Worst-case hardness suffices for derandomization: a new method for hardness-randomness trade-offs

Alexander E. Andreev
University of Moscow

Andrea E. F. Clementi*
University of Rome

José D. P. Rolim
University of Geneva

(Extended Abstract)

Abstract

Up to know, the known derandomization methods have been derived assuming average-case hardness conditions. In this paper we instead present the first worst-case hardness conditions sufficient to obtain $P = BPP$.

Our conditions refer to the worst-case circuit complexity of Boolean operators computable in time exponential in the input size. Such results are achieved by a new method that departs significantly from the usual known methods based on pseudo-random generators. Indeed, we prove new particular bounds on the circuit complexity of partial Boolean functions which are then used to derive efficient constructions of *hitting set generators*. As recently proved, such generators can derandomize any BPP-algorithm.

Our method also gives a worst-case hardness condition for the circuit complexity of Boolean operators computable in NC (with respect to their output size) to obtain an efficient reduction from any two-side error NC algorithm into a zero-error NC algorithm thus obtaining $ZNC = BPNC$.

*Contact author: Dip. di Scienze dell'Informazione, Universitá di Roma "La Sapienza", Via Salaria 113, 00198 Roma. E-mail: `clementi@dsi.uniroma1.it`

1 Introduction

1.1 Motivations and previous results. A major goal in complexity theory is the study of the real power of randomized algorithms, that is algorithms that make decisions based on the output of a random source of bits. To this aim, several recent works have been focused on the design of general methods that decrease (or remove) the amount of random bits used by these algorithms. A central question in this area is the relationship between the existence of computationally-hard functions and the existence of efficient derandomization methods. Yao [19], and Blum and Micali [8] introduced the concept of *Pseudo-Random Generator* (PSRG), any Boolean operator $G = \{G_n : \{0,1\}^{k(n)} \to \{0,1\}^n, n > 0\}$, (denoted by $G : k(n) \to n$) that, for a.e. n and for any Boolean function $f : \{0,1\}^n \to \{0,1\}$ whose circuit complexity $L(f)$ is at most n, satisfies: $|\mathbf{Pr}(f(\vec{y}) = 1) - \mathbf{Pr}(f(G_n(\vec{x})) = 1)| \leq 1/n$ (where $\vec{y}$ is chosen uniformly at random from $\{0,1\}^n$, and $\vec{x}$ from $\{0,1\}^{k(n)}$). The output sets of PSRG's are also called *discrepancy* sets for circuits of linear size.

According to the definition used in [14], a Boolean operator $Op : k(n) \to n$ is *quick* if it can be computed in time polynomial in n (note in passing that if $k(n) = O(\log n)$ then the "quick" condition is equivalent to assume that Op belongs to EXP). It is not hard to show [14] that the existence of a quick PSRG $G : k(n) \to n$ with $k(n) = O(\log n)$ implies $P = BPP$. Nisan and Wigderson [14] showed a method to construct *quick* PSRG's based on the existence of Boolean functions in EXP that have exponential *hardness* [14]. The hardness condition used by Nisan and Wigderson requires the existence of a function in EXP that not only has a hard *worst-case* circuit complexity[1] but also a hard *average-case* circuit complexity. More formally, a function $f : \{0,1\}^n \to \{0,1\}$ is (ϵ, L)-*hard* if, for any circuit C of size at most L, $|\mathbf{Pr}(C(\vec{x}) = f(\vec{x})) - 1/2| \leq \epsilon/2$. Given a Boolean function $F = \{F_n : \{0,1\}^n \to \{0,1\},\ n > 0\}$, the *hardness* at n of F (denoted as $H_F(n)$) is defined as the maximum integer h_n such that F_n is $(1/h_n, h_n)$-hard. Then, F has exponential hardness if $H_F(n) \geq 2^{\Omega(n)}$. Nisan and Wigderson showed a fundamental "Hardness vs Randomness" result.

Theorem 1.1 *[14] If a Boolean function F exists such that* i) *$F \in EXP$, and* ii) *F has exponential* hardness*, then there exists a quick PSRG $G : k(n) \to n$ where $k(n) = O(\log n)$, and consequently $P = BPP$.*

The hardness required by Nisan and Wigderson's construction of quick PSRG's thus refers to average-case complexity. Then a consequent and natural question is the following: Does any "worst-case" hardness assumption on the circuit complexity of Boolean functions computable in time exponential in the input size exist which allows to derive an efficient derandomization method (in particular, to obtain $P = BPP$)? A good point to motivate this question is to consider the novelty and the potentiality of any eventual precise relationship between one of the most studied problems in complexity theory, that is finding lower bounds for the worst-case circuit complexity of classes of recursive Boolean functions, and that of investigating the real computational power of randomness.

We give two answers to this question. Both answers make use of a new method (informally described in Section 1.3) that relies on a particular class of Boolean operators (different from PSRG's), denoted as *Hitting Set Generators*, which have been recently introduced in [5]. Let $L(f)$ denote the circuit complexity of a finite function $f : \{0,1\}^n \to \{0,1\}$ and, given any positive number dp, the term $L_{dp}(f)$ denotes the minimum size of circuits of depth dp which are able to compute f.

[1]As circuit complexity of a finite Boolean function f, we will always mean the size of the smallest circuit that computes f.

Definition 1.1 *Let $\epsilon(n)$, $\beta(n)$, and $\gamma(n)$ be polynomial-time computable functions such that for any $n \geq 1$: $0 < \epsilon(n) < 1$, $n \leq \beta(n) \leq 2^n$, and $\gamma(n) \geq \log n$. Then, a Boolean operator $H : k(n) \to n$ is an $(\epsilon(n), \beta(n), \gamma(n))$-*Hitting Set Generator *(in short, $(\epsilon(n), \beta(n), \gamma(n))$-HSG) if, for any Boolean function f such that $L_{\gamma(n)}(f) \leq \beta(n)$ and $\mathbf{Pr}\,(f = 1) \geq \epsilon(n)$, H is required to provide one "example" $\vec{y}$ for which $f(\vec{y}) = 1$, i.e., there exists $\vec{a} \in \{0,1\}^{k(n)}$ such that $f(H_n(\vec{a})) = 1$. When no depth constraint $\gamma(n)$ is imposed, we will use notation $(\epsilon(n), \beta(n))$-HSG.*

By making a simple comparison between the definition of discrepancy sets and that of hitting sets it should be clear that HSG's satisfy a property significantly weaker than that of PSRG's (see [5] for more discussions on the differences between PSRG's and HSG's). Nevertheless, Andreev *et al* [5] proved that, given any BPP-algorithm A, the output of any quick HSG can be transformed into an *ad hoc* discrepancy set for A by means of a deterministic polynomial-time algorithm.

Theorem 1.2 *[5] Let $k(n) = O(\log n)$ and let ϵ be any constant such that $0 < \epsilon < 1$. If there exists a quick (ϵ, n)-HSG $H : k(n) \to n$ then $P = BPP$.*

1.2 Our results. We give two worst-case hardness conditions which are sufficient to construct quick HSG's that satisfy Theorem 1.2 thus obtaining $P = BPP$. The circuit complexity of a Boolean operator H will be denoted as $L^{op}(H)$. Observe that if $L^{op}(k, n)$ denotes the worst-case circuit complexity of Boolean operators $H : k(n) \to n$, then it is known [11, 18] that, for any $\log n \leq k \leq n$, $L^{op}(k, n) = (1 + o(1))(2^k n)/(k + \log n)$. Furthermore, for a.e. Boolean operator $H : k \to n$, we have $L^{op}(H) = \Theta((2^k n)/(k + \log n))$. The first condition deals with the worst-case circuit-complexity of characteristic functions of sets generated by Boolean operators.

