Sketching Cuts in Graphs and Hypergraphs

Dmitry Kogan
Weizmann Institute of Science
Rehovot, Israel
dimakogan@gmail.com

Robert Krauthgamer
Weizmann Institute of Science
Rehovot, Israel
robert.krauthgamer@weizmann.ac.il

ABSTRACT

Sketching and streaming algorithms are in the forefront of current research directions for cut problems in graphs. In the streaming model, we show that $(1-\varepsilon)$-approximation for MAX-CUT must use $n^{1-O(\varepsilon)}$ space; moreover, beating 4/5-approximation requires polynomial space. For the sketching model, we show that every r-uniform hypergraph admits a $(1+\varepsilon)$-cut-sparsifier (i.e., a weighted subhypergraph that approximately preserves *all* the cuts) with $O(\varepsilon^{-2}n(r+\log n))$ edges. We also make first steps towards sketching general CSPs (Constraint Satisfaction Problems).

Categories and Subject Descriptors

E.1 [**Data**]: Data Structures; F.2 [**Theory**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms

Keywords

Sketching, Sparsifiers, Streaming, Hypergraphs, Max-Cut

1. INTRODUCTION

The emergence of massive datasets has turned many algorithms impractical, because the standard assumption of having (fast) random access to the input is no longer valid. One example is when data is too large to fit in the main memory (or even on disk) of one machine; another is when the input can be accessed only as a stream, e.g., because its creation rate is so high, that it cannot even be stored in full for further processing. Luckily, the nature of the problems has evolved too, and we may often settle on approximate, rather than exact, solutions.

These situations have led to the rise of new computational paradigms. In the *streaming model* (aka *data-stream*), the input can be accessed only as a stream (i.e., a single pass of sequential access), and the algorithm's space complexity (storage requirement) must be small relative to the stream size. In the *sketching model*, the input is summarized (compressed) into a so-called sketch, which is short yet suffices for further processing without access to the original input. The two models are related – sketches are often useful in the design of streaming algorithms, and vice versa. In particular, lower bounds for sketch-size often imply lower bounds on the space complexity of streaming algorithms.

ITCS'15, January 11–13, 2015, Rehovot, Israel.

ACM 978-1-4503-3333-7/15/01 ...$15.00.
http://dx.doi.org/10.1145/2688073.2688093.

Graph problems.

Recently, the streaming model has seen many exciting developments on *graph problems*, where an input graph $G = (V, E)$ is represented by a stream of edges. The algorithm reads the stream and should then report a solution to a predetermined problem on G, such as graph connectivity or maximum matching; see e.g. the surveys [37, 27]. Throughout it will be convenient to denote $n = |V|$, and to assume edges have weights, given by $w : E \to \mathbb{R}_+$. While initial efforts focused on polylogarithmic-space algorithms, various intractability results have shifted the attention to what is called the *semi-streaming* model, where the algorithm's space complexity is $\tilde{O}(n)$.[1] In general, this storage is not sufficient to record the entire edge-set.

Cuts in graphs is a classical topic that has been studied extensively for more than half a century, and the last two decades have seen a surge of attention turning to the question of their succinct representation. The pioneering work of Benczúr and Karger [5] introduced the notion of *cut sparsifiers*: given an undirected graph $G = (V, E, w)$, a $(1+\varepsilon)$-sparsifier is a (sparse) weighted subgraph $G' = (V, E', w')$ that preserves the value of every cut up to a multiplicative factor $1+\varepsilon$. Formally, this is written as

$$\forall S \subset V, \qquad 1 \le \frac{w'(S,\bar{S})}{w(S,\bar{S})} \le 1+\varepsilon; \tag{1}$$

where $w(S,\bar{S}) = \sum_{e\in E:|e\cap S|=1} w_e$ is used to denote the value of the cut. It is sometimes convenient to replace the left-hand side of (1) with $1-\varepsilon$ or $\frac{1}{1+\varepsilon}$, which affects $\varepsilon \le \frac{1}{2}$ by only a constant factor. In addition to their role in saving storage, sparsifiers are important because they can speed-up graph algorithms whose running time depends on the number of edges. Observe that sparsifiers are a particularly strong form of graph-sketches since on top of retaining the value of all cuts, they hold the additional property of being subgraphs, rather than arbitrary data structures.

[1] We use $\tilde{O}(f)$ to denote $O(f \operatorname{polylog} f)$, which suppresses logarithmic terms.

Ahn and Guha [1] built upon the machinery of cut sparsifiers to present an $\tilde{O}(n/\varepsilon^2)$-space streaming algorithm that can produce a $(1+\varepsilon)$-approximation to all cuts in a graph. Further improvements handle also edge deletions [2, 3, 10], or the stronger notion of spectral sparsification (see [16] and references therein). These results are nearly optimal, due to a space lower bound of $\Omega(n/\varepsilon^2)$ bits for sketching all cuts in a graph [4] (which improves an earlier bound of [1]).

Recent Directions.
These advances on sketching and streaming of graph cuts inspired new questions. One direction is to seek space-efficient streaming algorithms for *specific cut problems*, such as approximating MAX-CUT, rather than *all* cuts. A second direction concerns *hypergraphs*, asking whether cut sparsification, sketching and streaming can be generalized to hypergraphs. Finally, viewing cuts in graphs and hypergraphs as special cases of *constraint satisfaction problems* (CSPs), we ask whether also other CSPs admit sketches. Currently, there is a growing interest in generalizing graph cut problems to broader settings, such as sparsifying general *set systems* using small weighted samples [30], *high-dimensional expander theory* [20], sparsest-cuts in hypergraphs [26, 25], and applications of hypergraph cuts in *networking* [33].

