Towards a Study of Low-Complexity Graphs*

Sanjeev Arora[1], David Steurer[1], and Avi Wigderson[2]

[1] Computer Science Dept, Princeton University, Princeton NJ 08544, USA
{arora,dsteurer}@cs.princeton.edu
[2] Institute for Advanced Study, Princeton NJ
avi@ias.edu

Abstract. We propose the study of graphs that are defined by low-complexity distributed and deterministic agents. We suggest that this viewpoint may help introduce the element of *individual choice* in models of large scale social networks. This viewpoint may also provide interesting new classes of graphs for which to design algorithms.

We focus largely on the case where the "low complexity" computation is $\mathbf{AC}^0$. We show that this is already a rich class of graphs that includes examples of lossless expanders and power-law graphs. We give evidence that even such low complexity graphs present a formidable challenge to algorithms designers. On the positive side, we show that many algorithms from property testing and data sketching can be adapted to give meaningful results for low-complexity graphs.

1 Introduction

This paper tries to highlight some interesting families of graphs that we call *low complexity graphs*. These are graphs whose vertices are vectors of attributes (without loss of generality, vectors in $\{0,1\}^n$ for some n) and whose edge structure has low computational complexity, namely, determining whether (i,j) is an edge or not is a low-complexity computation involving i,j. (This definition assumes an *adjacency matrix* representation of the graph; we have an alternative definition for an *adjacency list* representation.)

There are many motivations for studying this concept. First, from a complexity-theoretic view such graphs are natural if we think of the graph as being defined by computationally bounded distributed agents, and they only use their personal information (namely, their names i,j) while computing their decision. Concretely, one could hope that many graph problems are easier to solve on low-complexity graphs than they are on general graphs (analogously to say, fixed treewidth graphs or minor-excluded graphs).

Second, such low-complexity graphs represent a plausible way to incorporate the concept of *individual choice* in models of formation of social network graphs (e.g., graphs that arise on the Web, Myspace, Facebook, etc.). Empirically these graphs are found to exhibit some strong properties such as *power law* distribution of degrees and top eigenvalues. A large body of work has been used to describe how such graphs can arise naturally as a result of distributed actions. The dominant model is some variant of a *preferential attachment* process, in

* Supported by NSF grants 0830673, 0832797, 528414.

S. Albers et al. (Eds.): ICALP 2009, Part I, LNCS 5555, pp. 119–131, 2009.

which nodes attach to the graph one by one, and when they try to (probabilistically) decide which other nodes to attach to, they prefer nodes with higher degree (i.e., "popular nodes"). By varying model parameters and assumptions (see the survey by Mitzenmacher [1]) a rich variety of graphs can be obtained.

While it is highly plausible that random choice coupled with a "follow the herd" behavior should be an important part of the story of how such networks come about, it is unclear if this is the complete story. For instance, it has been empirically observed that many graph problems are trivial to solve on the above random models but quite difficult on real-life graphs obtained from social networks [2].

One obvious missing element in the above models is that of *individual choice*. Clearly, people link to other people on Myspace or Facebook or the Web at least in part due to their interest profile, and they seek out interesting people or webpages using search tools (e.g., Google and other search engines) that allow them to explicitly or implicitly express (via boolean expressions) their choice function.

Our model of low-complexity graphs gives one way to incorporate the element of individual choice: the choice of a person with attribute vector i to link to a person with vector j is a low complexity computation involving i, j. For sake of clarity the model has been kept very bare: (i) The attributes are distributed uniformly in the population (i.e., the node set is $\{0,1\}^n$); (ii) The choice function for all nodes uses the same low-complexity computation —only their inputs are different, namely, their own attribute vectors. Note that this allows the adjacency list of each node i to be different because its attribute vector is unique.

(iii) The issue of random choice has been left out entirely, or to be more precise, is allowed only in very small doses as part of the choice function. The point in this paper is that even with these restrictions, great expressive power still remains —one can realize complicated graphs like power-law graphs and lossless expanders and extractors (see Section 2).

Now we formally define low complexity graphs. If $\mathcal{C}$ is a complexity class then an infinite family of graphs $\{G_n\}$ is said to be a *$\mathcal{C}$-graph family in the adjacency matrix representation* if for every $n \geq 1$, G_n has 2^n nodes and there is an algorithm in class $\mathcal{C}$ that, given indices $i, j \in \{0,1\}^n$, can compute whether or not (i, j) is an edge in the graph. A *$\mathcal{C}$-graph family in the adjacency list representation* is similarly defined, except we restrict it to be d-regular for some d (which could be a slowly growing function of n) and we require for each input size n a sequence of dn algorithms from $\mathcal{C}$ that given i compute the individual bits of the adjacency list of node i. (The output of each algorithm consists of one bit.) We can similarly define other low-complexity objects such as hypergraphs, circuits, etc.

Of course, similar definitions have been used in complexity theory before. For instance, the case $\mathcal{C} = \mathbf{DSPACE}(n)$ corresponds to *logspace uniform* circuits/graphs, and also corresponds to a common notion of *strongly explicit constructions* (used for example when we talk about explicit constructions of expanders). The case $\mathcal{C} = \mathbf{P}/poly$ coincides with *succinctly described* graphs [3].