Theorem 1.3 *Let δ be such that $0 < \delta \leq 1/2$, and let $k(n) = (1 + \Theta(1)) \log n$. If there exists a quick operator $H : k(n) \to n$ such that the characteristic function of its output sets*

$$F^H = \{F_n^H : \{0,1\}^n \to \{0,1\} \ , \ \textit{where} \ \ F_n^H(\vec{x}) = 1 \ \ \textit{if} \ \exists\, \vec{y} \in \{0,1\}^{k(n)} \textit{s.t.}\ H_n(\vec{y}) = \vec{x},\ n > 0\}$$

satisfies

$$L(F_n^H) \geq (1/2 + \delta)(2^{k(n)} n)/(k(n) + \log n),$$

then it is possible to construct a quick operator $H' : k'(n) \to n$ where $k'(n) = \Theta(\log n)$ such that H' is an (ϵ, n)-HSG for some constant $0 < \epsilon < 1$, thus obtaining $P = BPP$.

Another way to state the above theorem is the following. Assume that there exists a sparse language $S = \{S_n \subseteq \{0,1\}^n, n > 0\}$ that can be generated by an uniform algorithm which runs in time polynomial in n, and such that the worst-case circuit complexity of deciding S is not smaller (up to some constant factor) than the worst-case circuit complexity of generating languages S' having the same sparsity factor of S. Then $P = BPP$.

The second sufficient condition to obtain a quick HSG refers directly to the worst-case circuit complexity of Boolean operators instead of the characteristic functions of their output sets.

Theorem 1.4 *Let $k(n) = \Theta(\log n)$. Let $H : k(n) \to n$ be a quick operator such that for a.e. n,*

$$L^{op}(H_n) \geq L^{op}(k, n) - (2^{k(n)})/(k(n)^2).$$

Then, for any constant $0 < \epsilon < 1$, and for any positive integer q, it is possible to construct a quick $(1 - \epsilon, n^q)$-HSG $H' : k'(n) \to n$, where $k'(n) = \Theta(\log n)$, thus obtaining $P = BPP$.

Unfortunately, Andreev *et al*'s algorithm which gives Theorem 1.2 seems to be very hard to parallelize [5]. It turns out that we are not able to obtain worst-case hardness conditions for Boolean operators sufficient to derandomize any BPNC algorithm (i.e. to obtain $BPNC = NC$). However we show that, under hardness worst-case conditions comparable to that in Theorem 1.3, it is possible to construct an HSG for the class of Boolean circuits having linear size and polylogarithmic depth that can be used to define an "oracle" machine for approximating the fraction of the accepting computations of any BPNC algorithm. This oracle machine has the following property: either it returns a "failure" answer (this happens with small probability) or it gives a correct value. Hence, if we consider the three probabilistic parallel complexity classes, ZPNC (i.e. *zero error probability* NC), RNC (i.e. *one side error probability* NC), and BPNC (*two side bounded error probability* NC), we obtain an interesting collapsing result[2].

Theorem 1.5 *A constant* $0 < c_0 < 1$ *exists such that if an operator* $H : k(n) \to n$ *with* $k(n) = O(\log n)$ *exists such that 1)* H *is an NC operator*[3], *and 2) for any* $d \geq 1$ *there exists a constant* c *with* $0 < c_0 \leq c < 1$ *such that the characteristic function* F^H *of its output sets satisfies* $L_{\log^d n}(F^H_n) \geq c(2^{k(n)}n)/(k(n) + \log n)$*, then* $ZNC = BPNC$*.*

1.3 The interest in our method and further connections with other works. All of our proofs share a common method based on the following fact. There is a precise trade-off between the worst-case circuit complexity of partial Boolean functions and the number of 1's in their outputs. In particular, we formalize the intuitive fact that a partial Boolean function having a hard worst-case circuit complexity cannot return 0 for a "large" number of inputs. This property is used to construct the preliminary versions of our HSG's which are then combined with a convenient use of the properties of expanders graphs [3] (to obtain Theorem 1.3) and with a new analysis of the performances of the already mentioned Andreev *et al*'s algorithm [5] (to obtain Theorem 1.4). A similar framework has been developed to derive the collapsing result in Theorem 1.5.

It is interesting to observe that our method could yield weaker hardness conditions sufficient to construct efficient HSG's provided that a better and/or more general analysis on the above trade-offs will be individuated. In particular, the hardness conditions in Theorems 1.3 and 1.4 could be made weaker if new asymptotical relationships will be found between three main aspects of the complexity of a generic subset from $\{0,1\}^n$: the size (i.e. the sparsity factor) of the subset, the worst-case circuit complexity of its characteristic function, and the worst-case circuit complexity of Boolean operators that are able to *cover* the subset. In fact, hitting sets are not required to be generated but just covered since any set that contains a hitting set is still a hitting set (observe that this monotone property does not hold for discrepancy sets). Concerning *Covering* generators for Boolean subsets, new promising trade-offs results have been recently proved in [7].

Finally, we emphasize that the combination of another circuit-complexity bound and the properties of HSG's represents also the key ingredient in the proof of the polynomial-time simulation of BPP-algorithms using *very weak* random sources (see for example [16, 17]), a new result obtained in [6]. Again this witnesses the potentiality of the technique introduced in our paper.

Due to the lack of space, proofs will be only sketched and some of them will be given in the Appendix.

[2]Observe that $ZNC \subseteq RNC \subseteq BPNC$.

[3]With *"NC operator"*, we will always mean an operator which is computable in NC with respect to the size of its output

2 Preliminary results on the circuit complexity of partial Boolean functions

In this section we show some new precise bound on the Shannon function describing the trade-offs between the worst-case circuit complexity of partial Boolean functions and the number of inputs on which they output 1.

Let $\mathcal{F}(n, N, m)$ be the set of all partial Boolean functions $f(x_1, \ldots, x_n)$ defined on $N \leq 2^n$ inputs and assuming 1 on $m \leq N$ inputs. Furthermore, $L(n, N, m)$ denotes the worst-case circuit complexity of functions from $\mathcal{F}(n, N, m)$, and $L_{depth}(n, N, m)$ denotes the maximum value $L_{depth}(f)$ among all functions f from $\mathcal{F}(n, N, m)$. Lupanov proved the following result for the case of total Boolean functions.

Theorem 2.1 *[11] Let $L^{tot}(n, m) = L(n, 2^n, m)$. Then*

$$L^{tot}(n,m) \;=\; (1+o(1))\left(\log\binom{2^n}{m}\right) / \left(\log\log\binom{2^n}{m}\right) .$$

The following corollary is implicit in Lupanov's proof.

Corollary 2.1 *A constant $c > 0$ exists such that*

$$L^{tot}_{c\log(mn)}(n,m) \;=\; (1+o(1))\left(\log\binom{2^n}{m}\right) / \left(\log\log\binom{2^n}{m}\right) .$$

However, in order to construct quick HSG's we need that Lupanov's results hold also for partial Boolean functions. In particular, the generalization of the upper bounds implicitly given in Theorem 2.1 and in Corollary 2.1 cannot be derived directly from the proofs in [11]. Then we give a reduction from general Boolean functions to the restricted case of total Boolean functions which is based on a probabilistic construction of suitable linear operators (the reduction is described in the Appendix).

