between the threshold polynomial around $\lambda + \Delta$ and the threshold polynomial around $\lambda - \Delta$,

$$p(x) = p_{\lambda-\Delta}(x) - p_{\lambda+\Delta}(x)$$

where $M = 32(\beta - \alpha)^{-2} \log n$ and $\Delta = (\alpha + \beta)/2$. It is easy to check that

Lemma 4. *$p(x) \geq 1 - n^{-2}$ for every x that is α–close to λ (i.e., $|x - \lambda| < \alpha$) and $p(x) \leq n^{-2}$ for every x that is β–far from λ (i.e., $|x - \lambda| \geq \beta$).*

3.2 The Limitation of the Technique

In this subsection, we prove the accuracy of the above technique cannot be enhanced merely by choosing a different polynomial p. Approximating threshold functions by a polynomial is well-studied and well understood (see, for example, [13–15] and references therein). However, we need to adapt this work to our needs because we have an additional requirement that the magnitude of the polynomial's coefficients is small.

We start by formalizing the properties of p that were useful to us. We say that $\mathcal{P} = \{p_{\lambda,n}\}_{\lambda \in [0,1], n \in \mathbb{N}}$ is a family of polynomials if for every $\lambda \in [0,1]$ and $n \in \mathbb{N}$, $p_{\lambda,n}$ is a univariate polynomial with coefficients in $\mathbb{R}$.

Definition 3. *(Small family) Let $\mathcal{P}$ be a family of polynomials and fix $\lambda \in [0,1]$. For every $n \in \mathbb{N}$, write $p_{\lambda,n}(x) = \sum_{i=0}^{\deg(p_{\lambda,n})} c_{\lambda,n,i} x^i$. We say the family is $s(n)$–small if,*

- $\deg(p_{\lambda,n}) \leq 2^{s(n)}$,
- *For every $0 \leq i \leq \deg(p_{\lambda,n})$, $|c_{\lambda,n,i}| \leq 2^{s(n)}$, and*
- *There exists a deterministic Turing machine running in space $s(n)$ that outputs $c_{\lambda,n,0}, \ldots, c_{\lambda,n,\deg(p_{\lambda,n})}$.*

Definition 4. *(Distinguisher family) Let $\mathcal{P}$ be a family of polynomials and fix $n \in \mathbb{N}$. Given $\alpha < \beta$ in $(0,1)$ and $\eta < 1/2$, we say the family is (α, β, η)–distinguisher for $\lambda \in [0,1]$ if,*

– *For every* $x \in [0,1]$ *that is* α*–close to* λ, $p_{\lambda,n}(x) \in [1-\eta, 1]$, *and*
– *For every* $x \in [0,1]$ *that is* β*–far from* λ, $p_{\lambda,n}(x) \in [0, \eta]$.

Theorem 2. *Let* $\alpha, \beta, \lambda, \eta$ *be such that* $\alpha \leq \beta$, $\beta = o(1)$, $\eta = o(n^{-1})$ *and* $\lambda + \beta \leq \frac{1}{2}$. *Then there is no* (α, β, η)*–distinguisher family for* λ *that is* $O(\log n)$*–small.*

Proof. Assume there exists such a family $\{p_{\lambda,n}\}_{\lambda \in [0,1], n \in \mathbb{N}}$ with $s(n) = c' \log n$. We first show that without loss of generality p has logarithmic degree. Let $r_{\lambda,n}(x)$ be the residual error of truncating $p_{\lambda,n}(x)$ after $c \log n$ terms, for c that will soon be determined. Also, w.l.o.g., assume $x \in [0,1)$ is bounded away from 1. Then:

$$r_{\lambda,n}(x) \leq \sum_{i=c\log n+1}^{\deg(p_{\lambda,n})} |c_{\lambda,n,i}| \cdot x^i \leq n^{c'} \cdot \frac{x^{c \log n}}{1-x} \leq \frac{1}{1-x} n^{c' - c\log(1/x)}.$$

So, by taking $c = \lceil \frac{c'+2-\log(1-x)}{\log(1/x)} \rceil$ we obtain $r_{\lambda,n}(x) \leq n^{-2}$.

