On the de-randomization of space-bounded approximate counting problems

Dean Doron[a,1,*], Amnon Ta-Shma[a,1]

[a] *The Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel 6997801*

Abstract

It was recently shown that SVD and matrix inversion can be approximated in quantum log-space [1] for well formed matrices. This can be interpreted as a fully logarithmic quantum approximation scheme for both problems. We show that if $\mathsf{prBQL} = \mathsf{prBPL}$ then every fully logarithmic quantum approximation scheme can be replaced by a probabilistic one. Hence, if classical algorithms cannot approximate the above functions in logarithmic space, then there is a gap already for languages, namely, $\mathsf{prBQL} \neq \mathsf{prBPL}$.

On the way we simplify a proof of Goldreich for a similar statement for time bounded probabilistic algorithms. We show that our simplified algorithm works also in the space bounded setting (for a large set of functions) whereas Goldreich's approach does not seem to apply in the space bounded setting.

Keywords: Computational complexity, Approximation algorithms, Randomized algorithms, Space bounded computation, Space bounded quantum computation, Space bounded approximation schemes

1. Introduction

Two well known approximation problems are approximating the singular value decomposition (SVD) of a matrix and approximating matrix inverse. Both problems were extensively studied in the *time bounded* model, e.g., in [2, 3, 4]. In the *space bounded* model it was recently shown that SVD and matrix inversion can be additively approximated by a *quantum* algorithm using only logarithmic space [1]. This can be interpreted as a fully logarithmic quantum approximation scheme (with additive accuracy) for the problems.

The result [1] shows a possible gap between quantum and classical algorithms for the task of *approximating a function*. It is not clear, however, whether it also implies a possible gap between the power of quantum and classical log-space algorithms for *decision problems*. Thus, a natural question is the following. Suppose no classical log-space algorithm can approximate the SVD. Does that imply that $\mathsf{prBQL} \neq \mathsf{prBPL}$? In the contra-positive we ask whether a "de-quantumization" of decision classes (i.e., $\mathsf{prBQL} = \mathsf{prBPL}$) implies a de-quantumization of approximation schemes.

The question was also asked in the model of classical time bounded computations. A classical result by Stockmeyer [5] implies that the problem of *approximate counting* of a general NP predicate can be solved in $\mathsf{FBPP}^{\mathsf{NP}}$. Shaltiel and Umans [6] de-randomized the result under the assumption that $\mathsf{E}_{\|}^{\mathsf{NP}}$ requires exponential size single-valued nondeterministic circuits [2].

While in general we do not know how to approximate functions in #P better than in $\mathsf{FBPP}^{\mathsf{NP}}$, there exist #P-complete functions for which there exists a fully polynomial randomized approximation scheme (FPRAS) that *does not* require an oracle access to an NP language. Jerrum, Sinclair and Vigoda [7] proved this for the permanent. Goldreich [8] showed that derandomizing the FPRAS for the permanent (or any other FPRAS) can be done under the assumption that $\mathsf{prBPP} = \mathsf{P}$:

Theorem 1. *[8, Corollary 3.6] If* $\mathsf{prBPP} = \mathsf{P}$ *then every function that has an FPRAS also has such a*

*Corresponding author. Tel.: +972 54 5865755.
Email addresses: deandoron@mail.tau.ac.il (Dean Doron), amnon@tau.ac.il (Amnon Ta-Shma)

[1] Supported by the Israel science Foundation grant no. 994/14 and by the United States – Israel Binational Science Foundation grant no. 2010120.

[2] $\mathsf{E}_{\|}^{\mathsf{NP}}$ is the class E with *non-adaptive* oracle queries to NP. A *single-valued* nondeterministic circuit is the nonuniform analogue of $\mathsf{NP} \cap \mathsf{coNP}$.

deterministic scheme. Furthermore, for every polynomial p, there exists a polynomial p' such that if the probabilistic scheme runs in time p, then the deterministic one runs in time p'.

In [8] Goldreich is after a "uniform" pseudorandom generator (see [8] for exact details) and Theorem 1 is a corollary of a more general technique used in his construction of a uniform PRG. Roughly speaking, Goldreich's argument works as follows: Given a probabilistic Turing machine $M(x, y)$ that approximates $f(x)$ well (where x is the input and y is the internal random coins used by M), we construct a specific sequence y_0 such that $M(x, y_0)$ also approximates $f(x)$ well. The bits of y are fixed bit by bit, where each bit is determined by a single query to a certain prBPP problem that depends on the random bits that were set so far.