Theorem 2.2

$$L(n,N,m) \;=\; (1+o(1))\left(\log\binom{N}{m}\right)\left(\log\log\binom{N}{m}\right) + O(n) .$$

Furthermore a constant $c > 0$ exists such that

$$L_{c\log n}(n,N,m) \;=\; (1+o(1))\left(\log\binom{N}{m}\right)\left(\log\log\binom{N}{m}\right) + O(n) .$$

3 Hard characteristic functions and HSG's

The following theorem provides a first trade-offs between the hardness of characteristic functions of Boolean subsets and their hitting properties[4].

[4]Each result will be given in both sequential and "parallel" version. The latter will be included in square brackets.

Theorem 3.1 *Let $0 < c_2 < 1$ be a constant [and $d \geq 1$], and let $S_n \subseteq \{0,1\}^n$ be any subset such that $|S_n| \leq b_n$, where $b_n = n^{\Theta(1)}$. Suppose that for the characteristic function F_n of S_n we have*

$$i)\ L(F_n) \ \geq\ c_2 \frac{b_n n}{\log b_n + \log n} \quad [\ i')\ L_{\log^{d+1} n}(F_n) \ \geq\ c_2 \frac{b_n n}{\log b_n + \log n}\ ] \ .$$

Then, for any constant c_1, such that $0 < c_1 < c_2$, for any Boolean function $f(x_1, \ldots, x_n)$ such that

$$ii)\ \mathbf{Pr}(f=1) \geq 1 - 2^{(c_1-1)n}, \quad and \quad iii)\ L(f) \leq b_n \quad [\ iii')\ L_{\log^d n}(f) \leq b_n\ ],$$

there exists $\vec{a} \in S_n$ for which $f(\vec{a}) = 1$.

Sketch of the proof. Suppose, by contradiction, that f satisfies conditions $ii)$ and $iii)$ but for any $\vec{a} \in S_n$ we have $f(\vec{a}) = 0$. Let $Z \subseteq \{0,1\}^n$ be the subset of all inputs on which $f = 0$. Clearly, we have $S_n \subseteq Z \subseteq \{0,1\}^n$. Then consider the partial Boolean function $g(x_1, \ldots, x_n)$ defined as follows: $g(\vec{a}) = 1$ if $\vec{a} \in S_n$, $g(\vec{a}) = 0$ if $\vec{a} \in Z \setminus S_n$, and $g(\vec{a})$ is not defined if $\vec{a} \in \{0,1\}^n \setminus Z$. Since $|Z| \ \leq\ 2^{c_1 n}$ and $|S_n| \ \leq\ b_n$, from Theorem 2.2, we have

$$L(g) \ \leq\ (1+o(1)) \left(\log \binom{2^{c_1 n}}{b_n} \right) / \left(\log\log \binom{2^{c_1 n}}{b_n} \right) + O(n) \leq (1+o(1)) c_1 (b_n n)/(\log b_n + \log n) \ .$$

From $S_n \subseteq Z$, it is easy to prove that, given any $\vec{a}$, $F_n(\vec{a})$ can be computed[5] as $g(\vec{a}) \wedge \neg f(\vec{a})$. Hence

$$L(F_n) \leq L(g) + L(f) + O(1) \leq (1+o(1)) c_1 \frac{b_n n}{\log b_n + \log n} + b_n + O(1) \leq (1+o(1)) c_1 \frac{b_n n}{\log b_n + \log n}.$$

For sufficiently large n, this last upper bound is in contradiction with hypothesis (i) of our theorem. The "parallel" version of the theorem can be easily derived using the same contradiction argument. □

In which follows, we will consider HSG's which always have a monotone function prize $k(n)$ such that, for any $n > 0$, $k(n+1) - k(n) \leq 1$ and $n^\alpha \geq k(n) \geq \log n$ where $0 < \alpha < 1$. Let $H : k(n) \to n$ be a Boolean operator with $k(n) = \Theta(\log n)$, and let $F^H = \{F_n^H \ :\ \{0,1\}^n \to \{0,1\},\ n > 0\}$ be the corresponding family of the characteristic functions.

Corollary 3.1 *Suppose that a quick [NC] operator $H : k(n) \to n$ exists such that $k(n) = (1 + \Theta(1)) \log n$ and a constant $0 < c_2 < 1$ exists such that, for a.e. n, $L(F_n^H) \geq c_2(2^{k(n)} n)/(k(n) + \log n)$ [$L_{\log^{d+1} n}(F_n^H) \geq c_2(2^{k(n)} n)/(k(n) + \log n)$ for some $d \geq 1$]. Then, for any positive constant q and for any constant c_1 such that $0 < c_1 < c_2$, it is possible to construct a quick [NC] operator $H' : k'(n) \to n$ with $k'(n) = \Theta(\log n)$ and such that H' is an $(1 - 2^{(c_1-1)n}, n^q)$-HSG [H' is an $(1 - 2^{(c_1-1)n}, n^q, \log^d n)$-HSG].*

Sketch of the proof. Since $k(n) = (1 + \Theta(1)) \log n$, we can assume that $k(n) = (1 + \delta) \log n$ for some constant $\delta >$. Define $s(n) = \lceil \frac{2q}{\delta} \log n \rceil$. Given any $\vec{a} \in \{0,1\}^n$, $[\vec{a}]_{i_1,i_2}$ denotes the substring $\vec{a}_{i_1}, \ldots, \vec{a}_{i_2}$. We consider the new quick operator $H' : k'(n) \to n$ where $k'(n) = k(2^{s(n)} n) + s(n)$ defined as follows:

[5] Here we assume that $0 \wedge ? = 0$.

$$H'_n(\vec{a},\vec{b}) \; = \; [H_{n2^{s(n)}}(\vec{a})]_{t_1,t_2} \; , \quad \text{where } t_1 = n(\phi(\vec{b})-1) \; , \quad \text{and } t_2 = t_1 + n - 1$$

(observe that $\vec{a} \in \{0,1\}^{k(2^{(s(n)}n)}$, $\vec{b} \in \{0,1\}^{s(n)}$, and $\phi(\vec{b})$ is the decimal representation of $\vec{b}$). Consider a Boolean function $f(x_1,\ldots,x_n)$ such that $\mathbf{Pr}\,(f=1) \geq 1 - 2^{(c_1-1)n}$ and $L(f) \leq n^q$. Then for

$$f^*(x_1,\ldots,x_{2^{s(n)}n}) \; = \; \bigvee_{t=1}^{2^{s(n)}} f(x_{(t-1)n+1}, x_{(t-1)n+2}, \ldots, x_{tn}) \; ,$$

it is easy to prove that $\mathbf{Pr}\,(f^*=1) \geq 1 - 2^{(c_1-1)2^{s(n)}n}$, and

$$L(f^*) \leq 2^{s(n)} + 2^{s(n)}L(f) \leq (1+o(1))2^{s(n)}n^q \leq (1+o(1))2^{s(n)}2^{\frac{\delta}{2}s(n)} \; \leq (1+o(1))\left(2^{s(n)}n\right)^{1+\frac{\delta}{2}}.$$

Thus for sufficiently large n we have $L(f^*) \leq \left(2^{s(n)}n\right)^{1+\delta} \leq 2^{k(2^{s(n)}n)}$. By applying Theorem 3.1 with $b_n = 2^{k(2^{s(n)}n)}$, we have that there exist $\vec{a} \in \{0,1\}^{2^{s(n)}n}$ and $t \in [1,\ldots,2^{s(n)}]$ such that $f^*([H_{2^{s(n)}n}(\vec{a})]_{(t-1)n,tn}) = 1$. It follows that there exist $\vec{a} \in \{0,1\}^{k(2^{s(n)}n)}$ and $\vec{b} \in \{0,1\}^{s(n)}$ such that $f(H'_n(\vec{a},\vec{b})) = 1$.