We now show that $O(\log n)$–degree polynomials cannot decay around λ fast enough. Assume to the contrary that there exists such a distinguisher family, so $|p_{\lambda,n}(x)| < n^{-1}$ for $x \in [\lambda + \beta, 1]$. The following lemma states that if a function has a small value on an interval, than it cannot be too large outside it. Namely,

Lemma 5. *[16, Theorem 2.9.11] Let* $T_n(x)$ *be the Chebyshev polynomial (of the first kind) of degree* n. *Then, if the polynomial* $P_n(x) = \sum_{i=0}^{n} c_i x^i$ *satisfies the inequality* $|P_n(x)| \leq L$ *on the segment* $[a,b]$ *then at any point outside the segment we have*

$$|P_n(x)| \leq L \cdot \left| T_n\left(\frac{2x-a-b}{b-a} \right) \right|.$$

For properties of the Chebyshev polynomials see [17, Chapter 1.1]. We mention a few properties that we use. An explicit representation of $T_n(x)$ is given by $T_n(x) = \frac{\left(x-\sqrt{x^2-1}\right)^n + \left(x+\sqrt{x^2-1}\right)^n}{2}$. $|T_n(-x)| = |T_n(x)|$ and T_n is monotonically increasing for $x > 1$. Also,

$$|T_n(1+\delta)| \leq \left(1 + \delta + \sqrt{(1+\delta)^2 - 1}\right)^n \leq \left(1 + 4\sqrt{\delta}\right)^n \leq e^{4n\sqrt{\delta}} \leq 2^{8n\sqrt{\delta}} \quad (2)$$

for $0 \leq \delta \leq 1$. Then:

$$\begin{aligned} |p_{\lambda,n}(\lambda)| &\leq n^{-1} \cdot \left| T_{c\cdot\log n}\left(\tfrac{\lambda-\beta-1}{-\lambda-\beta+1} \right) \right| \\ &= n^{-1} \cdot \left| T_{c\cdot\log n}\left(1 + \tfrac{2\beta}{1-\lambda-\beta} \right) \right| && \text{By } |T_n(x)| = |T_n(-x)| \\ &\leq n^{-1} \cdot |T_{c\cdot\log n}(1+4\beta)| && \text{By the monotonicity of } T_n(x) \text{ for } x > 1 \text{ and } \lambda + \beta \leq \tfrac{1}{2} \end{aligned}$$

By Equation (2) $|p_{\lambda,n}(\lambda)| \leq n^{-1} 2^{32c\sqrt{\beta}\log n} \leq n^{-1+32c\sqrt{\beta}}$. As $\beta = o(1)$ for n large enough we have $|p_{\lambda,n}(\lambda)| \leq n^{-1/2}$, contradicting the fact that $|p_{\lambda,n}(\lambda)| \geq 1 - n^{-1}$.

We note that for values very close to 1, polynomials of higher degrees are useful, and indeed better approximations are possible. In particular, one can separate a 1 eigenvalue from $1 - \frac{1}{n}$ by using the polynomial x^{n^2}.

References

1. Nisan, N.: Pseudorandom generators for space-bounded computation. Combinatorica **12**, 449–461 (1992)
2. Saks, M.E., Zhou, S.: $\mathrm{BP}_H\mathrm{SPACE(S)} \subseteq \mathrm{DSPACE}(\mathrm{S}^{3/2})$. J. Comput. Syst. Sci. **58**, 376–403 (1999)
3. Reingold, O.: Undirected connectivity in log-space. J. ACM **55** (2008)
4. Aleliunas, R., Karp, R.M., Lipton, R., Lovasz, L., Rackoff, C.: Random walks, universal traversal sequences, and the complexity of maze problems. In: 20th Annual Symposium on Foundations of Computer Science, pp. 218–223 (1979)
5. Cook, S.A.: A taxonomy of problems with fast parallel algorithms. Information and Control **64** (1985); International Conference on Foundations of Computation Theory
6. Csansky, L.: Fast parallel matrix inversion algorithms. SIAM Journal of Computing **5**, 618–623 (1976)
7. Watrous, J.: Space-bounded quantum complexity. Journal of Computer and System Sciences **59**, 281–326 (1999)
8. van Melkebeek, D., Watson, T.: Time-space efficient simulations of quantum computations. Electronic Colloquium on Computational Complexity (ECCC) **17**, 147 (2010)
9. Ta-Shma, A.: Inverting well conditioned matrices in quantum logspace. In: Proceedings of the 45th Annual ACM Symposium on Symposium on Theory of Computing, STOC 2013, pp. 881–890. ACM, New York (2013)
10. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Phys. Rev. Lett. **103**, 150502 (2009)
11. Ben-Or, M., Eldar, L.: Optimal algorithms for linear algebra by quantum inspiration. CoRR abs/1312.3717 (2013)
12. Hoffman, K.: Banach Spaces of Analytic Functions. Dover Books on Mathematics Series. Dover Publications, Incorporated (2007)
13. Saff, E.B., Totik, V.: Polynomial approximation of piecewise analytic functions. Journal of the London Mathematical Society **s2–39**, 487–498 (1989)
14. Eremenko, A., Yuditskii, P.: Uniform approximation of sgn(x) by polynomials and entire functions. Journal d'Analyse Mathématique **101**, 313–324 (2007)
15. Diakonikolas, I., Gopalan, P., Jaiswal, R., Servedio, R.A., Viola, E.: Bounded independence fools halfspaces. SIAM Journal on Computing **39**, 3441–3462 (2010)
16. Timan, A.: Theory of Approximation of Functions of a Real Variable. Dover books on advanced mathematics. Pergamon Press (1963)
17. Rivlin, T.: The Chebyshev polynomials. Pure and applied mathematics. Wiley (1974)