It is known that many standard problems (e.g., connectivity) are intractable for succinctly represented graphs, when the input to the algorithm is the circuit that represents the graph. Classes such as **PPAD** [4] are also defined in this context. The difference in our paper is that we are interested in letting $\mathcal{C}$ be a very low complexity classes: say, $\mathbf{NC}^0$, $\mathbf{AC}^0$, or the set of graphs with $\log n$ decision tree complexity. $\mathbf{AC}^0$, the class of functions computable by bounded depth circuits of unlimited fan-in and polynomial size, will be a frequent choice in this paper. Furthermore, in our lower bound results we usually think of the entire graph as being presented to the algorithm (as opposed to just the circuit). The algorithms we present in Section 4 work even in the model where the algorithm is presented the circuit defining the input graph.

In fact, one motivation for studying low-complexity graphs is that this fits in the longer-term program of understanding low-complexity graphs (which are just truth tables of low-complexity functions) as a prelude to proving complexity lower bounds. Of course, the results of Razborov and Rudich [5] caution us against hoping for quick successes in this endeavor if the $\mathcal{C}$ is too complex. Accordingly, this paper generally restricts attention to classes no more powerful than $\mathbf{AC}^0$. Our results suggest that even these graphs can be quite complex and difficult. In Section 3 we show that solving any of the standard **NP** or **P** problems on $\mathbf{AC}^0$-graphs is probably no easier than solving them on general graphs, *even when the entire graph is given as input.*

In light of the hardness results one should relax one's expectations of what kind of algorithms can be possible. One could try for approximation algorithms. We observe using the current proof of the PCP Theorem [6] that approximation is intractable for a host of standard problems on $\mathbf{NC}^1$-graphs (Section 3.1). It is an open problem whether this result can be extended to $\mathbf{AC}^0$-graphs, and a positive resolution seems to require a new proof of the PCP Theorem. We can show however that $\mathbf{AC}^0$ graphs can prove to be difficult (ie have high integrality gaps) for current approximation algorithms that are based upon semidefinite programming (Theorem 9).

In light of the results about seeming difficulties of approximation, one can try to further relax what it means to solve the problem. An attractive approach is to try *sampling-based* algorithms such as the ones developed in property testing and streaming algorithms. We show that many known algorithms can be interpreted as follows: given a circuit representing a low-complexity graph, these algorithms return another low-complexity circuit that approximately represents a solution. (See Section 4.)

Though many known sampling-based algorithms are tight when the graph is given as a black box, it is conceivable that they can be improved upon when the graph is given as a small circuit. (Theorem 11 explains why the black box lower bounds don't apply.) Perhaps designing such algorithms can be a new motivation for revisiting lowerbound techniques for classes such as $\mathbf{AC}^0$, or even $\mathbf{NC}^0$. For instance we do not currently know if constant degree $\mathbf{NC}^0$-graphs can be expanders in the adjacency list representation. We observe in Section 2.1 that expanders of logarithmic degree can be realized as $\mathbf{NC}^0$ graphs.

This paper spans ideas from multiple areas, and consequently we have to assume familiarity with many standard terms and results and necessarily omit many details. We also see the results in this manuscript as representative rather than exhaustive. Even the basic model can be extended in many ways. For instance, if nodes are allowed a small amount of randomness and a small amount of computation then one obtains complexity-based extension of the theory of $G(n,p)$ graphs. We hope that many such extensions will be studied in future.

The recently studied model of random dot product graphs [7,8] shares some ideas with our low complexity graphs. There, the vertices are sampled from some distribution on the sphere, and the probability of an edge is proportional to the inner product of the endpoints. The inner product can be seen as a low-complexity function of the endpoints (albeit not $\mathbf{AC}^0$) that determines the edge probability.

2 Constructions of Interesting $\mathbf{AC}^0$ Graphs

In this section we show how some well-known graphs can be realized as $\mathbf{AC}^0$ graphs. Throughout, N denotes the number of nodes in the graph, and the nodes are assumed to be vectors of n bits (i.e., binary-valued attributes).

To set the context, we start with some simple examples of graphs that are *not* $\mathbf{AC}^0$, by the well-known results of Ajtai [9] and of Furst, Saxe, and Sipser [10].

Example 1 (Graphs that are not $\mathbf{AC}^0$). *Parity Graph*: Its edge set is $\{(x,y) : \oplus_i(x_i \oplus y_i) = 1\}$ where $x, y \in GF(2)^n$ and $\oplus$ addition mod 2.

Inner Product Graph: Its edge set is $\{(x,y) : x \odot y = 1\}$ where $x, y \in GF(2)^n$ and $\odot$ is inner product mod 2.

Threshold graph: Its edge set is $\{(x,y) : x$ and y agree on at least $^2/_3$rd of their bits.$\}$.

The fact that such simple operations are impossible in $\mathbf{AC}^0$ makes the task of designing $\mathbf{AC}^0$ graphs difficult. Next, we list simple graphs that *are* $\mathbf{AC}^0$. These will be building blocks in more complicated constructions later on.