$\square$

3.1 Improved HSG's using expanders

Corollary 3.1 gives a quick HSG for the class of polynomial size circuits (functions) C that have a very large fraction of 1's, i.e. $\mathbf{Pr}\,(C=1) \geq 1 - 2^{-cn}$ for some positive constant smaller than 1. However, this hitting property does not suffice to derandomize BPP-algorithms (see Theorem 1.2). It is in fact required to hit all linear-size circuits having "only" a constant fraction of $1's$. To this aim, we will combine the HSG in Corollary 3.1 with a *random walk* on *expanders*, a tool that has been often used in decreasing randomness in probabilistic algorithms.

An undirected graph $G(V,E)$ is a (d,c)-*expander* if the maximum degree of a vertex is d, and for every set $W \subseteq V$ of cardinality $|W| \leq |V|/2$, the inequality $|N(W) - W| \geq c|W|$ holds, where $N(W)$ denotes the set of all vertices adjacent to some vertex in W. The expanding properties of a graph can be established by determining the value of its second largest eigenvalue (see for example [3]). Indeed, if λ is an upper bound on the second largest eigenvalue of any d-regular graph $G(V,E)$, then G is a (d,c)-expander for $c = (d-\lambda)/2d$. Expander graphs have the following important "hitting" property proved by *Ajtai et al* [1] (for a proof of this fact see for example Theorem 2.7 - p.124 - of [3]).

Theorem 3.2 *Let $G(V,E)$ be a d-regular graph, and assume that its second largest eigenvalue is at most $\lambda > 0$. Given any subset $W \subseteq V$ such that $|W| = \alpha n$ ($\alpha < 1$). Then, for every $t > 0$, the number of walks of length t in G that avoid W is at most* $n(1-\alpha)^{1/2}((1-\alpha)d^2 + \lambda^2)^{t/2}$.

In [9], a polynomial-time algorithm is presented that, given $n > 0$, and $d \leq n$, constructs a d'-regular expanders G such that $d' = O(d)$, $|V| = O(n)$, and its second largest eigenvalues $\lambda > 0$ is such that $\lambda \leq 2\sqrt{d-1}$ (such graphs are called *Ramanujan* graphs).

For any $n > 0$, consider a d-regular *Ramanujan expander* $EP_n = (V_n, X_n)$ where $2^n < |V_n| \leq 2^{n+1}$ [9]. Observe that the Boolean strings with last component equal 0 correspond to the input set of the function we want to hit. This assumption is required when EP_n cannot be constructed on vertex sets whose size is exactly a power of 2. Let $l = \lceil \log d \rceil$. We suppose that d is a large but constant value. Then, we consider the operator $EPR_{n,t} : \{0,1\}^{n+l\cdot(2^t-1)+t} \to \{0,1\}^n$, such that

$$EPR_{n,t}(\vec{a}, \vec{u}_1, \ldots, \vec{u}_{2^t-1}, \vec{s}) \ , \qquad \vec{a} \in \{0,1\}^n \ , \ \vec{u}_i \in \{0,1\}^l \ , \ \vec{s} \in \{0,1\}^t \ ,$$

are the first n components of the $\phi(\vec{s})$-th vertex of the EP_n-walk of length 2^t which starts from vertex $(\vec{a}, 0)$ and is uniquely determined by the sequence of edge choices in the neighborhood of each vertex: $\phi(\vec{u}_1), \ldots, \phi(\vec{u}_{2^t-1})$. Observe that if $t = \Theta(\log n)$, the operator $EPR_{n,t}$ can be computed in time polynomial in n. Consider now a Boolean function $g(x_1, \ldots, x_n)$, and the operator $EPR^g_{n,t}$: $\{0,1\}^{n+l\cdot 2^t} \to \{0,1\}$ that performs the OR among the values of g computed on the input points visited by a fixed EP_n-walk of length 2^t, i.e.,

$$EPR^g_{n,t}(\vec{a}, \vec{u}_1, \ldots, \vec{u}_{2^t}) \ = \ \bigvee_{\vec{s} \in \{0,1\}^t} g\left(EPR_{n,t}(\vec{a}, \vec{u}_1, \ldots, \vec{u}_{2^t-1}, \vec{s})\right) \ . \qquad (1)$$

As consequence of Theorem 3.2, we can prove the following bound (for the proof see the Appendix).

Lemma 3.1 *If* $\mathbf{Pr}\left(g = 0\right) \leq c < \frac{1}{2}$, *then* $\mathbf{Pr}\left(EPR^g_{n,t} = 0\right) \leq \left(c + \frac{\lambda}{d}\right)^{2^t-2}$.

Theorem 3.3 *Assume that there exists a quick operator $H : k(n) \to n$, such that $k(n) = (1 + \Theta(1))\log n$ and the characteristic functions of its output sets satisfies*

$$L(F^H_n) \geq ((\log(4\lambda)/(\log d) + \delta)(2^{k(n)}n)/(k(n) + \log n)$$

for some constant $\delta > 0$. Then it is possible to construct a quick operator $H'' : k''(n) \to n$ with $k''(n) = \Theta(\log n)$ and such that H'' is an $(1-\epsilon, n)$-HSG for some constant $0 < \epsilon < 1$, thus $P = BPP$.

Sketch of the proof. By applying Corollary 3.1 with $c_2 = ((\log(4\lambda/\log d) + \delta)$ we have that, for any positive integer q, a quick operator $H' : k'(n) \to n$ exists such that $k'(n) = \Theta(\log n)$ and H' is a quick $(1 - 2^{(c_1-1)n}, n^q)$-HSG, where $c_1 = (\log(4\lambda)/\log d) + \delta/2 < c_2$. Let $l = \lceil \log d \rceil$ and $t(n) = \lceil 2\log n \rceil$. Then we define the new quick operator $H'' : k''(n) \to n$, such that

$$H''_n(\vec{a}, \vec{b}) = EPR_{n,t(n)}(H'_{n+l\cdot(2^{t(n)}-1)}(\vec{a}), \vec{b}) \ , \ \text{ where } \ k''(n) = n + l \cdot (2^{t(n)} - 1) + t(n) \ .$$