1.1 Our Results

We first address a natural question raised in [14, Question 10], whether the well-known MAX-CUT problem admits approximation strictly better than factor 1/2 by streaming algorithms that use space sublinear in n. Here, MAX-CUT denotes the problem of computing the *value* of a maximum cut in the input graph G (and not the cut itself), since *reporting* a cut requires space $\Omega(n)$ (see Section 2.2 for a short proof). We prove that for every fixed $\varepsilon \in (0, \frac{1}{5})$, streaming algorithms achieving $(1-\varepsilon)$-approximation for MAX-CUT must use $n^{1-O(\varepsilon)}$ space. In fact, even beating 4/5-approximation requires polynomial space. Our result is actually stronger and holds also in a certain sketching model. Previously, it was known that streaming computation of MAX-CUT *exactly* requires $\Omega(n^2)$ bits [36]. Our proof is by reduction from the BOOLEAN HIDDEN HYPERMATCHING problem, and captures the difficulty of distinguishing, under limited communication, whether the graph is a vertex-disjoint union of even-length cycles (in which case the graph is bipartite) or of odd-length cycles (in which case we can bound the maximum cut value). See Section 2 for details.

Second, we study sparsification of cuts in *hypergraphs*, and prove that every r-uniform hypergraph admits a sparsifier (weighted subhypergraph) of size $\tilde{O}(rn/\varepsilon^2)$ that approximates all cuts within factor $1 \pm \varepsilon$. This result immediately implies sketching and streaming algorithms (following [1]). Here, the weight of cut $(S, \bar{S})$ in a hypergraph $H = (V, E, w)$ is the total weight of all hyperedges $e \in E$ that intersect both S and $\bar{S}$.[2] This question was raised by de Carli Silva, Harvey and Sato [7, Corollary 8], who show that every r-uniform hypergraph has a sparsifier of size $O(n)$ that approximates all cuts within factor $\Theta(r^2)$. Along the way, we establish interesting, if not surprising, bounds on the number of approximately minimum cuts in hypergraphs. Technically, this is our most substantial contribution, see Section 3 for details.

Finally, as a step towards understanding the sketching complexity of a wider range of CSPs, we show that every k-SAT instance on n variables admits a sketch of size $\tilde{O}(kn/\varepsilon^2)$ that can be used to $(1+\varepsilon)$-approximate the value of all truth assignments. We prove this result in Section 3.3 by reducing it to hypergraph sparsification. We remark that sketching of CNF formulae was studied in a different setting, where some computational-complexity assumptions were used in [8] to preclude a significant size-reduction that *preserves the satisfiability* of the formula. Our sparsification result differs in that it *approximately preserves the value* of every assignment.

1.2 Related Work

Independently of our work, Kapralov, Khanna and Sudan [15] study the same problem of approximating MAX-CUT in the streaming model. They first prove that for every fixed $\varepsilon > 0$, streaming algorithms achieving $(1-\varepsilon)$-approximation for MAX-CUT must use $n^{1-O(\varepsilon)}$ space. (This is similar to our Theorem 2.1.) They then make significant further progress, and show that achieving an approximation ratio strictly better than (the trivial) 1/2 requires $\tilde{\Omega}(\sqrt{n})$ space. In fact, this result holds even if the edges of the graph are presented in a random (rather than adversarial) order.

Hypergraph sparsifiers of size $O(n^2/\varepsilon^2)$ are implied by a result of Newman and Rabinovich [30] for the following problem of approximating measures on *set systems*. Let $\mathcal{F}$ be a set system over a finite set X, let μ be a measure on X (which naturally extends to a measure on $\mathcal{F}$) and let $\varepsilon \in (0, 1)$. The goal is to construct a measure μ^*, supported on a *small* subset of X, such that the extensions of μ and of μ^* to $\mathcal{F}$ approximate each other, i.e., $\forall S \in \mathcal{F}$, $\mu^*(S) \in (1 \pm \varepsilon)\mu(S)$. They show a construction in which the support size of μ^* can be bounded by a structural parameter of $\mathcal{F}$ called *triangular rank* and denoted $\mathrm{trk}(\mathcal{F})$. Specifically, in their construction $|\operatorname{supp}(\mu^*)| = O(\mathrm{trk}(\mathcal{F}) \cdot \log |\mathcal{F}|/\varepsilon^2)$. They also define *splitting set systems* – a special class of set systems in which $X, \mathcal{F} \subset 2^V$ are two families of subsets of some underlying set V. For splitting systems they prove [30, Claim 4.1] that $\mathrm{trk}(\mathcal{F}) \leq |V| - 1$. The archetype of splitting set systems is in fact graph and hypergraph cuts, where V is the set of vertices, X is the set of hyperedges, and $\mathcal{F}$ is the set of cuts. Therefore their construction implies hypergraph sparsifiers of size $O(n^2/\varepsilon^2)$.[3]

More broadly, the general theme of *graph compression* – succinctly representing a graph while preserving some of its combinatorial properties – is studied extensively in the literature, with many examples of various flavors. A classical example is a *Gomory-Hu tree* [12] which is a weighted tree that represents the minimum $s-t$ cut values for all pairs of vertices in an input graph. Another notable example is the notion of *graph spanners* (defined by Peleg and Schäffer [31])

[2]Another possible definition, see [7, Corollary 7], is $\sum_{e\in E} w_e \cdot |e \cap S| \cdot |e \cap \bar{S}|$. The latter definition seems technically easier for sparsification, although both generalize the case of ordinary graphs ($r = 2$).

[3]Their argument does not imply a bound stronger than $O(n^2/\varepsilon^2)$ even for r-uniform hypergraphs. They show that $\mathrm{trk}(\mathcal{F}) \geq \mathrm{VCdim}(\mathcal{F})$ where $\mathrm{VCdim}(\mathcal{F})$ is the Vapnik-Chervonenkis dimension of the set system. Even when $\mathcal{F}$ is the family of cuts of a (2-uniform) graph, it holds $\mathrm{VCdim}(\mathcal{F}) \geq n - 1$, which follows, for example, from considering a path on n vertices, and observing that the set of $n - 1$ edges is shattered by the set of cuts.