Example 2 (Some $\mathbf{AC}^0$ graphs). *The N-cycle.* The attribute vector labeling the nodes can be interpreted as a binary number in $[0, N-1]$, and the set of edges is $\{x, x+1 \bmod N\}$. Since $x \to x+1$ is computable in $\mathbf{AC}^0$, we conclude that this is an $\mathbf{AC}^0$ graph.

The r-dimensional grid on N vertices. Reasoning similarly as above, this is also an $\mathbf{AC}^0$ graph for every fixed r. This will be useful in Theorem 5.

The following is also $\mathbf{AC}^0$ for every $k \leq n$: the graph whose edge set is $\{(x,y) :$ x, y agree on the first k bits$\}$.

Next, since *approximate thresholds* can be computed probabilistically in $\mathbf{AC}^0$ (actually even in depth 3 [11]), we can construct an $\mathbf{AC}^0$ graph that is an approximate and noisy version of the Threshold graph of Example 1. (This could be useful in modeling social networks because "approximate threshold" of shared attributes could be a plausible strategy for setting up connections.) For any constant ϵ (to be thought of as $o(1)$), the following holds. For (x,y)'s such that x, y

agree in $\geq 2/3 + \epsilon$ of the bits, $\{x, y\}$ is an edge. For (x, y)'s such that x, y agree in $< 2/3 - \epsilon$ of the bits, (x, y) is a non-edge. All other pairs (x, y) may or may not be edges.

Finally, by definition, the set of $\mathbf{AC}^0$ graphs is closed under taking graph complements, union and intersection of edge sets, and taking AND product $G_1 \times G_2$, which is the graph whose vertex set is $V(G_1) \times V(G_2)$ and the edge set is $\{((u_1, v_1), (u_2, v_2)) : (u_1, u_2) \in E(G_1) \text{ AND } (v_1, v_2) \in E(G_2)\}$.

The following fact, a consequence of Håstad's [12] lower bound for $\mathbf{AC}^0$, suggests that every $\mathbf{AC}^0$ graph has some structure. More precisely, it shows that $\mathbf{AC}^0$-graphs are never good Ramsey graphs (in contrast to random graphs where the largest bipartite clique and the largest bipartite independent set are of size $O(\log N)$).

Proposition 1. *There is a polynomial-time algorithm that given an* $\mathbf{AC}^0$ *graph on* N *nodes (in the adjacency matrix representation) finds either a bipartite clique or independent set of size* $2^{\Omega(\log N/\mathrm{poly}(\log \log N))}$ *(in fact the algorithm can find many —say, superlogarithmic—number of these).*

2.1 Expanders and Lossless Expanders

In this subsection we show $\mathbf{AC}^0$-graphs can exhibit highly "pseudorandom" behavior. For a long time the only known explicit constructions of these graphs involved algebraic operations that are provably impossible in $\mathbf{AC}^0$. We use recent constructions involving the zig-zag product. For background and myriad applications of expanders please see the extensive survey of Hoory, Linial, Wigderson [13].

These graphs will be sparse and we choose to describe them in the adjacency list representation. Note that for constant degree graphs the adjacency matrix representation is at least as rich as the adjacency list representation. A d-regular graph $G = (V, E)$ is an *eigenvalue* c*-expander* if the second eigenvalue of the normalized Laplacian of G is at least c (sometimes called the "spectral gap"). In a *lossless expander*, vertex neighborhoods are essentially as large as possible. Graph G is an expander with *loss* ϵ if every set S of size at most $\alpha|V|$ has $|\Gamma(S)| \geq d(1 - \epsilon)|S|$ (where $\alpha = \alpha(\epsilon, d)$ does not depend on the size of the graph). Lossless expanders have proved useful in a host of settings. See Capalbo et al. [14] for the first explicit construction and further references.

Theorem 3. *For every* $\epsilon > 0$ *there is a* $d = d(\epsilon) > 0$ *such that there is an* $\mathbf{AC}^0$ *family of* d*-regular expanders with loss* ϵ*.*

Proof. It will be convenient to think of the graph being represented by a circuit that given (v, i) with $v \in V$ and $i \in [d]$ (given in binary), and its output will be a vertex $u \in V$ which is the ith neighbor of v.

Let $G : N \times M \to N$ and $H : M \times D \to M$ be two circuits (note that the degree of G is the size of H). Then their zig-zag product $G \textcircled{z} H : (N \times M) \times (D^2) \to (N \times M)$ is defined by $(G \textcircled{z} H)((v, k), (i, j)) = (G(v, H(k, i)), H(H(k, i), j))$.

Note that the output of $G ⓩ H$ is simply a composition of the given circuits in serial, and its depth is the sum of depths of their depths (and size at most twice the sum of their sizes).

Below we rely upon two works. We use the simplified expander construction of Alon et al. [15] that gives a c-eigenvalue expander using only two applications of the zig-zag product. Then we do a zigzag product with a constant size graph as in Capalbo et al. [14] to end up with with a lossless expander. We assume familiarity with these constructions and only describe why they work for us.