From the construction of Ramanujan expanders shown in [9], we can assume that $\frac{1}{2} \leq \lambda \leq 2\sqrt{d-1}$. Define $\epsilon = \lambda/(2d)$ (observe that for a sufficiently large constant d we can assume that $0 < \epsilon < 1$) and consider any Boolean function $g(x_1, \ldots, x_n)$ such that $\mathbf{Pr}\left(g = 1\right) \geq 1 - \epsilon$, and $L(g) \leq n$. Then, we define $f = EPR^g_{n,t(n)}$ where $N = n + l \cdot (2^{t(n)} - 1)$ (see Eq. 1). Clearly, f has polynomial-size circuit, thus we can choose q in the definition of H' such that $L(f) \ \leq \ N^q$. Lemma 3.1 implies that

$$\mathbf{Pr}\left(f = 0\right) = \mathbf{Pr}\left(EPR^g_{n,t(n)} = 0\right) \leq \left(\epsilon + \frac{\lambda}{d}\right)^{2^{t(n)}-2} \leq 2^{\frac{\log(2\lambda)-\log d}{\log d+1}(N-n-2)} \leq 2^{\left(\frac{\log(4\lambda)}{\log d}-1\right)N}$$

(observe that the last inequality holds for a.e. n). By definition, we know that H' hits function f, i.e., there exists $\vec{a} \in \{0,1\}^{k'(n+l\cdot(2^{t(n)}-1)}$ such that $f(H'_{n+l(2^{t(n)}-1)}(\vec{a})) = 1$. From the definition of f, there exists $\vec{b} \in \{0,1\}^{t(n)}$ such that $g(H''_n(\vec{a}, \vec{b})) = g(EPR_{n,t(n)}(\vec{a}, \vec{b})) = 1$.

□

4 Hitting Set Generators for BPNC

Ramanujan's graphs cannot be used to derive NC Hitting Set Generators since no efficient parallel method to perform random walks on such graphs is presently available. However, Zuckermann [20] recently introduced an NC construction of *samplers* [20] which can replace the role of expanders in our construction. In particular, we can use the following result.

Theorem 4.1 *[20] Any BPNC algorithm that uses n random bits and has error probability bounded by $1/3$ can be simulated by a BPNC algorithm that uses $r(n) = O(n)$ random bits and has error probability bounded by $(1/2)^n$.*

Informally speaking, this result allows us to consider only "parallel" circuits having a fraction of 1's not smaller than $1 - 2^{-cn}$ for some fixed constant $0 < c < 1$. By using the same method of Section 3.1, we can combine Corollary 3.1 and Theorem 4.1 to obtain the following result

Theorem 4.2 *A constant $0 \le c_z < 1$ exists such that the following holds. Assume that there exists an NC operator $H : k(n) \to n$ with $k(n) = (1 + \Theta(1)) \log n$ and such that, for any constant $d \ge 1$, the characteristic functions of its output sets satisfy $L_{\log^{d+1} n}(F_n^H) \ge \delta(2^{k(n)} n/(k(n) + \log n)$, for some constant $\delta \ge c_z$. Then it is possible to construct an NC operator $H' : k'(n) \to n$ with $k'(n) = \Theta(\log n)$ and such that H' is an $(1 - \epsilon, n, \log^d n)$-HSG for any constant $0 < \epsilon < 1$ and $d \ge 1$.*

Corollary 4.1 *A constant $0 < c_z < 1$ exists such that if an NC operator $H : k(n) \to n$ exists that satisfies the same conditions of Theorem 4.2 then ZNC = BPNC.*

Sketch of the proof. A generic computation of a BPNC-algorithm on a fixed input of size m can always be seen as a Boolean circuit $f : \{0,1\}^n \to \{0,1\}$ (where $n = n(m)$ is some polynomial in m) with complexity $L_{log^d n}(f) \le n$ for some fixed $d \ge 1$; n is the number of random bits used by the algorithm for inputs of length m. The thesis will be proved by showing a ZNC algorithm that gives with high probability a good approximation of the value $E = \mathbf{Pr}(f = 1)$; Consider a random table of inputs for f, i.e., $T = (\vec{a}_1, \dots \vec{a}_r)$, (where inputs are chosen from $\{0,1\}^n$ independently with uniform distribution). Note that we can generate this table by a simple ZNC-algorithm. For any $\vec{\alpha} \in \{0,1\}^n$, we define function $Med(f, T, \alpha) = (1/r) \sum_{i=1}^{r} f(\vec{a}_i \oplus \vec{\alpha})$. Then from Chernoff's bound it is not hard to show that for any random table T of size $r \ge n^3$, the probability that

$$\forall\ \vec{\alpha} \in \{0,1\}^n \ \ |Med(f, T, \alpha) - E| \le (1/n) \tag{2}$$

is at least $1 - 1/n$.

The key idea of our simulation is to use the HSG, given by Theorem 4.2, as a "Verifier" oracle to check the condition in Eq. 2. Indeed, using the NC $(1 - \epsilon, n, \log^d n)$-HSG H with $\epsilon < 1/7$ given by Theorem 4.2, we compute in parallel the following parameters

$$d_{min}(f, T, H) = \min_{\gamma \in \{0,1\}^{k(l(f,n))}} Med(f, T, [H_{l(f,n)}]_{1,n}(\gamma))$$

$$d_{max}(f, T, H) = \max_{\gamma \in \{0,1\}^{k(l(f,m))}} Med(f, T, [H_{l(f,n)}]_{1,n}(\gamma))$$

where $l(f,n)$ is a fixed polynomial in n easily derived from the proof of Lemma 6 of [5]. Further, from this lemma we can prove that since H is a $(\epsilon, n, \log^d n)$-HSG then

$$d_{min}(f,T,H) - \epsilon \ \leq \ \mathbf{Pr}\left(f(x_1,\ldots,x_n)=1\right) = E \ \leq \ d_{max}(f,T,H) + \epsilon \ . \tag{3}$$

The claim given by Eq. 2 implies that $d_{max} - d_{min} \leq 2/n$ with probability at least $1-(1/n)$. Hence the ZNC-algorithm simply checks condition $d_{max} - d_{min} \leq 2/n$. If this inequality is true then the ZNC-algorithm just check the value $d_{max} - d_{min}/2$ and decide whether to accept or not. If instead the inequality is not true then it will answer a 'failure' answer. The correctness of the algorithm is then a consequence of Eq. 3.

□

5 Hitting sets from hard Boolean operators

The construction of an efficient HSG from a Boolean operator which has hard circuit-complexity is based on the following "contradiction" argument. Suppose that a Boolean operator $T \ : \ \{0,1\}^m \longrightarrow \{0,1\}^n$ is not a HSG for a certain class of circuits defined by the parameters $\epsilon(n)$ and $\beta(n)$ (see Def. 1.1). Roughly speaking, this negative fact implies that the output sequence of T can be represented by a new binary sequence which contains a "large" number of 0's (this number depends on $\epsilon(n)$ and $\beta(n)$). Then, using Andreev *et al*'s technique shown in [5], it is possible to compress this new binary sequence in order to prove an upper bound on the circuit complexity of T. This bound is obtained by a new analysis of the compression rate achieved by this technique and by applying the upper bound for the Shannon function $L(n,N,m)$ in Theorem 2.2. If T is supposed to have a hard circuit complexity, we get a contradiction.