A Simulatable Matrices

Lemma 1. *The family of transition matrices of (directed or undirected) graphs is simulatable.*

Proof. Let $G = (V, E)$ be a graph with n vertices and let A be its transition matrix. Let $k \in \mathbb{N}$, $s, t \in [n]$ and $\delta, \varepsilon > 0$. Consider the algorithm that on input k, s, t, takes T independent random walks of length k over G starting at vertex

s. The algorithm outputs the ratio of walks that reach vertex t. Let Y_i be the random value that is 1 if the i-th trial reached t and 0 otherwise. Then, for every i, $\mathbb{E}[Y_i] = A^k[s,t]$. Also, $Y_1, \ldots, Y_T$ are independent. By Chernoff,

$$\Pr[|\frac{1}{T}\sum_{i=1}^{T} Y_i - A^k[s,t]| \geq \varepsilon] \leq 2e^{-2\varepsilon^2 T}$$

Taking $T = \text{poly}(\varepsilon^{-1}, \log \delta^{-1})$, the error probability (i.e., getting an estimate that is ε far from the correct value) is at most δ. Altogether, the algorithm runs in space $O(\log(Tnk|E|)) = O(\log(nk\varepsilon^{-1}) + \log\log\delta^{-1})$, assuming $|E| = \text{poly}(n,k)$.

We say $\|A\|_\infty \leq c$ if for every $i \in [n]$, $\sum_j |A[i,j]| \leq c$. We show:

Lemma 2. *The family of real matrices with infinity norm at most* 1 *is simulatable.*

Proof. We prove the result to real matrices, with positive or negative entries, as long as they have bounded infinity norm. By generalizing the sign of an entry to its *phase*, the result easily applies to complex matrices as well.

Let A be a real matrix of dimension n such that $\|A\|_\infty \leq 1$. Let $d_i(A) = \sum_j |A[i,j]|$.Let $k \in \mathbb{N}$, $s,t \in [n]$ and $\delta, \varepsilon > 0$. Note that:

$$\begin{aligned} A^k[s,t] &= \sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\cdots\sum_{i_{k-1}=1}^{n} A[s,i_1]\cdot A[i_1,i_2]\cdot\ldots\cdot A[i_{k-1},t] \\ &= \sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\cdots\sum_{i_{k-1}=1}^{n} \frac{|A[s,i_1]|}{d_s(A)}\cdot\frac{|A[i_1,i_2]|}{d_{i_1}(A)}\cdot\ldots\cdot \\ &\quad \times\frac{|A[i_{k-1},t]|}{d_{i_{k-1}}(A)}\cdot p\left(A,\langle s,i_1,i_2,\ldots,i_{k-1},t\rangle\right), \end{aligned}$$

where

$$p\left(A,\langle s,i_1,i_2,\ldots,i_{k-1},t\rangle\right) = \frac{d_s(A)\cdot d_{i_1}(A)\cdot\ldots\cdot d_{i_{k-1}}(A)}{\text{sgn}\left(A[s,i_1]\cdot A[i_1,i_2]\cdot\ldots\cdot A[i_{k-1},i_t]\right)}.$$

Consider the algorithm that on input k,s,t, takes T independent random walks of length k over G starting from vertex s. Iterating over all random walks, the algorithm approximates $\frac{1}{T}\sum_i y(i)$, where $y(i) = p(A,i)$ if the walk i reached t, and 0 otherwise. Correspondingly, let Y_i be the random value that is $p(A,i)$ if the i'th walk reached t and 0 if it did not. Then,

$$\mathbb{E}[Y_i] = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_{k-1}=1}^{n} A[s, i_1] \cdot A[i_1, i_2] \cdot \ldots \cdot A[i_{k-1}, t] \cdot p(A, \langle s, i_1, \ldots, i_{k-1}, t \rangle) = A^k[s, t].$$

Denote the algorithm's outcome by $M(k, s, t)$. As in Lemma 1, and using the fact that $|p(A, i)| \leq 1$, the algorithm can (ε, δ)–approximates $\mathbb{E}[Y_i]$ by choosing T which is $\text{poly}(\varepsilon^{-1}, \log \delta^{-1})$. Following the same analysis as of Lemma 1, the algorithm runs in $O(\log nk\varepsilon^{-1} + \log \log \delta^{-1})$ space. We conclude that A is simulatable.