As in [15] start with a Cayley expander G on $N = (F_2)^n$ of degree $M = n^2$, arising from construction of ϵ-biased sets. As addition in $GF(2)^n$ is carried out in $\mathbf{NC}^0$, this graph is an $\mathbf{AC}^0$-graph (we must move from $\mathbf{NC}^0$ to $\mathbf{AC}^0$ when using the edge index to obtain the actual representation of that vector from the generating set). This graph is composed once with a smaller copy of itself with n^2 vertices, and then again with another expander that is small enough to be trivially an $\mathbf{AC}^0$-graph. This gives an eigenvalue expander that is clearly an $\mathbf{AC}^0$-graph.

Now let us say a few words about the construction of lossless expanders on [14]. For these constructions it is useful to have the circuits output "extra information." Above, given (v, i) they produce only the ith neighbor of v, and now we'll ask them to output more information (whose main function is "keeping the entropy" in the input) beyond the neighbor. E.g. in [16] the circuit G will also include the index of the edge along which that neighboring vertex was reached. This modified circuit, $G : N \times M \to N \times M$ is called there a "rotation map". The objects which are multiplied using the extension of zig-zag in [14] are not expanders, but other pseudorandom objects related to extractors. In some of them as well the output has two parts, carrying a similar information as in the "rotation map".

With this in mind, the construction of lossless expanders in [14] has the exact same structure as the one above, and explicitly describes them as composition of functions (and as noted above, easily interpreted in circuit terms). The two components needed are an eigenvalue expander, which was constructed above in $\mathbf{AC}^0$ and a constant-sized graph, which is in $\mathbf{AC}^0$ trivially. □

Open questions for $\mathbf{NC}^0$ graphs. Can $\mathbf{NC}^0$-graphs be constant-degree expanders? We note that in the adjacency matrix representation the answer is "No", since any nontrivial $\mathbf{NC}^0$ circuit cannot even represent a constant-degree graph.

However, in the adjacency list representation the answer to the question is less clear (e.g., logarithmic degree $\mathbf{NC}^0$ expanders exist in this representation).

2.2 Constructions of Power-Law Graphs

Many real-life graphs have a degree distribution that satisfies the *power law*: *for some parameter $\alpha > 0$, the number of nodes of degree $> x$ is proportional to $x^{-\alpha}$.* As mentioned in the introduction, the standard explanation for these is some form of randomized attachment process where nodes favor other nodes with high degree ("popular nodes"). It has also been observed empirically that the largest few eigenvalues also satisfy a power law with exponent $\alpha/2$: *if the largest degrees*

are $d_1, d_2, d_3, \ldots$, the largest eigenvalues are close to $\sqrt{d_1}, \sqrt{d_2}, \sqrt{d_3}, \ldots$. This has also been explained for random graph models by Mihail and Papadimitriou [17].

Theorem 4. *For every $\alpha > 2$ there is an $\mathbf{AC}^0$ family of graphs whose degrees satisfy the power-law with exponent α, and furthermore the top $k = N^{1/2\alpha}$ eigenvalues satisfy the eigenvalue power law.*

Proof. Many constructions are possible but since we only have to ensure power-laws for the degrees and some top eigenvalues we give a particularly easy one.

We note that since $\alpha > 2$ the number of edges is $\sum_d N/d^{\alpha-1} = O(N)$ and the max degree is less than $N^{1/\alpha}$. For simplicity we make all degrees powers of 2. We divide vertices into $O(\log n)$ classes $S_1, S_2, \ldots, S_k$, where $|S_i| = N/2^{i\alpha}$ and degree of vertices in S_i is close to 2^i (we say "close to" because the construction will allow degrees to deviate a little). Thus the maximum degree vertex is in S_k and has degree $N^{1/\alpha}$.

Vertices in S_i consist of vectors in $GF(2)^n$ whose first $i\alpha$ bits are 0 and the bit immediately after these is 1. Thus the S_i's are disjoint, and furthermore verifying whether a vector x lies in S_i is an $\mathbf{AC}^0$ computation.

For now we describe the construction as randomized and allow poly(n) random bits, which can obviously be appropriately chosen and hardwired so that certain bad events listed below do not happen. For each i pick a random shift $a_i \in GF(2)^n$ and connect $x \in S_i$ to $x + a_i + b$ where b is a vector whose last $n - i$ bits are zero. If $x + a_i + b$ is in S_j for $j \geq 5$ then leave out the edge. Note that the adjacency matrix of these connections is computable in $\mathbf{AC}^0$.

By construction, the degree of each node in S_i is almost 2^i, except that an occasional node x may have some missing edges because $x+a_i+b$ happened to lie in $\cup_{j\geq 5} S_j$. Since this set has size $\sum_{j\geq 5} N/2^{j\alpha} < N/2^{5\alpha-1}$ the chance that this happens (since shift a_i is random) is less than $1/2^{5\alpha-1}$. By Markov's inequality the fraction of nodes in S_i (for $i \geq 7$) that have more than $1/10$ th of their edges missing is less than $1/4$.