– spar subgraphs that approximately preserve the shortest path distances between all pairs of vertices in the graph. The related notion of *distance oracles* (introduced by Thorup and Zwick [32]) deals with arbitrary data structures, rather than subgraphs, that can approximate the distances between all pairs of vertices, with emphasis on achieving low space and very fast query time. Other models aim to preserve the combinatorial property of interest only with respect to some predetermined (small) subset of the vertices, called *terminals*. For example, Moitra [28] introduced the notion of *vertex sparsifiers* – graphs (not necessarily subgraphs) that approximately preserve the values of minimum cuts separating any partition of the terminals. In a subsequent work, Leighton and Moitra [23] extended this definition and introduced *flow sparsifiers* – graphs that approximately preserve the congestion of every multicommodity flow with endpoints supported on the set of terminals.

2. SKETCHING MAX-CUT

The classical MAX-CUT problem is perhaps the simplest MAX-CSP problem. Thus, it has been studied extensively, leading to fundamental results both in approximation algorithms [11] and in hardness of approximation [21]. It is thus natural to study MAX-CUT also in the streaming model. As mentioned above, preserving the values of *all* cuts in a graph requires linear space even if only approximate values are required [1, 4], which raises the question whether smaller space suffices to approximate only the MAX-CUT value (as mentioned above, it is natural to require the algorithm to report only the value of the cut as opposed to the cut itself, see Section 2.2).

Sketching all cuts in a graph clearly preserves also the maximum-cut value, and thus an $\tilde{O}\left(\frac{n}{\varepsilon^2}\right)$ space streaming algorithm for $(1-\varepsilon)$-approximation of MAX-CUT follows immediately from [1]. Yet since the maximum cut value is always $\Omega(m)$, where m is the total number (or weight) of all edges, a similar result can be obtained more easily by uniform sampling (achieving εm additive approximation for all cuts) [35, Theorem 21]. The latter approach has the additional advantage that it immediately extends to hypergraphs.

It turns out that this relatively straightforward approach is not far from optimal, as we prove that streaming algorithms for $(1-\varepsilon)$-approximation of MAX-CUT require $n^{1-O(\varepsilon)}$ space.

THEOREM 2.1. *Fix a constant $\varepsilon \in (0, \frac{1}{5})$. Then every (randomized) streaming algorithm that computes a $(1-\varepsilon)$-approximation of the* MAX-CUT *value in n-vertex graphs requires space $\Omega(n^{1-1/t})$ for $t = \lfloor \frac{1}{2\varepsilon} - \frac{1}{2} \rfloor$, which in particular means space $n^{1-O(\varepsilon)}$.*

To prove this result, we consider the somewhat stronger *one-way two-party communication model*, where instead of arriving as a stream, the set of edges of a graph is split between two parties, who engage in a communication protocol to compute (approximately) the graph's maximum-cut value. Since a lower bound in this model immediately translates to the original streaming model, the theorem above follows immediately from Theorem 2.3 below.

2.1 Proving Theorem 2.1

DEFINITION 2.2 (MAX-CUT$^\varepsilon$). *Let $G = (V, E_A \cup E_B)$ be an input graph on $|V| = n$ vertices with maximum cut value[4] c^*, and $\varepsilon > 0$ some small constant.* MAX-CUT$^\varepsilon$ *is a two-player communication game where Alice and Bob receive the edges E_A and E_B respectively and need to output a value c' such that with high probability $(1-\varepsilon)c^* \le c' \le c^*$.*

THEOREM 2.3. *Fix a constant $\varepsilon \in (0, \frac{1}{5})$. Then the randomized one-way communication complexity of* MAX-CUT$^\varepsilon$ *is $\Omega(n^{1-1/t})$ for $t = \lfloor \frac{1}{2\varepsilon} - \frac{1}{2} \rfloor$.*

The proof is by a reduction from the following communication problem studied in [34].

DEFINITION 2.4 (BHH_n^t). *The* BOOLEAN HIDDEN HYPERMATCHING *problem is a communication complexity problem where*

- *Alice gets a boolean vector $x \in \{0,1\}^n$ where $n = 2kt$ for some integer k,*
- *Bob gets a perfect hypermatching M on n vertices where each edge has t vertices and a boolean vector w of length n/t.*

Let Mx denote the length-n/t boolean vector

$$\left(\bigoplus_{1 \le i \le t} x_{M_{1,i}}, \ldots, \bigoplus_{1 \le i \le t} x_{M_{n/t,i}} \right)$$

where $(M_{1,1}, \ldots, M_{1,t}), \ldots, (M_{n/t,1}, \ldots, M_{n/t,t})$ are the edges of M. It is promised that either $Mx \oplus w = 1^{n/t}$ or $Mx \oplus w = 0^{n/t}$. The problem is to return 1 in the former case, and to return 0 in the latter.

LEMMA 2.5 ([34, THEOREM 2.1]). *The randomized one-way communication complexity of BHH_n^t where $n = 2kt$ for some integer $k \ge 1$ is $\Omega(n^{1-1/t})$.*

PROOF OF THEOREM 2.3. We show a reduction from BHH_n^t to MAX-CUT$^\varepsilon$. Consider an instance (x, M, w) of the BHH_n^t problem: Alice gets $x \in \{0,1\}^n$, and Bob gets a perfect hypermatching M and a vector $w \in \{0,1\}^{n/t}$.