In our case we deal with space bounded problems. It seems Goldreich's approach does not generalize to the space bounded case, as the random coins string is kept on the work tape, and its length is bounded by the time complexity of the Turing machine – which can be polynomial.

In this note we give an alternative (and simpler) proof that directly computes the approximated value using prBPP oracle calls, and we show this approach does generalize to a large class of functions in the space bounded model. In particular we show that if no probabilistic algorithm can approximately compute the SVD of certain well formed matrices[3] – a task that quantum algorithms can do with logarithmic space – then there is already a separation of the decision classes and $\mathsf{prBQL} \neq \mathsf{prBPL}$. We give an intuitive explanation of our proof technique at the beginning of Section 3 followed by a formal argument.

Throughout the paper we use the notations of common complexity classes freely. Exact definitions can be found, for example, in [9]. For a complexity decision class $\mathcal{C}$, we denote $\mathsf{pr}\mathcal{C}$ as the corresponding promise class and $\mathsf{F}\mathcal{C}$ as the corresponding function class. In Section 2 we define space bounded approximation schemes. We remark that we are not aware of any previous definition of space bounded approximation schemes. In Section 3 we present and prove our result.

[3]Specifically, bounded norm matrices with well-separated singular values. An $n \times n$ matrix has well-separated singular values if there exists some constant c such that for all $i \neq j$, $|\sigma_i - \sigma_j| \geq n^{-c}$.

2. Definitions

We first define approximation for real valued functions $f : \{0,1\}^\star \to \mathbb{R}$. We have two notions of approximation: additive and multiplicative. We say y additively approximates $f(x)$ with accuracy δ if $y \in [f(x) \pm \delta]$. We say y multiplicatively approximates $f(x)$ with accuracy δ if $y \in [(1 \pm \delta)f(x)]$.

We assume (as is customary in previous works) that the approximation algorithm M outputs a dyadic rational number, i.e., we interpret the string $M(x)$ as the rational number whose binary representation is the binary string $M(x)$. This, in particular, implies that if M has time complexity $t(n)$ then M can only output integers whose absolute value is at most $2^{t(n)}$, and this assumption is implicitly used in previous works. This assumption also implies that if $M(x)$ is a non-zero rational number then $|M(x) - 0| \geq 2^{-t(n)}$.

We say a function f is $R(n)$–bounded if $|f(x)| \leq R(n)$ for every $x \in \{0,1\}^n$ and some known uniform function R. Since we are only interested in functions that can be approximated by polynomial time algorithms we can assume w.l.o.g. that $R = 2^{\mathrm{poly}(n)}$, because no polynomial time algorithm can approximate a higher value. However, for specific functions f sometimes a better bound can be taken.

We begin with the time bounded model where it is customary to use *multiplicative* approximation:

Definition 2. A fully polynomial randomized approximation scheme (FPRAS) for an $R(n)$–bounded function $f : \{0,1\}^n \to [0, R(n)]$ is a randomized Turing machine M that on input x, an accuracy parameter δ and a confidence parameter ε, runs in $\mathrm{poly}(|x|, \delta^{-1}, \log \varepsilon^{-1}, \log R(n))$ time and outputs $M(x) \in [(1 \pm \delta)f(x)]$ with probability at least $1 - \varepsilon$, where we interpret the string $M(x)$ as the dyadic rational number whose binary representation is given by the binary string $M(x)$.

A fully polynomial (deterministic) approximation scheme (FPAS) is obtained if M in the above definition is deterministic, ε is set to 0, and the dependence on ε is removed.

In this paper we define log-space approximation schemes, but use *additive approximation* rather than multiplicative approximation, mainly because the major examples we have for such approximation schemes only achieve additive accuracy. We define:

Definition 3. A fully logarithmic randomized (resp. quantum) approximation scheme for an $R(n)$–bounded function $f : \{0,1\}^n \to [-R(n), R(n)]$ is a randomized (resp. quantum) Turing machine M that on input x, an error parameter δ and a confidence parameter ε, runs in $\mathrm{poly}(|x|, \delta^{-1}, \log \varepsilon^{-1}, \log R(n))$ time, uses $O(\log |x| + \log \delta^{-1} + \log\log \varepsilon^{-1} + \log\log R(n))$ space and outputs $M(x) \in [f(x) \pm \delta]$ with probability at least $1 - \varepsilon$.