Now we argue about the top eigenvalues along the lines of Mihail and Papadimitriou. One can show that the edges of k highest-degree vertices form a union of (mostly) disjoint stars with high probability. Since the largest eigenvalue of a d-star is $\sqrt{d-1}$, the graph formed by these edges has largest eigenvalues roughly $\sqrt{d_1}, \ldots, \sqrt{d_k}$. Furthermore, it is true that with high probability, for all $i \in [k]$, our graph contains much less than $\sqrt{d_i}$ edges between the leaves of the d_i-star. Hence, by eigenvalue interlacing, even including these edges does not spoil the eigenvalue power law. Finally, one can show that the maximum degree of the remaining edges is too small to influence the first k eigenvalues. □

3 Hardness Results

One reason to study specific graph families such as bounded tree-width graphs is that one can often solve certain problems more easily on them as compared to general graphs. In this section we explore whether such better algorithms exist

for low-complexity graphs. The negative results in this section will guide the choice of what kind of algorithms to expect.

Note that N is the number of nodes in the graph, $n = \log_2 N$ is the number of labels in the vertices, and the algorithm is provided the entire graph as input.

The first result shows that there are unlikely to be efficient algorithms for the usual **NP**-complete problems on $\mathbf{AC}^0$-graphs. We say the **NP**-complete problem is *usual* if its **NP**-completeness can be proved using the classical Cook-style proof followed by a simple gadget-based reduction in the style of Karp. (This notion is obviously not precise. The precise version of the next theorem would involve replacing "usual **NP**-complete" by a list of a few thousand explicit problems.)

Theorem 5. *If* $\mathbf{NEXP} \neq \mathbf{EXP}$ *then none of the usual* ***NP****-complete problem can be solved on* $\mathbf{AC}^0$*-graphs in polynomial time.*

Proof (sketch). Suppose a polynomial-time algorithm were to solve a usual **NP**-complete problem, say CLIQUE, on $\mathbf{AC}^0$-graphs. We show how to use it to decide any language $L \in \mathbf{NTIME}(2^{n^c})$ deterministically in time $2^{O(n^c)}$, which contradicts the Theorem's hypothesis.

For any input $x \in \{0,1\}^n$ we can use the standard Cook-Karp style reduction to produce a graph with $2^{O(n^c)}$ nodes which has a clique of a certain size iff $x \in L$. Let us show that this instance is an $\mathbf{AC}^0$-graph, and therefore amenable to be solved by the algorithm.

Recall that the Cook-style reduction consists of writing constraints for the 2×3 "windows" in the tableau of M's computation on x. The window —and hence also each edge in the instance of CLIQUE—is defined by indices of the type $(i + b_1, j + b_2)$ where $b_1 \in \{0,1,2\}, b_2 \in \{0,1\}$, and $i, j \leq 2^{n^c}$. As noted in Example 2 the function $i \to i + 1$ is computable in $\mathbf{AC}^0$ when i is given in binary, we can easily design an $\mathbf{AC}^0$ circuit of size $\mathrm{poly}(n^c)$ (with x hardwired in it) that represents the CLIQUE instance. This shows that the instance is an $\mathbf{AC}^0$-graph. □

At first glance the use of the conjecture $\mathbf{NEXP} \neq \mathbf{EXP}$ (which implies $\mathbf{P} \neq \mathbf{NP}$) in the previous result may seem like overkill. But there is a (folklore) converse of sorts known.

Theorem 6. *If* $\mathbf{NEXP} = \mathbf{EXP}$ *then every* ***NP*** *problem on* $\mathbf{P}/\mathrm{poly}$*-graphs (thus, also on* $\mathbf{AC}^0$*-graphs) can be solved deterministically in* $n^{\mathrm{poly}(\log n)}$ *time.*

Proof (sketch). Suppose A is an exponential time deterministic algorithm for a **NEXP**-complete problem. Let L be any **NP** language. To solve it on $\mathbf{P}/\mathrm{poly}$-graphs of size N, we note that the input can be represented by the circuit whose size is $\mathrm{poly}(\log N)$. Though the circuit is not provided as part of the input, it can be recovered in $\exp(\mathrm{poly}(\log N))$ time by exhaustive search. Now we can think of the problem as really one in **NEXP** where the input is this circuit. Now we can use A to solve this problem in time $\exp(\mathrm{poly}(\log N))$. □

How about **P**-complete problems on $\mathbf{AC}^0$ graphs? Can we solve them more efficiently than on general graphs? The following theorem –proved completely analogously to the previous two theorems—suggests that we cannot.

Theorem 7. *If* $\mathbf{EXP} \neq \mathbf{PSPACE}$ *then no usual* $\boldsymbol{P}$*-complete problem can be solved on* $\mathbf{AC}^0$*-graphs in polylog space. If* $\mathbf{EXP} = \mathbf{PSPACE}$ *then every problem in* $\boldsymbol{P}$ *can be solved on* $\boldsymbol{P}/\mathrm{poly}$*-graphs in polylog space.*

3.1 Results about Hardness of Approximation

We already saw that problems like CLIQUE cannot be optimally solved on $\mathbf{AC}^0$-graphs in polynomial time if $\mathbf{NEXP} \neq \mathbf{EXP}$. What about approximation? We note that the current proof of the PCP Theorem (specifically, the generalization from **NP** to **NEXP**) and related results imply that current inapproximability results can be transfered to show hardness of the same problem on $\mathbf{NC}^1$-graphs. The following is a sample result.