We construct a graph G for the MAX-CUT$^\varepsilon$ problem as follows (see Figure 2.1 for an example):

- The vertices of G are $V = \{v_i\}_{i=1}^{2n} \cup \{u_i\}_{i=1}^{2n} \cup \{w_i\}_{i=1}^{2n/t}$.
- The edges E_A given to Alice are: for every $i \in [n]$, if $x_i = 0$, Alice is given two "parallel" edges (u_{2i-1}, v_{2i-1}), (u_{2i}, v_{2i}); if $x_i = 1$, Alice is given two "cross" edges $(u_{2i-1}, v_{2i}), (u_{2i}, v_{2i-1})$.
- The edges E_B given to Bob are: for each hyperedge $M_j = (i_1, i_2, \ldots, i_t) \in M$ (where the order is fixed arbitrarily):
  - For $k = 1, 2, \ldots, t-1$, Bob is given $(u_{2i_k-1}, v_{2i_{k+1}-1})$ and $(u_{2i_k}, v_{2i_{k+1}})$
  - For $k = t$, Bob is given (u_{2i_t}, w_{2j}) and (v_{2i_t-1}, w_{2j-1});
  - If $w_j = 0$ Bob is given two "parallel" edges (w_{2j}, v_{2i_1}) and (w_{2j-1}, v_{2i_1-1});
    if $w_j = 1$, Bob is given two "cross" edges (w_{2j}, v_{2i_1-1}) and (w_{2j-1}, v_{2i_1})

[4] For the proof of the lower bound it suffices to restrict our attention to unweighted graphs, with all edges having unit weight.

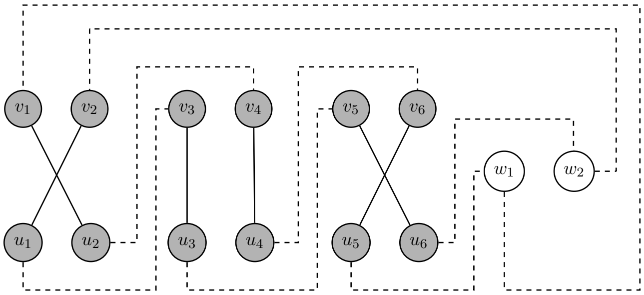

Figure 1: An example of a gadget constructed in the proof of Theorem 2.3 for $t = 3$, a matching M that contains the hyperedge $M_1 = (1, 2, 3)$, $x_1 = 1$, $x_2 = 0$, $x_3 = 1$ and $w_1 = 0$. The result is two paths of length 7. Alice's and Bob's edges are shown as solid and dashed lines respectively.

By definition, for each $j \in [n/t]$, if $M_j = (i_1, i_2, \ldots, i_t) \in M$ and $(Mx)_j \oplus w_j = 0$ we have $\sum_{k=1}^{t} x_{i_k} \oplus w_j = 0$. Since the number of 1 bits in the latter sum is even, when we start traversing from u_{2i_1} we go through an even number of "cross" edges and complete a cycle of length $2t + 1$. Similarly when starting our traversal at u_{2i_1-1} we complete a different cycle of the same length. Therefore if (x, M, w) is a 0-instance the graph consists of $\frac{2n}{t}$ paths of (odd) length $2t + 1$ each. Therefore the maximum cut value is $c_0^* = 2t \cdot \frac{2n}{t} = 4n$.

On the other hand if $(Mx)_j \oplus w_j = 1$, starting our traversal at u_{2i_1-1}, we pass an odd number of cross edges and end up at u_{2i_1}, from where we once again pass an odd number of cross edges, to complete a cycle of total length $2 \cdot (2t + 1) = 4t + 2$ that ends back in u_{2i_1-1}. Therefore, if (x, M, w) is a 1-instance the graph consists of n/t paths of (even) length $4t + 2$ each. The maximum cut value in this case is $c_1^* = 4n + 2\frac{n}{t}$.

Observing that $c_0^*/c_1^* = \frac{4n}{4n+2n/t} = \frac{2t}{2t+1} < 1 - \varepsilon$, we conclude that a randomized one-way protocol for MAX-CUT$^\varepsilon$ (on input size $n' = 4n + n/t = O(n)$) gives a randomized one-way protocol for BHH_n^t. By Lemma 2.5 the Theorem follows. $\square$

PROOF OF THEOREM 2.3. Any streaming algorithm for MAX-CUT$^\varepsilon$ leads to a one-way communication protocol in the two party setting. Moreover the communication complexity of this protocol is exactly the space complexity of the streaming algorithm. Hence by Theorem 2.3 the streaming space complexity is at least as high as the one way randomized communication complexity. $\square$

2.2 Reporting a Vertex-Bipartition (rather than a value)

We show a simple $\Omega(n)$ space lower bound for reporting a vertex-bipartition that gives an approximate maximum cut.

PROPOSITION 2.6. *Let $\varepsilon \in (0, \frac{1}{2})$ be some small constant. Suppose* **sk** *is a polynomial time sketching algorithm that outputs at most $s = s(n, \varepsilon)$ bits, and* **est** *is an estimation algorithm, such that together, for every n-vertex graph G, (with high probability) they output a vertex-bipartition that gives an approximately maximum cut; i.e.,* $\mathbf{est}(\mathbf{sk}(G)) = S$ *such that $w(S, \bar{S}) \geq (1 - \varepsilon)\tilde{w}$ where $\tilde{w}$ is the maximum cut in G. Then $s \geq \Omega_\varepsilon(n)$.*

PROOF. Let $\mathcal{C} \subset \{0,1\}^n$ be a binary error-correcting code of size $|\mathcal{C}| = 2^{\Omega(n)}$ with relative distance ε. We may assume w.l.o.g. that for every $x \in \mathcal{C}$ the hamming weight $|x|$ is exactly $n/2$ (for instance by taking $\mathcal{C}' = \{x\bar{x} : x \in \mathcal{C}\}$ where $\bar{x}$ denotes the bitwise negation of x), and that there are no $x, y \in \mathcal{C}$ such that $|x - \bar{y}| \leq \frac{\varepsilon}{2} n$ (since for every $x \in \mathcal{C}$ there could be at most one "bad" y, and we can discard one codeword out of every such pair).