5.1 Compressing Boolean operators

Let $T \ : \ \{0,1\}^m \rightarrow \{0,1\}^n$ and $C(x_1,\ldots,x_n)$ be a circuit with n inputs. Given $\vec{\alpha} \in \{0,1\}^n$, consider the function $Med(f,T,\vec{\alpha}) = 2^{-m}\sum_{\vec{u}\in\{0,1\}^m} C(T(\vec{u}) \oplus \vec{\alpha})$ (as in the proof of Corollary 4.1). It is easy to prove that $\mathbf{E}\left(Med(C,T,\vec{\alpha})\right) = \mathbf{Pr}\left(C(x_1,\ldots,x_n)=1\right)$ where the expected value is computed with respect to $\vec{\alpha}$. We briefly describe here the *Andreev et al*'s technique introduced in [5]. Let $\vec{\alpha}_1$ and $\vec{\alpha}_2$ be two different elements in $\{0,1\}^n$. Define $d_1 = Med(C,T,\vec{\alpha}_1)$ and $d_2 = Med(C,T,\vec{\alpha}_2)$ and assume that $D = d_2 - d_1 > 0$. The j-th component of $\vec{a}$ will be denoted as $[\vec{a}]^j$. Since we are considering the case in which $D > 0$, we can assume that there exists an index s for which $[\vec{\alpha}_1]^s \neq [\vec{\alpha}_2]^s$. Consider the operator $T^{\#} \ : \{0,1\}^m \rightarrow \{0,1\}^n$ defined as follows $T^{\#}(\vec{u}) = T(\vec{u}) \oplus ([T(\vec{u})]^s \cdot (\vec{\alpha}_1 \oplus \vec{\alpha}_2))$ where the operation "$\cdot$" is the standard scalar product. The s-th component of $T^{\#}(\vec{u})$ satisfies the following equations:

$$[T^{\#}(\vec{u})]^s = [T(\vec{u})]^s \oplus ([T(\vec{u})]^s \cdot ([\vec{\alpha}_1]^s \oplus [\vec{\alpha}_2]^s)) = [T(\vec{u})]^s \oplus [T(\vec{u})]^s \cdot 1 \ = 0 \ . \tag{4}$$

Observe also that the set $\{T^{\#}(\vec{u}) \oplus \vec{\alpha}_1, T^{\#}(\vec{u}) \oplus \vec{\alpha}_2\}$ is equal to the set $\{T(\vec{u}) \oplus \vec{\alpha}_1 \ , \ T(\vec{u}) \oplus \vec{\alpha}_2\}$. Let

$$N(\sigma,\phi_1,\phi_2) \ = \ |\{\vec{u} \ : \ [T(\vec{u})]^s = \sigma \ , \ C(T(\vec{u}) \oplus \vec{\alpha}_1) = \phi_1 \text{ and } C(T(\vec{u}) \oplus \vec{\alpha}_2) = \phi_2\}| \ . \tag{5}$$

We can now introduce the function which approximates the s-th component of $T(\vec{u})$. Consider the function Q defined as follows:

$$Q_{N(\sigma,\phi_1,\phi_2)}(x,y) \;=\; \begin{cases} x & \textbf{if} \quad\quad\quad\quad\quad x \neq y \\ 1 & \textbf{if} \quad x=y=0 \text{ and } N(1,0,0) \geq N(0,0,0) \\ 0 & \textbf{if} \quad x=y=0 \text{ and } N(1,0,0) < N(0,0,0) \\ 1 & \textbf{if} \quad x=y=1 \text{ and } N(1,1,1) \geq N(0,1,1) \\ 0 & \textbf{if} \quad x=y=1 \text{ and } N(1,1,1) \geq N(0,1,1) \end{cases}$$

In which follows we will consider the function N as a fixed parameter, and thus we will omit the index $N(\sigma,\phi_1,\phi_2)$ in the definition of Q. Then the approximation function for the s-th bit of $T(\vec{u})$ is $Z(\vec{u}) = Q(C(T^{\#}(\vec{u}) \oplus \vec{\alpha}_1), C(T^{\#}(\vec{u}) \oplus \vec{\alpha}_2))$, $i = 1,\ldots,m$. Our next goal is to estimate the number of errors generated by $Z(\vec{u})$. Let $ND(\sigma,\phi_1,\phi_2)$ be the number of inputs $\vec{u}$ such that the following conditions are satisfied: *i)* $[T(\vec{u})]^s \oplus Z(\vec{u}) = 1$ (i.e. there is an error); *ii)* $[T(\vec{u})]^s = \sigma$; *iii)* $C(T(\vec{u}) \oplus \vec{\alpha}_1) = \phi_1$; *iv)* $C(T(\vec{u}) \oplus \vec{\alpha}_2) = \phi_2$.

The following Lemma gives an upper bound on the number of errors in approximating the s-th bit of $T(\vec{u})$.

Lemma 5.1 *[5]* $\sum_{(\sigma,\phi_1,\phi_2)\in\{0,1\}^3} ND(\sigma,\phi_1,\phi_2) \leq m\left(\frac{1}{2} - \frac{d_2-d_1}{2}\right)$.

5.1.1 Some new hardness-compression trade-offs

Using Lemma 5.1, we are now able to give an useful bound on the circuit complexity of T. Observe that function $U(\vec{u}) = [T(\vec{u})]^s \oplus Z(\vec{u})$ with $\vec{u} \in \{0,1\}^m$, singles out the positions in T for which an error occurs. The following Lemmas are easy consequences of Lemma 5.1, and our Theorem 2.2 (the proofs are sketched in the Appendix).

Lemma 5.2 $L(T) \;\leq\; L^{op}(m, n-1) + L(U) + O(L(C)) + O(n)$.

Lemma 5.3 *If for some constant c_1 we have that $D \geq c_1$, then there exists a constant $c_2 < 1$ such that $L(U) \;\leq\; c_2(2^m/m)$.*

5.2 The Hitting Set Generator

In order to derive our HSG, we will make use of the following result given by Lupanov (see also [18]). Let $L^{op}(k,n)$ denote the worst-case circuit complexity of Boolean operators having k variables and n outputs.

Theorem 5.1 *[11]* $L^{op}(k,n) = (1+o(1))(2^k n)/(k+\log n)$.

Theorem 5.2 *Assume that a quick operator $H : k(n) \to n$ exists such that $k(n) = (1+\Theta(1))\log n$, and for a.e. n $L^{op}(H_n) \;\geq\; L(k(n),n) - (2^{k(n)})/(k(n)^2)$. Then, it is possible to construct a $(1/2,n)$-HSG $H' : k'(n) \to n$ such that $k'(n) = \Theta(\log n)$. Hence, $P = BPP$.*

Sketch of the proof. Let $k'(n) = k(n + 2^{t(n)}) + t(n)$ where $t(n) = \lceil 2\log n \rceil$, and define, for any $\vec{a} \in \{0,1\}^{k(n+2^{t(n)})}$ and $\vec{b} \in \{0,1\}^{t(n)}$, $H'_n(\vec{a},\vec{b}) = H_{n+\phi(\vec{b})}(\vec{a})$, where $\phi(\vec{b})$ denotes the standard decimal representation of $\vec{b}$ (note that if the length of $\vec{a}$ is greater than $k(n+\phi(\vec{b}))$, we simply ignore the last bits of $\vec{a}$). From the assumption on $k(n)$, it is not hard to check that there exists a constant $\delta > 0$ such that, for any sufficiently large n, two distinct integers n_1, n_2 exist such that $n < n_1 < n_2 \leq n + 2^{t(n)}$, $n_2 = \lceil (1 + \delta n_1) \rceil$ and $k(n_1) = k(n_2)$.