Theorem 8. *If* $\mathbf{NEXP} \neq \mathbf{EXP}$ *then the MAX-CLIQUE problem on* $\boldsymbol{NC}^1$*-graphs cannot be approximated within any constant factor in polynomial time.*

Proof (sketch). The only known proof [18,19,20] of the result $\mathbf{NEXP} = \mathbf{PCP}(\mathrm{poly}(n), 1)$ involves taking polynomial extension and then using procedures for polynomial testing/correcting and sum-check. These involve finite field arithmetic on n-bit vectors (where the graph size is 2^n), which is possible in $\mathbf{NC}^1$. The proof has a second step involving verifier composition which is trivially in $\mathbf{NC}^1$ because of the smaller inputs involved.

With these observations and the known reduction from $\mathbf{PCP}(\mathrm{poly}(n), 1)$ to approximating MAX-CLIQUE [21] we can finish the proof as in Theorem 5. □

The stumbling block in proving a similar result for $\mathbf{AC}^0$ graphs is that current PCP constructions rely upon field operations that cannot be done in $\mathbf{AC}^0$. It is conceivable that this is inherent, and one way to show this would be to give a better CLIQUE approximation algorithm for $\mathbf{AC}^0$ graphs. This would probably involve an interesting new result about $\mathbf{AC}^0$.

At the same time, simple approximation algorithms such as basic SDP relaxations will probably not lead to better approximation in most cases. For example, for MAX-CUT the integrality gap of the standard SDP relaxation is achieved on instance of finite size, which are clearly $\mathbf{AC}^0$.

Theorem 9. *The worst-case integrality gaps of the standard SDP relaxation on* $\mathbf{AC}^0$ *graphs for MAX-CUT (see [22]) is* 0.878.. *(i.e., same as for general graphs).*

4 Algorithms for Low-Complexity Graphs

The hardness results in Section 3 greatly constrict our options in terms of what kinds of algorithms to shoot for on $\mathbf{AC}^0$ graphs.

Of course, the only hope in designing such algorithms is to exploit something about the structure of $\mathbf{AC}^0$ graphs (e.g., Proposition 1). Unfortunately, this hope is somewhat dashed in Section 3.

In light of these hardness results, it is reasonable to turn to sampling-based algorithms from areas such as property testing and sublinear algorithms, which result in a fairly weak approximation (necessarily so, since only a small portion of the graph is examined). Since the introduction of the property testing idea by Goldwasser, Goldreich, and Ron [23], a large body of literature has grown up in this area. Note that such algorithms are natural to consider for $\mathbf{AC}^0$ graphs, since we are given a circuit oracle for the edges.

We notice that many algorithms in this area implicitly give a stronger result than formally stated: in many cases if the graph is $\mathbf{AC}^0$, then the approximate *solution* to the problem (namely, a cut, assignment, etc.), which is an object of size $\exp(n)$, can also be represented as an $\mathbf{AC}^0$ circuit. As an illustrative example, we use MAX-CUT.

Theorem 10. *For any $\epsilon > 0$ there is a polynomial-time randomized algorithm that, given an $\mathbf{AC}^0$ circuit computing $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ representing a graph in the adjacency matrix representation, produces an $\mathbf{AC}^0$ circuit computing some $g\colon \{0,1\}^n \to \{0,1\}$ circuit such that the cut $(g^{-1}(0), g^{-1}(1))$ in the graph has capacity at least $OPT - \epsilon n^2$ where OPT is the capacity of a MAX CUT in the graph.*

Proof. For any $\epsilon > 0$, the sampling algorithm of Alon et al. [24] samples $k = O(1/\epsilon^5)$ vertex pairs and examines whether or not the corresponding edges are present in the graph. This suffices for it to estimate the capacity of the maximum cut within additive error ϵN^2. In fact the proof of correctness shows something stronger. It gives a function $h\colon \{0,1\}^{\binom{k}{2}} \times \{0,1\}^k \to \{0,1\}$ such that the following is true for any graph $G = (V, E)$ of any size. For any subset of k vertices S let $E_S \in \{0,1\}^{\binom{k}{2}}$ be the characteristic vector showing which of the $\binom{k}{2}$ pairs of S are connected by an edge. If v is any other vertex, let v_S be a characteristic vector showing which of the k nodes of S are connected to v by an edge. For every S let $g_S\colon V \to \{0,1\}$ be defined as $g_S(v) = h(E_S, v_S)$. (Thus g_S implicitly defines a cut of the graph.) Then if S is chosen uniformly at random, then the probability is at least 0.99 that the cut represented by g_S has capacity at least $OPT - \epsilon n^2$

Thus the randomized algorithm is to sample S at random and output the function g_S. Note that if the graph was $\mathbf{AC}^0$ then so is g_S. □

In fact, using the subsequent work of Alon et al. [25] there is a similar algorithm (though again not mentioned explicitly) as in Theorem 10 for every testable graph property. For example, the property of being C-colorable is testable for any constant C. The analog of Theorem 10 for this problem shows that the final $\mathbf{AC}^0$ circuit will compute a C-coloring that is a proper coloring of some graph that differs in at most ϵn^2 edges from G. The sampling complexity is much worse than for MAX-CUT, though still constant for every constant $\epsilon > 0$.