The quantum space model we use has classical control and allows intermediate measurements, see [10, 1]. A fully logarithmic (deterministic) approximation scheme is obtained if M in the above definition is deterministic, ε is set to 0, and the dependence on ε is removed. We let FLAS abbreviate "fully logarithmic approximation scheme" and FLRAS (resp. FLQAS) the randomized (resp. quantum) versions.

If a function f computes a matrix, we say a Turing machine M approximates f if it approximates each entry of the matrix. With additive approximation this is equivalent to approximating the matrix in the norm $\|A\|_{\max} = \max_{i,j} |A_{i,j}|$. Notice that if $\tilde{A}$ approximates an $n \times n$ matrix A in the above norm for arbitrary polynomially small δ, then it also approximates A well in the spectral norm because $\|A - \tilde{A}\|_2 \le n^2 \|A - \tilde{A}\|_{\max} \le n^2 \delta$.

Under this notation the result in [1] shows that the function computing the SVD of a real matrix A that has a polynomially bounded norm and well-separated singular values and the function inverting a well-conditioned matrix with a non-negligible norm both have an FLQAS. We remark that the function computing the SVD of a matrix $A = UDV$ with polynomially bounded norm is polynomially bounded. This is because every singular value $D[i,i] = \sigma_i$ satisfies $\sigma_i \le \|A\|_2$ and the entries of the unitary matrices U, V are obviously bounded. The same holds for the function computing the inverse of a well-conditioned matrix with a non-negligible norm, as the entries of A^{-1} are bounded by $\|A^{-1}\|_2 = \kappa(A)/\|A\|_2$ (where $\kappa(A)$ is the condition number of A).

3. Derandomizing randomized and quantum space-bounded approximation schemes

We first reprove Goldreich's result. We start with an intuitive explanation.

Suppose $\mathsf{prBPP} = \mathsf{P}$ and f has an FPRAS. By definition, there exists a polynomial time probabilistic algorithm M that on input x and accuracy parameter δ outputs a value $M(x,\delta)$ that with a good probability is multiplicatively δ–close to $f(x)$. Define a promise problem Π such that $\langle x, \zeta, y\rangle$ is a yes instance if $f(x) > (1+\zeta)y$ and a no instance if $f(x) \le (1-\zeta)y$. We see that $\Pi \in \mathsf{prBPP}$ (because M essentially solves it) and by our assumption that $\mathsf{prBPP} = \mathsf{P}$ we have $\Pi \in \mathsf{P}$.

Having that we can employ a binary search to approximate $f(x)$. We start with the a-priori known lower and upper bounds $0 \le f(x) \le R(n) = 2^{\mathrm{poly}(n)}$ that hold for the output of any polynomial time algorithm. We then determine whether we are in the lower or upper half of this interval by calling Π. More precisely, if we know $f(x)$ belongs to the interval $[l, h]$, then with one call to the promise problem we can reduce the interval to a new one of length roughly $(\frac{1}{2} + \delta)(h - l)$. Repeating the binary search $O(\log R(n)/\delta)$ times we deterministically approximate $f(x)$ with δ multiplicative accuracy.

We now give the formal details:

Theorem 4. *Suppose* $\mathsf{prBPP} = \mathsf{P}$. *Then every function f that has an FPRAS also has an FPAS.*

Proof. Assume f has an FPRAS. Let M be a polynomial time probabilistic algorithm that on input $x \in \{0,1\}^\star$ and an accuracy parameter δ given in unary with $\frac{1}{\delta}$ bits, outputs a value $M(x,\delta)$ that with probability at least $2/3$ is multiplicatively δ–close to $f(x)$. We show how to construct a deterministic algorithm M' that on input $x \in \{0,1\}^\star$ and an accuracy parameter δ given in unary with $\frac{1}{\delta}$ bits, outputs a value $M'(x,\delta)$ that is multiplicatively δ–close to $f(x)$.

We construct the *output* $M'(x,\zeta)$ bit by bit, where each bit is determined by a single call to a prBPP promise problem. Consider the following promise problem Π:

Yes instance: $\langle x, \zeta, y\rangle$ such that $f(x) > (1+\zeta)y$.

No instance: $\langle x, \zeta, y\rangle$ such that $f(x) \le (1-\zeta)y$.