Fix a codeword $x \in \{0,1\}^n$ and consider the complete bipartite graph $G_x = (V, E)$ where $V = [n]$ and $E = \{(i, j) : x_i = 0 \wedge x_j = 1\}$. The maximum cut value in G_x is obviously $\tilde{w} = n^2/4$. Let $y \in \{0,1\}^n$ such that $\frac{1}{2}\varepsilon n \leq |x - y| \leq \frac{n}{2}$. Identifying x, y with subsets $S_x, S_y \subseteq [n]$, and using the fact that $|S_x \triangle S_y| = |x - y| \geq \frac{1}{2}\varepsilon n$, the value of the cut $(S_y, \bar{S}_y)$ in G_x is

$$|E(S_y, \bar{S}_y)| = \frac{n^2}{4} - |S_x \setminus S_y| \left(\frac{n}{2} - |S_y \setminus S_x|\right) - |S_y \setminus S_x| \left(\frac{n}{2} - |S_x \setminus S_y|\right) < (1 - \Omega(\varepsilon))\frac{n^2}{4}.$$

Let $\mathbf{sk}(G_x)$ be the sketch of G_x, and let $\mathbf{est}(\mathbf{sk}(G_x)) = S$ be the output of the estimation algorithm on the sketch of G_x. Therefore if the sketch succeeds (which by our assumption happens with high probability) and the cut $(S, \bar{S})$ has value at least $(1 - \Omega(\varepsilon))\tilde{w}$, then by the preceding argument the corresponding vector x_S is of relative hamming distance smaller than $\frac{\varepsilon}{2}$ from x and then one can decode x from S.[5] By standard arguments from information theory, the size s of a sketch that succeeds with high probability must be at least $\Omega(\log |C|) = \Omega_\varepsilon(n)$. $\square$

2.3 2/3-Approximation of Max-Cut in the Two Party Model

While [15] have recently shown that a polynomial number of bits is necessary for any non-trivial (i.e., strictly better than 1/2) approximation of MAX-CUT *in the streaming model*, we remark that a 2/3-approximation communication protocol that uses only a logarithmic number of bits exists *in the one-way two-party model*. In the latter model, the problem of giving a $(1 - \varepsilon)$-approximation of the maximum cut exhibits an exponential gap in the communication complexity between the case of $\varepsilon = 1/5$, where we have shown that a polynomial number of bits is necessary, and the case $\varepsilon = 1/3$, for which logarithmically many bits suffice, as follows from the following simple protocol.

PROPOSITION 2.7. *Let $G = (V, E_A \uplus E_B)$ be an input graph on $|V| = n$ vertices. Let w_A and w_B be the maximum cut values in $G_A = (V, E_A)$ and $G_B = (V, E_B)$ respectively. Then the maximum cut value w in G satisfies*

$$\tfrac{2}{3}(w_A + w_B) \leq w \leq w_A + w_B.$$

PROOF. Consider cuts $C_A, C_B : V \to \{0,1\}$ such that $w(C_A) = w_A$ and $w(C_B) = w_B$. Let $C : V \to \{0,1\}$ be a cut chosen uniformly at random from $\{C_A, C_B, C_A \oplus C_B\}$ where we define $(C_A \oplus C_B)(v) = C_A(v) + C_B(v) \pmod 2$ for every $v \in V$. For an edge $e = (u, v) \in C_A$, either $C_B(u) \oplus C_B(v) = 1$ or $(C_A \oplus C_B)(u) + (C_A \oplus C_B)(v) =$

[5]Since the cuts $(\bar{S}, S)$ has the same value as $(S, \bar{S})$, the vector x_S can actually be ε-close to $\bar{x}$, but by taking our code to have no codeword being close to the negation of another codeword we can always try decoding both x_S and $\bar{x}_S$.

$(C_A(u) + C_A(v)) + (C_B(u) + C_B(v)) = 1 + 0 = 1$. Either way $\Pr_{C \in_R \{C_A, C_B, C_A \oplus C_B\}}[e \in C] = \frac{2}{3}$. Similarly the same holds for an edge $e \in C_B$. Therefore by linearity of expectation a random cut in $\{C_A, C_B, C_A \oplus C_B\}$ has value at least $\frac{2}{3}(w_A + w_B)$. The second inequality is trivial. $\square$

Corollary 2.8. *The one-way communication complexity of* Max-Cut$^{1/3}$ *is* $O(\log n)$.

Proof. Alice uses her input to compute the value w_A and sends it to Bob. Bob uses his input to compute the value w_B and outputs $\frac{2}{3}(w_A + w_B)$. $\square$

3. SKETCHING CUTS IN HYPERGRAPHS

In their celebrated work, Benczúr and Karger [5] (with further improvements and simplifications in [18, 19, 6]) showed an effective method to sketch the values of *all* the cuts of an undirected (weighted) graph $G = (V, E, w)$ by constructing a *cut-sparsifier*, which is a subgraph with different edge weights, that contains only $\tilde{O}\left(n/\varepsilon^2\right)$ edges, and approximates the weight of every cut in G up to multiplicative factor $1 \pm \varepsilon$. We generalize the ideas of Benczúr and Karger to obtain cut-sparsifiers of hypergraphs, as stated below. Such sparsifiers (and sketches) can be computed by streaming algorithms that use $\tilde{O}(rn)$ space for r-uniform hypergraphs using known techniques (of [1] and subsequent work). Throughout this work we allow r-uniform hypergraphs to contain also hyperedges with less than r endpoints (for instance by allowing duplicate vertices in the same hyperedge).

Theorem 3.1. *For every r-uniform hypergraph $H = (V, E, w)$ and an error parameter $\varepsilon \in (0, 1)$, there is a subhypergraph H_ε (with different edge weights) such that:*

- *H_ε has $O\left(n(r + \log n)/\varepsilon^2\right)$ hyperedges.*
- *The weight of every cut in H_ε is within $1 \pm \varepsilon$ times the weight of the corresponding cut in H.*

Furthermore, H can be constructed in $O(mn^2)$ time where $m = |E|$ is the number of hyperedges in the original hypergraph.