4.1 Algorithms for the Adjacency List Representation

Property testing has also been studied in the adjacency list representation, and some algorithms transfer to the $\mathbf{AC}^0$ setting. For instance the work of [26] implies

that it is possible to test in $\text{poly}(n/\epsilon)$ time whether the $\mathbf{AC}^0$-graph is ϵ-close to a graph that is minor-free (ϵ-close means in this context that the symmetric difference of the edge sets of the two graphs is at most ϵNd).

However, it has been shown that testing many interesting graph properties require examining $\Omega(\sqrt{N})$ vertices in the adjacency list model. We note that this need not rule out existence of good algorithms for $\mathbf{AC}^0$ graphs in the sense of Theorem 10. The reason is that the lowerbounds assume an edge oracle that is *black-box*, whereas $\mathbf{AC}^0$ circuits of size n are learnable (albeit only with a very inefficient algorithm) with $\text{poly}(n) = \text{poly}(\log N)$ queries.

Theorem 11. *Every graph property is ϵ-testable (with 2-sided error) using $\text{poly}(\log N/\epsilon)$ evaluations of the $\mathbf{AC}^0$ circuit representing the adjacency list of the d-regular graph.*

Proof. The adjacency list consists of $d \log N$ bits, and each is computed by a circuit of size $\text{poly}(\log N)$ whose output is a single bit. Each of these $d \log N$ circuits can be learnt after evaluating it on $\text{poly}(\log N/\epsilon)$ random points. Namely, if we simply pick the circuit that best fits those $\text{poly}(\log N/\epsilon)$ values, then the standard Chernoff bound calculation shows that this circuit is with high probability (over the choice of the sample points) correct for all but $\epsilon/10d \log N$ fraction of the inputs. Doing this for all $d \log N$ circuits ensures that we get an $\mathbf{AC}^0$ representation of the graph that is $\epsilon/10$-close to the true graph. Having obtained such a representation, we can try all graphs that are $\epsilon/9$-close to our graph and check if any of them have the property. □

4.2 Adapting Sketching Algorithms to Low-Complexity Graphs

Sketching algorithms are given a data matrix M, and using random sampling they construct a *sketch* $S(M)$ of this data which can be used to approximate the value of some specific function f on M.

Here we note that if M is a low-complexity matrix —in other words, $M(i,j)$ can be produced by a low-complexity computation given i, j — many known sketching algorithms produce the sketch $S(M)$ that is also a low-complexity object. Usually this is trivial to see if by "low complexity" we mean $\mathbf{NC}^1$ but sometimes it is true for even $\mathbf{AC}^0$ thanks to the following fact about $\mathbf{AC}^0$.

Theorem 12 (Approximate counting). *Let $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be computable by an $\mathbf{AC}^0$ circuit and define $g\colon \{0,1\}^n \to \{0,1\}$ as $g(x) = \sum_y f(x,y)$. Then for every $\epsilon, c > 0$ there is a $\mathbf{AC}^0$ circuit that given x as input outputs 1 if $g(x) > c$ and 0 if $g(x) < c - \epsilon$, and an arbitrary value otherwise.*

Proof. We give a probabilistic construction. Pick $m = \text{poly}(n)$ random vectors $y_1, y_2, \ldots, y_m$ and for any x try to compute $\sum_{i \le m} f(x, y_i)$. Using our observation in Example 2 there is an $\mathbf{AC}^0$ circuit (even explicit thanks to [11]) that outputs 1 if $\sum_{i \le m} f(x,y_i) > c$ and 0 if $\sum_{i \le m} f(x,y_i) < c - \epsilon/2$. Letting $m > n^2/\epsilon^2$ allows us to conclude via Chernoff bounds that $\sum_i f(x,y_i)$ correctly estimates $g(x) = \sum_y f(x,y)$ for *all* $x \in \{0,1\}^n$. □

As an example we describe how to compute a special sketch of a matrix M called *low rank decomposition*.