Let thus $k = k(n_1) = k(n_2)$. From Theorem 5.1, we have

$$L^{op}(k, n_2) - L^{op}(k, n_1) \sim \frac{2^k n_2}{k + \log n_2} - \frac{2^k n_1}{k + \log n_1} \sim (n_2 - n_1) \frac{2^k}{k + \log n_1} ,$$

and there exists n_3, such that $n_1 \leq n_3 < n_2$, for which

$$L^{op}(k, n_3 + 1) - L^{op}(k, n_3) \geq (1 - o(1)) \frac{2^k}{k + \log n_1} . \tag{6}$$

Suppose now that our generator does not hit a Boolean function $f(x_1, \ldots, x_n)$ such that $\mathbf{Pr}\,(f = 1) \geq 1/2$, and $L(f) \leq n$. It follows that for $d_1 = Med(f, H_{n_3+1}, \vec{0})$, we have $d_1 = 0$. But we also have that an $\alpha_2 \in \{0,1\}^n$ exists for which $d_2 = Med(f, H_{n_3+1}, \alpha_2) \geq \frac{1}{2}$. By applying Lemma 5.2 and Lemma 5.3, there exists a constant $0 < c_2 < 1$ such that $L^{op}(H_{n_3+1}) \leq L^{op}(k, n_3) + c_2(2^k/k) + O(L(f)) + O(n)$, and $L^{op}(k, n_3) \geq L^{op}(H_{n_3+1}) - c_2(2^k/k) - O(n)$. Since $L^{op}(H_{n_3+1}) \geq L^{op}(k, n_3 + 1) - (2^k/k^2)$, it follows that $L^{op}(k, n_3 + 1) \leq L^{op}(H_{n_3+1}) + (2^k/k^2)$, and

$$L^{op}(k, n_3 + 1) - L^{op}(k, n_3) \leq \frac{2^k}{k^2} + c_2 \frac{2^k}{k} + O(n) . \tag{7}$$

The value $c_2 < 1$ depends only on $D = d_2 - d_1 = 1/2$, consequently is constant. Without loss of generality, we can assume that $k(n) \geq q \log n$, for some convenient big constant q (it is sufficient to consider the first n output bits of H_{n^r} , where r depends on q). Then, comparing Eq. 6 and Eq. 7, we have a contradiction. It follows that H' is an $(\frac{1}{2}, n)$-HSG. □

Acknowledgements. We are grateful to Luca Trevisan for several interesting comments.

References

[1] Ajtai M, Komlos J, and Szemeredi E. (1987), Deterministic simulation in LOGSPACE, Proc. of *19th ACM STOC*, 132-140.

[2] Alon N. (1986), “Eigenvalues and Expanders”, *Combinatorica*, 6, 83-96.

[3] Alon N. and Spencer J.H. (1992), *The Probabilistic Method*, Wiley-Interscience Publication.

[4] Andreev A., Clementi A., and Rolim J. (1996), “Optimal Bounds on the Approximation of Boolean Functions, with Consequences on the Concept of Hardness”, in *XIII STACS'96*, LNCS, 1046, 319-329. Also available via ftp/WWW in the electronic journal *ECCC* (TR95-041).

[5] Andreev A., Clementi A., and Rolim J. (1996), “Hitting Sets Derandomize BPP”, in *XXIII International Colloquium on Algorithms, Logic and Programming (ICALP'96)*, LNCS. Also available via ftp/WWW in the electronic journal *ECCC* (TR95-061)

[6] Andreev A., Clementi A., Rolim J., and Trevisan L.(1996), “Simulating BPP using very weak random sources: a non oblivious approach”, unpublished manuscript.

[7] Andreev A., I. Vihlyantsev "On the circuit complexity of Covering Boolean Operators", in preparation (1996).

[8] Blum M., and Micali S. (1984), "How to generate cryptographically strong sequences of pseudo-random bits", *SIAM J. of Computing*, 13(4), 850-864.

[9] A. Lubotzky, R. Phillips, and P. Sarnak. (1988), "Ramanujan graphs", *Combinatorica*, 8(3):261–277, 1988.

[10] Lupanov, O.B. (1956) "About gating and contact-gating circuits", *Dokl. Akad. Nauk SSSR* 111, 1171-11744.

[11] Lupanov, O.B. (1965), "About a method circuits design – local coding principle", *Problemy Kibernet.* 10, 31-110 (in Russian).

[12] G.A. Margulis (1973), "Explicit Construction of Concentrators", *Problems of Inform. Transmission*, 325-332.

[13] Nechiporuk E.I. (1965) , "About the complexity of gating circuits for the partial Boolean matrix", *Dokl. Akad. Nauk SSSR*, 163, 40-42.

[14] Nisan N., and Wigderson A. (1994), "Hardness vs Randomness", *J. Comput. System Sci.* 49, 149-167 (also presented at the *29th IEEE FOCS*, 1988).

[15] Sipser M. (1986), "Expanders, Randomness or Time vs Space", in *in the 1st Conference on Structures in Complexity Theory*, LNCS 223, 325-329.

[16] Srinvasan A., and Zuckermann D. (1994), "Computing with very weak random sources", in *28th IEEE FOCS*, 264-275.

[17] Ta-Shma A. (1996), "On extracting randomness from weak random sources", in *28th ACM STOC*,

[18] Wegener, I. (1987), *The complexity of finite Boolean functions*, *Wiley-Teubner Series in Computer Science.*

[19] Yao A. (1982), "Theory and applications of trapdoor functions", in *23th IEEE FOCS*, 80-91.

[20] Zuckermann D. (1996), "Randomness-Optimal Sampling, Extractors, and Constructive leader Election", in *28th ACM STOC*, 286-295.

A APPENDIX: The proofs

A.1 Proof of Theorem 2.2

Let $\mathcal{F}(n,N,m)$ be the set of all partial Boolean functions $f(x_1,\ldots,x_n)$ defined on $N \le 2^n$ inputs and assuming 1 on $m \le N$ inputs. Furthermore, $L(n,N,m)$ denotes the worst-case circuit complexity of functions in $\mathcal{F}(n,N,m)$, and $L_{depth}(n,N,m)$ denotes the maximum value $L_{depth}(f)$ among all functions f from $\mathcal{F}(n,N,m)$. Lupanov [11] proved an optimal bound for $L(n,N,m)$ when $N = 2^n$ (i.e. for total functions). In order to use this bound in the construction of our HSG's, we instead require the precise bound for partial Boolean functions.

Theorem.

$$L(n,N,m) \;=\; (1+o(1))\frac{\log\binom{N}{m}}{\log\log\binom{N}{m}} + O(n)\ .$$

Further, there exists constant $c > 0$ such that

$$L_{c\log n}(n,N,m) \;=\; (1+o(1))\frac{\log\binom{N}{m}}{\log\log\binom{N}{m}} + O(n)\ .$$

Proof. The method used to derive the above upper bound is based on a probabilistic construction of *linear operators* having some new variants of the "well-distribution" property previously shown in [4] to obtain optimal bounds on the circuit complexity of approximating Boolean functions.