Π is in prBPP, by the following algorithm: Run M with accuracy $\zeta' = \zeta/2$ and accept if and only if $M(x,\zeta') \ge y$. For every $\langle x, \zeta, y\rangle \in \Pi_{\mathrm{Yes}}$, with probability at least $2/3$,

$$M(x,\zeta') \ge (1-\zeta') f(x) > (1-\zeta')(1+\zeta) y \ge y.$$

Similarly, for every $\langle x, \zeta, y\rangle \in \Pi_{\mathrm{No}}$, $M(x,\zeta') < y$ with probability at least $2/3$. As we assume

$\mathsf{prBPP} = \mathsf{P}$, there exists a deterministic algorithm for Π that we denote M_Π.

Let T denote the running time of M on the input $\langle x, \delta \rangle$. Set $l_0 = 0$, $h_0 = 2^T$ and $z_0 = \frac{l_0+h_0}{2}$. Notice that $l_0 \leq f(x) \leq h_0$. We now run the following binary search for $f(x)$:

For $i = 0, \ldots, \infty$ do:

- If $h_i < 2^{-T}$ output 0 and halt.
- If $\frac{h_i - l_i}{h_i + l_i} \leq \frac{\delta}{2}$ output z_i and halt.
- Query M_Π on $\langle x, \delta/4, z_i \rangle$. If the query is answered positively, set $l_{i+1} = (1 - \delta/4)z_i$ and $h_{i+1} = h_i$. Otherwise, set $h_{i+1} = (1 + \delta/4)z_i$ and $l_{i+1} = l_i$.
- Set $z_{i+1} = (l_{i+1} + h_{i+1})/2$.

First notice that because M_Π deterministically solves Π, we always preserve the invariance $l_i \leq f(x) \leq h_i$.

We now claim that if we output a value, then this value is a good multiplicative approximation to $f(x)$. To see that notice that we halt if either $h < 2^{-T}$ or $\frac{h-l}{h+l} \leq \frac{\delta}{2}$. In the first case $f(x) \leq h < 2^{-T}$. However, if $f(x)$ is non-zero then $f(x) \geq 2^{-T}$ (see Section 2). Hence $f(x) = 0$ and we output the correct value. If the latter condition happens then $|z - f(x)| \leq \delta f(x)$. To see that notice that $f(x) - z \leq h - z = h - \frac{l+h}{2} = \frac{h-l}{2} = \frac{h-l}{h+l} z \leq \frac{\delta}{2} z$. Similarly, $f(x) - z \geq -\frac{\delta}{2} z$. Thus $|f(x) - z| \leq \frac{\delta}{2} z$ which implies $z \in [(1 \pm \delta) f(x)]$.

To show that we always halt and to bound the number of iterations, let $d_i = h_i - l_i$. We claim that $d_{i+1} \leq \frac{3}{4} d_i$. This is true as $d_{i+1} \leq \frac{d_i}{2} + \frac{\delta}{4} z_i$ (immediately from the way the procedure works) and $\frac{d_i}{2} + \frac{\delta}{4} z_i \leq \frac{3}{4} d_i$ (which is true whenever $\frac{h_i - l_i}{h_i + l_i} > \frac{\delta}{2}$). Also, $\frac{h_i - l_i}{h_i + l_i} \leq 2^T d_i$ (because $h_i \geq 2^{-T}$). Hence, $\frac{h_i - l_i}{h_i + l_i} \leq 2^{2T} (\frac{3}{4})^i$ and we must stop within $O(T + \log(1/\delta))$ steps.

Altogether, the running time is $O(T + \log(1/\delta))$ times the time complexity of M_Π, which is polynomial in T as required. □

A similar procedure works for additive error by appropriately changing the promise problem Π to work with additive accuracy. We give a formal proof of this in Theorem 5 for the space bounded model, but a similar argument also works for the time bounded model.

Next, we claim that the same proof works also in the space bounded setting as long as the function f is polynomially bounded, i.e., $f : \{0,1\}^n \to [-R(n), R(n)]$ for $R(n) = \text{poly}(n)$. The reason is simple: the space complexity of the deterministic machine M' constructed in the proof of Theorem 4 is $O(\log(\frac{nR(n)}{\delta}))$. Formally,

Theorem 5. *Suppose* $\mathsf{prBPL} = \mathsf{L}$. *Then every* $R = \text{poly}(n)$*–bounded function* f *that has an FLRAS has an FLAS as well.*

Proof. The proof resembles the proof of Theorem 4. However, as the approximation is now additive, the promise problem Π is given by:

Yes instance: $\langle x, \zeta, y \rangle$ such that $f(x) \geq y + \zeta$.