Theorem 13. *Let A be an $N \times N$ matrix that is $\mathbf{AC}^0$, and whose entries have absolute value at most $\log N$. Then there is a randomized $\mathrm{poly}(n)$ time algorithm that given $\epsilon > 0$ produces an $\mathbf{AC}^0$ matrix M with all entries of absolute values $O(\log N)$ such that $|A - M|_F \leq |A - M^*|_F + \epsilon N^2$, where $|\cdot|_F$ denotes Frobenius (i.e. sum of squares) norm and M^* is the rank k matrix that minimizes $|A - M^*|_F$.*

Proof. The theorem is implicit in the paper of Frieze, Kannan, Vempala [27] together with some simple observations. For simplicity we assume the matrix is 0/1. The FKV algorithm starts by sampling $s = \mathrm{poly}(k, \epsilon)$ columns according to the probability distribution where column i is picked with probability proportional to its squared ℓ_2 norm. Since we can estimate the squared ℓ_2 norm up to additive error ϵ by Theorem 12 the sampling of columns is easily implemented using standard rejection sampling. Next the FKV algorithm samples s rows using a similar sampling. Let S be the submatrix defined by the sampled columns and W be the final $s \times s$ matrix.

Then it computes the top k singular vectors $u_1, u_2, \ldots, u_k$ of the above $s \times s$ submatrix and scaling factors $c_1, c_2, \ldots, c_k$ (depending only on W) and lets $v_t = c_t S u_t$. The final approximation is $M = A \cdot (\sum_t v_t v_t^T)$.

Now we observe that if suffices to compute the u_i's up to precision $1/\mathrm{poly}(s)$ which is a constant. Thus the u_i's are computable in $\mathbf{AC}^0$ via brute force. Then $v_t = c_t S u_t$ is computable up to precision ϵ/k^2 in $\mathbf{AC}^0$ by Theorem 12, which implies that M is also computable up to precision ϵ in $\mathbf{AC}^0$. □

Acknowledgment. We thank Noga Alon, Avrim Blum, Russell Impagliazzo, Adam Klivans, Rocco Servedio and Commandur Seshadri for useful conversations.

References

1. Mitzenmacher, M.: A brief history of generative models for power law and lognormal distributions. Internet Mathematics 1(2) (2003)
2. Hopcroft, J.: Personal communication (2008)
3. Galperin, H., Wigderson, A.: Succinct representations of graphs. Inf. Control 56(3), 183–198 (1983)
4. Papadimitriou, C.H.: On the complexity of the parity argument and other inefficient proofs of existence. J. Comput. Syst. Sci. 48(3), 498–532 (1994)
5. Razborov, A.A., Rudich, S.: Natural proofs. J. Comput. Syst. Sci. 55(1), 24–35 (1997)
6. Arora, S., Barak, B.: Computational Complexity: A modern approach. Cambridge University Press, Cambridge (2009)
7. Nickel, C.L.M.: Random dot product graphs: A model for social networks. PhD thesis, Johns Hopkins University (2008)
8. Young, S.J., Scheinerman, E.R.: Random dot product graph models for social networks. In: Bonato, A., Chung, F.R.K. (eds.) WAW 2007. LNCS, vol. 4863, pp. 138–149. Springer, Heidelberg (2007)

9. Ajtai, M.: σ_1^1 formulae on finite structures. Annals of Pure Appl. Logic 24 (1983)
10. Furst, M.L., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Mathematical Systems Theory 17(1), 13–27 (1984)
11. Viola, E.: On approximate majority and probabilistic time. In: IEEE Conference on Computational Complexity, pp. 155–168 (2007)
12. Hastad, J.: Almost optimal lower bounds for small depth circuits. In: Randomness and Computation, pp. 6–20. JAI Press (1989)
13. Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bull. AMS (43), 439–561 (2006)
14. Capalbo, M.R., Reingold, O., Vadhan, S.P., Wigderson, A.: Randomness conductors and constant-degree lossless expanders. In: STOC, pp. 659–668 (2002)
15. Alon, N., Schwartz, O., Shapira, A.: An elementary construction of constant-degree expanders. In: SODA, pp. 454–458 (2007)
16. Reingold, O., Vadhan, S.P., Wigderson, A.: Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In: FOCS, pp. 3–13 (2000)
17. Mihail, M., Papadimitriou, C.H.: On the eigenvalue power law. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 254–262. Springer, Heidelberg (2002)
18. Babai, L., Fortnow, L., Lund, L.: Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity 1, 3–40 (1991); Prelim version FOCS 1990
19. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. Journal of the ACM 45(1), 70–122 (1998); Prelim version FOCS 1992
20. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM 45(3), 501–555 (1998); Prelim version FOCS 1992
21. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. Journal of the ACM 43(2), 268–292 (1996); Prelim version FOCS 1991
22. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42(6), 1115–1145 (1995)
23. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. J. ACM 45(4), 653–750 (1998)
24. Alon, N., de la Vega, W.F., Kannan, R., Karpinski, M.: Random sampling and approximation of max-csps. J. Comput. Syst. Sci. 67(2), 212–243 (2003)
25. Alon, N., Fischer, E., Newman, I., Shapira, A.: A combinatorial characterization of the testable graph properties: it's all about regularity. In: STOC, pp. 251–260 (2006)
26. Benjamini, I., Schramm, O., Shapira, A.: Every minor-closed property of sparse graphs is testable. In: STOC, pp. 393–402 (2008)
27. Frieze, A.M., Kannan, R., Vempala, S.: Fast monte-carlo algorithms for finding low-rank approximations. In: FOCS, pp. 370–378 (1998)