A Boolean function $l : \{0,1\}^n \to \{0,1\}$ is linear if it can be represented in the following way:

$$l(x_1,\ldots,x_n) = \alpha_1 x_1 \oplus \ldots \oplus \alpha_n x_n \oplus \beta\ ,$$

where $\alpha_1,\ldots,\alpha_n,\beta$ are Boolean constants. The set of all linear functions with n variables is denoted as $\mathcal{L}_n$. Moreover, a vector function $\vec{l} = (l_1, l_2, \ldots, l_s) \in (\mathcal{L}_n)^s$ $(s \ge 1)$ is called *linear operator*. The circuit complexity of linear operators has been studied in [13]. In particular, we will use the following result [13]. For any linear operator $\vec{l} = (l_1,\ldots,l_s) \in (\mathcal{L}_n)^s$ $(s \ge 1)$, we have

$$L(\vec{l}) \;=\; O\left(\frac{ns}{\log n}\right) + O(n)\ . \tag{8}$$

We are now ready to give the proof of our theorem. For the sake of brevity, we will only prove the restricted case

$$n^{1+\epsilon} \le m \le n^{O(1)} \text{ for some } \epsilon > 0, \text{ and } N = 2^{\Omega(n)}\ , \tag{9}$$

which is what we need to construct our HSG's in the next sections. The proof of the general case will be given in the full version of this paper, but is not relevant for our goal. The proof consists of a reduction from our case to that of total functions for which we can apply Theorem 2.1. Consider a partial Boolean function $f(x_1,\ldots,x_n)$, let M_α be the set of vector $\vec{a} \in \{0,1\}^n$ for which $f(\vec{a}) = \alpha$. We have $|M_1| = m$ and $|M_0| = N - m$. Consider a randomly chosen linear operator (with uniform distribution) $\vec{l} = (l_1, \ldots, l_k) \in (\mathcal{L}_n)^k$ where $k = \lceil \log N + \log m \rceil + 2$. Then, observe that for any choice of two fixed elements $\vec{a}, \vec{b} \in \{0,1\}^n$ such that $\vec{a} \neq \vec{b}$, we have

$$\mathbf{Pr}\left(\vec{l}(\vec{a}) = \vec{l}(\vec{b})\right) = 2^{-k} \ .$$

Consequently,

$$\mathbf{Pr}\left(\exists \vec{a} \in M_1 \ \exists \vec{b} \in M_0 \ : \ \vec{l}(\vec{a}) = \vec{l}(\vec{b}) \ \right) \leq |M_0| * |M_1| * 2^{-k} \leq 1/4 \ .$$

Hence, for the negation of the above event we have

$$\mathbf{Pr}\left(\forall \vec{a} \in M_1 \ \forall \vec{b} \in M_0 \ : \ \vec{l}(\vec{a}) \neq \vec{l}(\vec{b}) \ \right) \geq 3/4 \ .$$

From the above probabilistic argument, we can state that there exists $\vec{l} \in (\mathcal{L}_n)^k$ such that

$$\forall \vec{a} \in M_1 \ \forall \vec{b} \in M_0 \ : \ \vec{l}(\vec{a}) \neq \vec{l}(\vec{b}) \tag{10}$$

We define the total Boolean function $g(y_1, \ldots, y_k)$ as follows

$$g(\vec{y}) = 1 \text{ if } \exists \ \vec{a} \in M_1 \text{ such that } \ \vec{l}(\vec{a}) = \vec{u} \ \text{ and } \ g(\vec{y}) = 0 \ \text{ otherwise} \ .$$

Note that if f is defined on $\vec{a}$ then $f(\vec{a}) = g(\vec{l}(\vec{a}))$; Then, From Eq. 8, we obtain that for some constant c_1

$$L_{2c_1 \log n}(f) \leq L_{c_1 \log n}(\vec{l}) + L_{c_1 \log n}(g) \leq$$
$$\leq O(n(\log N + \log m + 2)) + L_{c_1 \log n}(g) \leq O(n^2) + L_{c_1 \log n}(g) \ .$$

Furthermore, Condition 9 implies that

$$\log \binom{N}{m} = (1 + o(1)) m \log N \ , \ \text{ and } \ n^2 = o(\frac{\log \binom{N}{m}}{\log\log \binom{N}{m}}) \ .$$

Since g is a total function we can apply Theorem 2.1, i.e. for some constant c_2

$$L_{c_2 \log n}(g) \leq (1 + o(1)) \frac{\log \binom{N}{m}}{\log\log \binom{N}{m}} \ .$$

Without loss of generality we can suppose that $c_1 > c_2$, and consequently

$$L_{2c_1 \log n}(f) \leq O(n^2) + L_{c_2 \log n}(g) \leq (1 + o(1)) \frac{\log \binom{N}{m}}{\log\log \binom{N}{m}} \ .$$

The thesis is then proved by defining $c = 2c_1$.

□

A.2 Proof of Lemma 3.1

Lemma. If $\mathbf{Pr}\,(g = 0) \leq c < \frac{1}{2}$, then

$$\mathbf{Pr}\left(EPR^{g}_{n,t} = 0\right) \leq \left(c + \frac{\lambda}{d}\right)^{2^t - 2} .$$

Sketch of the proof. Let $0 < \alpha < 1$, and let $W \subseteq V_n$ be any subset such that $|W| \leq \alpha n$. From Theorem 3.2, we know that the number of walks of length m in EP_n that avoid W is at most

$$|V_n|(1-\alpha)^{1/2}((1-\alpha)d^2 + \lambda^2)^{m/2} .$$

Furthermore, observe that the number of all walks of length in EP_n is $|V_n|d^m$, and the value $\mathbf{Pr}\,(g = 0)$ computed on the set of vertices representing strings with last components equal to 0 is at most $2c$. It follows that

$$\mathbf{Pr}\left(EPR^{g}_{n,t} = 0\right) \leq \sqrt{2c}\left(2c + \frac{\lambda^2}{d^2}\right)^{2^t - 1} \leq \left(c + \frac{\lambda}{d}\right)^{2^t - 2} .$$

□

A.3 Proofs of Lemmas 5.2 and 5.3

Lemma.

$$L(T) \;\leq\; L^{op}(m, n-1) + L(U) + O(L(C)) + O(n) .$$

Sketch of the proof. Observe first that $[T(\vec{u})]^s = U(\vec{u}) \oplus Z(\vec{u}) = U(\vec{u}) \oplus Q(C(T^{\#}(\vec{u}) \oplus \alpha_1), C(T^{\#}(\vec{u}) \oplus \alpha_2))$. IConsequently, T can be computed as follows

$$T(\vec{u}) \;=\; T^{\#}(\vec{u}) \oplus (\vec{e}^{\,n}_s \cdot (U(\vec{u})) \oplus Q(C(T^{\#}(\vec{u}) \oplus \vec{\alpha}_1), C(T^{\#}(\vec{u}) \oplus \vec{\alpha}_2))) \qquad (11)$$

where $\vec{e}^{\,n}_s \in \{0,1\}^n$ is the Boolean vector having only the s-th component equal to 1. From Eq. 4 we have that $[T^{\#}(\vec{u})]^s = 0$ for any $\vec{u} \in \{0,1\}^m$, hence T has $n-1$ output variables. It follows that the thesis is an immediate consequence of Eq. 11. □

Lemma. If for some constant c_1 we have that $D \geq c_1$, then there exists a constant $c_2 < 1$ such that $L(U) \;\leq\; c_2(2^m/m)$.
Sketch of the proof. Lemma 5.1 implies that the number of inputs $\vec{u} \in \{0,1\}^m$ for which $U(\vec{u}) = 1$ is at most $2^m(1/2 - (d_2 - d_1)/2)$. Then the thesis follows from the upper bound shown in Theorem 2.2. □