A key combinatorial property exploited in the Benczúr-Karger analysis is an upper bound on the number of cuts of near-minimum weight [17]. It asserts that the number of minimum-weight cuts in an n-vertex graph is at most n^2 (which had been previously shown by [24] and [9]), and more generally, there are at most $n^{2\alpha}$ cuts whose weight is at most $\alpha \geq 1$ times the minimum (more refined bounds for $\alpha = 4/3$ and $\alpha = 3/2$ appear in [29] and [13] respectively). These bounds are known to be tight (e.g., for an n-cycle). Correctly generalizing this property to r-uniform hypergraphs appears to be a nontrivial question. A fairly simple analysis generalizes the latter bound to $n^{r\alpha}$, but using new ideas, we manage to obtain the following tighter bound.

Theorem 3.2. *Let $H = (V, E, w)$ be a weighted r-uniform hypergraph with n vertices and minimum cut value $\hat{w}$. Then for every half-integer $\alpha \geq 1$, the number of cuts in H of weight at most $\alpha\hat{w}$ is at most $O(2^{\alpha r} n^{2\alpha})$.*

We prove this "cut-counting" bound in Section 3.1. With this bound at hand, we prove Theorem 3.1 similarly to the original proof of [5] for graphs, as outlined in Section 3.2.

Cuts in hypergraphs are perhaps one of the simplest examples of CSPs, which we now formally define.

Definition 3.3. *A* Constraint Satisfaction Problem *is a quintuple (Σ, X, P, C, w) where:*

- *Σ is a finite alphabet,*
- *$X = \{x_1, \ldots, x_n\}$ is a set of variables taking their values from Σ,*
- *$P = \{P_1, \ldots, P_k\}$ is set of r-ary predicates,*
- *$C = \{C_1, \ldots, C_m\}$ is a set of constraints, where each constraint C_i consists of one of the predicates P_j and a sequence of variables $(x_{ij})_{j=1}^r$ from X,*
- *$w : C \to \mathbb{R}_+$ is a weight function on the set of constraints.*

For example, in the case of cuts in hypergraphs, the vertices are variables over the binary alphabet, and the hyperedges are constraints defined by the predicate Not-All-Equal. A natural question is whether general CSPs admit sketches as well, where a sketch should provide an approximation to the value of every assignment to the CSP (as usual, the value of an assignment is the total weight of constraints it satisfies). Specifically we think of both Σ and P as being of constant size, and are interested in the dependence on n and r. Although we are still far from answering this question in full generality, we prove that for the well-known SAT problem, sketching is indeed possible.

Theorem 3.4. *For every error parameter $\varepsilon \in (0, 1)$, there is a polynomial time sketching algorithm that produces from an r-CNF formula Φ on n variables a sketch of size $\tilde{O}(rn/\varepsilon^2)$, that can be used to $(1 \pm \varepsilon)$-approximate the value of every assignment to Φ.*

3.1 Counting Near-Minimum Cuts in Hypergraphs

In this subsection we prove our upper bound on the number of near-minimum cuts (Theorem 3.2). We generalize Karger's min-cut algorithm [17] to hypergraphs, and then show that its probability to output any individual cut is not small (Theorem 3.6), which immediately yields a bound on the number of distinct cuts. Finally, we show that the exponential dependence on r in Theorem 3.2 is necessary (Section 3.1.3).

3.1.1 A Randomized Contraction Algorithm

Consider the following generalization of Karger's contraction algorithm [17] to hypergraphs.

Algorithm 3.5 ContractHypergraph

Input: an r-uniform weighted hypergraph $H = (V, E, w)$
a parameter $\alpha > 1$
Output: a cut $C = (S, V \setminus S)$

```
H' ← H
while |V(H')| > αr do
    e ← random hyperedge in H' with probability proportional to its weight
    contract e by merging all its endpoints and removing self-loops[6]
C' ← random cut in H' (bipartition of V(H'))
return the cut C in H induced[7] by the cut C'
```

[6] Self-loops refers to hyperedges that contain only a single vertex. Note also that the cardinality of an edge can only decrease as a result of contractions.

[7] Since after the sequence of contractions, each vertex in $V(H)$ corresponds to exactly one vertex in $V(H')$, a vertex

THEOREM 3.6. *Let $H = (V, E, w)$ be a weighted r-uniform hypergraph with minimum cut value $\hat{w}$, let $n = |V|$, and let $\alpha \geq 1$ be some half-integer. Fix $C = (S, V \setminus S)$ to be some cut in H of weight at most $\alpha\hat{w}$. Then Algorithm 3.5 outputs the cut C with probability at least $\frac{Q_{n,r,\alpha}}{2^{\alpha r-1}-1}$ for*

$$Q_{n,r,\alpha} = \frac{2\alpha+1}{(r+1)} \binom{n-\alpha(r-2)}{2\alpha}^{-1}$$

if $\alpha r < n$ and $Q_{n,r,\alpha} = 1$ otherwise.

Since Theorem 3.6 gives a lower bound on the probability to output a specific cut (of certain weight), and different cuts correspond to disjoint events, the theorem implies that the number of cuts of weight at most $\alpha\hat{w}$ is at most

$$\frac{2^{\alpha r-1}-1}{Q_{n,r,\alpha}} \leq \frac{\left(2^{\alpha r-1}-1\right)(r+1)}{2\alpha+1}\binom{n}{2\alpha} = O\left(2^{\alpha r} n^{2\alpha}\right),$$

proving Theorem 3.2.