No instance: $\langle x, \zeta, y \rangle$ such that $f(x) \leq y - \zeta$.

Π is in prBPL, by exactly the same algorithm. As $\mathsf{prBPL} = \mathsf{L}$, let M_Π be the deterministic algorithm for Π. We then run a similar loop, starting with $l \leftarrow -R(n)$ and $h \leftarrow R(n)$ and iterating as long as $|h - l| > 2\delta$ while the update for l and h is done additively.

The correctness follows from very similar reasonings as $d_{i+1} \leq \frac{d_i}{2} + \frac{\delta}{2} \leq \frac{3}{4} d_i$ in every iteration, hence the loop terminates after $\Theta(\log \delta^{-1} + \log R(n))$ iterations.

The time complexity of the FLAS is polynomial in $R(n)$ and δ^{-1}, as desired. The space required to store l, h and z, in addition to the space required to simulate M_Π is bounded by $O(\log n + \log \delta^{-1} + \log R(n))$. As $\log R(n) = O(\log n)$, we also obtain the required space constraint for an FLAS. □

The proof of Theorem 5 also generalizes to the quantum case, namely,

Theorem 6. *Suppose* $\mathsf{prBQL} = \mathsf{prBPL}$. *Then every* $R = \text{poly}(n)$*–bounded function* f *that has an FLQAS has an FLRAS as well.*

Proof. Let Π be the same promise problem defined in the proof of Theorem 5. Then, by the same reasoning as in the proof of Theorem 5, $\Pi \in \mathsf{prBQL}$. Thus, if $\mathsf{prBQL} = \mathsf{prBPL}$ then $\Pi \in \mathsf{prBPL}$. We can amplify the success probability of the probabilistic algorithm so that it succeeds with probability at least $1 - \xi$ for ξ that we choose later.

Run the same binary search algorithm with the randomized algorithm for Π, and assume it uses T iterations. A similar argument to the one in the proof of Theorem 5 shows that with probability $1 - T\xi$ the algorithm outputs x to within an additive factor δ. By setting $\xi = \frac{\varepsilon}{T}$, the FLRAS with accuracy parameter δ and confidence parameter ε

runs in time $T \cdot \mathrm{poly}(n, \delta^{-1}, \log \xi^{-1}, \log R(n)) = \mathrm{poly}(n, \delta^{-1}, \log \varepsilon^{-1})$ and uses $O(\log R(n) + \log n + \log \delta^{-1} + \log\log \xi^{-1}) = O(\log n + \log \delta^{-1} + \log\log \varepsilon^{-1} + \log\log R(n))$ space, as desired. □

As explained before, [1] shows that the function computing the SVD of a real matrix A that has a polynomially bounded norm and well-separated singular values, and the function inverting a well-conditioned matrix with a non-negligible norm both have an FLQAS, and both compute a polynomially bounded function. Thus, either $\mathsf{prBQL} \neq \mathsf{prBPL}$, or else there must exist an FLRAS for both problems, implying that approximately inverting a matrix is essentially in prBPL.

References

[1] A. Ta-Shma, Inverting well conditioned matrices in quantum logspace, in: Proceedings of the 45th annual ACM symposium on Symposium on theory of computing, STOC '13, ACM, New York, NY, USA, 2013, pp. 881–890. doi:10.1145/2488608.2488720.

[2] N. Halko, P.-G. Martinsson, J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM review 53 (2) (2011) 217–288.

[3] D. A. Spielman, S.-H. Teng, Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems, arXiv preprint cs/0607105.

[4] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Physical review letters 103 (15) (2009) 150502.

[5] L. Stockmeyer, On approximation algorithms for #P, SIAM Journal on Computing 14 (4) (1985) 849–861.

[6] R. Shaltiel, C. Umans, Pseudorandomness for approximate counting and sampling, computational complexity 15 (4) (2006) 298–341.

[7] M. Jerrum, A. Sinclair, E. Vigoda, A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries, J. ACM 51 (4) (2004) 671–697.

[8] O. Goldreich, In a world of P=BPP, in: Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation, Springer, 2011, pp. 191–232.

[9] S. Aaronson, The complexity zoo, http://cse.unl.edu/~cbourke/latex/ComplexityZoo.pdf (2014).

[10] D. van Melkebeek, T. Watson, Time-space efficient simulations of quantum computations, Electronic Colloquium on Computational Complexity (ECCC) 17 (2010) 147.