PROOF. Fix $C = (S, V \setminus S)$ to be some cut of weight $\alpha\hat{w}$ in H. For $t \in [n]$, denote by I_t the iteration of the algorithm where H' contains t vertices. Since a contraction of a hyperedge may reduce the number of vertices by anywhere between 1 and $r-1$, in a specific execution of the algorithm, not necessarily all the $\{I_t\}_{t=1}^n$ occur. Similarly, let the random variable E_t be the edge contracted in iteration t.

We say that an iteration I_t is *bad* if $E_t \in C$ (i.e., the hyperedge contains vertices from both S and $V \setminus S$). Otherwise, we say it is *good* (including iterations that do not occur in the specific execution such as $I_1, \ldots, I_{\alpha r}$). For any fixed $e_n, \ldots, e_{t+1} \in E$ define

$$q_t(e_n, \ldots, e_{t+1}) = \Pr\left[I_t, \ldots, I_1 \text{ are good} | E_n = e_n, \ldots, E_{t+1} = e_{t+1}\right]$$

Note that q_n is simply the probability that all iterations of the algorithm are good i.e., no edge of the cut C is contracted. When that happens, in step 5 of the algorithm, there exists a cut C' in H' that corresponds to the cut C in H. Since at that stage, there are at most αr vertices in H', the probability of choosing C' is at least $\frac{1}{2^{\alpha r-1}-1}$. Hence the overall probability of outputting cut C is at least $q_n \cdot \frac{1}{2^{\alpha r-1}-1}$. We thus need to give a lower bound on q_n. To this end we prove below the following lemma.

LEMMA 3.7. *$q_t(e_n, \ldots, e_{t+1}) \geq Q_{t,r,\alpha}$ for every $t \in [n]$, and every $e_n, \ldots, e_{t+1} \in E \setminus C$.*

Using the lemma for $t = n$ bounds the overall probability of outputting cut C and proves Theorem 3.6. $\square$

3.1.2 Proof of Lemma 3.7

We prove the lemma by (complete) induction on t. For the base case, note that $q_t(e_n, \ldots, e_{t+1}) = 1$ for $1 \leq t \leq \alpha r$ since no contractions take place in those iterations.

For the general case, fix an iteration I_t and from now on, condition on some set of values $E_n = e_n, \ldots, E_{t+1} = e_{t+1}$. All probabilities henceforth are thus conditioned, and for brevity we omit it from our notation. Observe that depending on the cardinality of E_t, the next iteration (*after* iteration I_t) may be one of $I_{t-1}, \ldots, I_{t-r+1}$. Let $p_i = \Pr[|E_t| = i]$ and let $y_i = \Pr[|E_t| = i \wedge E_t \in C]$.[8,9] We can now write a recurrence relation:

$$\begin{aligned}
&q_t(e_n, \ldots, e_{t+1}) = \\
&= \Pr\Big[I_t, \ldots, I_1 \text{ are good} \mid E_n = e_n, \ldots, E_{t+1} = e_{t+1}\Big] \\
&= \sum_{i=2}^{r} \Pr\Big[|E_t| = i \wedge E_t \notin C\Big] \\
&\qquad \cdot \Pr\Big[I_{t-i+1}, \ldots, I_1 \text{ are good} \mid |E_t| = i, E_t \notin C\Big] \\
&= \sum_{i=2}^{r} (p_i - y_i) \mathbb{E}_{E_t}\Big[q_{t-i+1}(e_n, \ldots, e_{t+1}, E_t) \mid |E_t| = i, E_t \notin C\Big] \\
&\geq \sum_{i=2}^{r} (p_i - y_i) Q_{t-i+1,r,\alpha}.
\end{aligned}$$

For $i = 2, \ldots, r$ let $W_i = \sum_{e' \in H' : |e'| = i} w(e')$ be the total weight of hyperedges in H' of cardinality i (at iteration t) and let $W = \sum_{i=2}^{r} W_i$ be the total weight in H'.

Observe that $p_i = \frac{W_i}{W}$ since E_t is chosen with probability proportional to the hyperedge's weight, and $\sum_{v \in V'} deg(v) = \sum_{i=2}^{r} i \cdot W_i$ since a hyperedge of cardinality i is counted i times on the left-hand side. By averaging, there exists a vertex $v \in V(H')$ such that $deg(v) \leq \frac{1}{t}\sum_{i=2}^{r} i \cdot W_i$, and since it induces a cut in H whose weight is exactly $deg(v)$, we obtain that $\hat{w} \leq deg(v) \leq \frac{1}{t}\sum_{i=2}^{r} i \cdot W_i$.

Next note that

$$\sum_{i=2}^{r} y_i = \Pr[E_t \in C] \leq \frac{\alpha\hat{w}}{W} \leq \tfrac{\alpha}{t}\sum_{i=2}^{r} i \cdot \frac{W_i}{W} = \tfrac{\alpha}{t}\sum_{i=2}^{r} i \cdot p_i,$$

where the first inequality uses the conditioning on all previous iterations being good, which means that all hyperedges in C have survived in H', and thus $w_H(C) = w_{H'}(C)$.

Altogether, to prove the lemma it suffices to show that the value of the following linear program is at least $Q_{t,r,\alpha}$. From now on we omit the subscripts r and α, denoting $Q_t = Q_{t,r,\alpha}$.

$$\begin{aligned}
\text{minimize } & \sum_{i=2}^{r} (p_i - y_i) Q_{t-i+1} \\
\text{subject to } & 0 \leq y_i \leq p_i \qquad \forall i = 2, \ldots, r \\
& \sum_{i=2}^{r} p_i = 1 \\
& \sum_{i=2}^{r} y_i \leq \tfrac{\alpha}{t}\sum_{i=2}^{r} i \cdot p_i.
\end{aligned}$$

First observe that the last constraint implies

$$\sum_{i=2}^{r} y_i \leq \tfrac{\alpha}{t}\sum_{i=2}^{r} i \cdot p_i \leq \tfrac{\alpha}{t}\sum_{i=2}^{r} r \cdot p_i = \tfrac{\alpha r}{t}\sum_{i=2}^{r} p_i < \sum_{i=2}^{r} p_i, \quad (2)$$

which means that in every *feasible* solution there is always some $y_i < p_i$. This implies that in every *optimal* solution, the last constraint is tight, since otherwise increasing such a

bipartition in H' naturally induces a vertex bipartition in H.

[8]Since not all iterations occur in all executions, it might be the case that *no* edge is contracted in iteration t. In that case iteration t is good, and hence by the induction hypothesis the claim holds.

[9]Note that $|e|$ refers to the edge's cardinality, whereas $w(e_i)$ refers to its weight.

y_i will decrease the value of the solution, without violating any of the other constraints.

It is easy to see that this linear program is both feasible and bounded, and therefore has an optimal solution that is basic (i.e., a vertex of the polytope). The dimension of the linear program (i.e., the number of variables) is $2r-2$, and thus in a basic feasible solution (at least) $2r-2$ of the $2r$ constrains must be tight. Therefore there are at most 2 loose (i.e., not tight) constraints among the $2r-2$ constraints $0 \le y_i \le p_i$, meaning there are at most 2 indices i, j such that $p_i \neq 0$. We proceed by analyzing the four possible cases:

- $0 < y_i = p_i$ and $0 < y_j = p_j$. This case is not possible, since that would have implied $\sum_{i=2}^{r} y_i = \sum_{i=2}^{r} p_i$, contradicting (2).
- $0 = y_i < p_i$ and $0 = y_j < p_j$. This case is also not possible since that would have implied $\sum_{i=2}^{r} y_i = 0$, contradicting the tightness of the last constraint in an optimal solution.
- $0 = y_i < p_i$ and $0 < y_j = p_j$. Since all other $p_\ell = 0$, the other LP constraints become

$$p_i + p_j = 1$$
$$0 + p_j = y_i + y_j = \tfrac{\alpha}{t}(ip_i + jp_j).$$

Solving the two equations we obtain:

$$\begin{aligned} \mathrm{LP} &= \left(1 - \tfrac{\alpha i}{t+\alpha i-\alpha j}\right) Q_{t-i+1} \\ &\ge \left(1 - \tfrac{\alpha i}{t+\alpha i-\alpha r}\right) Q_{t-i+1} = \tfrac{t-\alpha r}{t+\alpha i-\alpha r} Q_{t-i+1}. \quad (3) \end{aligned}$$

To use the induction hypothesis, we distinguish between two cases:

1. $t - i + 1 \ge \alpha r$, in which case it is thus sufficient to prove the following claim.

Claim 3.8. *For every half-integer $\alpha \ge 1$ and integers $r \ge i \ge 2$ and $t \ge \alpha r + i - 1$, it holds $\frac{Q_{t-i+1,r,\alpha}}{Q_{t,r,\alpha}} \ge \frac{t+\alpha i-\alpha r}{t-\alpha r}$.*

Proof. Recall that $Q_t = \frac{2\alpha+1}{(r+1)}\binom{t-\alpha(r-2)}{2\alpha}^{-1}$ and denote $t' = t - \alpha r$. Then

$$\begin{aligned} \frac{Q_{t-i+1,r,\alpha}}{Q_{t,r,\alpha}} &= \frac{\frac{2\alpha+1}{(r+1)}\binom{t'+2\alpha}{2\alpha}}{\frac{2\alpha+1}{(r+1)}\binom{t'-i+2\alpha+1}{2\alpha}} \\ &= \frac{(t'+2\alpha)\cdots(t'+1)}{(t'+2\alpha-i+1)\cdots(t'+1-i+1)} \\ &= \frac{(t'+2\alpha)\cdots(t'+2\alpha-i+2)}{t'\cdots(t'-i+2)} \\ &= \left(1+\frac{2\alpha}{t'}\right)\cdots\left(1+\frac{2\alpha}{t'-i+2}\right) \\ &\ge \left(1+\frac{2\alpha}{t'}\right)^{i-1} \\ &\ge 1 + \frac{2\alpha(i-1)}{t'} \\ &\ge 1 + \frac{\alpha i}{t'} = \frac{t+\alpha i-\alpha r}{t-\alpha r}. \quad \square \end{aligned}$$

2. $t - i + 1 < \alpha r$, in which case $Q_{t-i+1} = 1$. Here we get

$$\begin{aligned} \mathrm{LP} &\ge 1 - \tfrac{\alpha i}{t-\alpha r+\alpha i} \ge 1 - \tfrac{\alpha i}{\alpha i+1} = \tfrac{1}{\alpha i+1} \ge \tfrac{1}{\alpha r+1} \\ &\ge \tfrac{2\alpha+1}{(r+1)\binom{t-\alpha(r-2)}{2\alpha}} = Q_t, \end{aligned}$$

where the last inequality follows from the fact that $t - \alpha(r-2) \ge \alpha r + 1 - \alpha(r-2) \ge 2\alpha + 1$.

- $0 < y_i < p_i$ and $0 = y_j = p_j$. In this case $p_i = 1$, $y_i = \frac{\alpha i}{t}$, and therefore

$$\mathrm{LP} = \left(1 - \tfrac{\alpha i}{t}\right) Q_{t-i+1} \ge \left(1 - \tfrac{\alpha i}{t-\alpha(r-i)}\right) Q_{t-i+1},$$

which is exactly as in (3) in the previous case.

Having bounded the value of the linear program, this completes the proof of Lemma 3.7.

3.1.3 Lower Bound

For completeness, we remark that at least for $\alpha > 1$, the exponential dependence on r in Theorem 3.2 is indeed necessary. Consider a "sunflower" hypergraph on $n = rm - m + 1$ vertices that consists of m hyperedges of size r, intersecting at a single vertex, supplemented with m two-uniform cliques of size r each – one for each of the hyperedges. Each of the cardinality-r hyperedges is given weight 1 and each of the cardinality-two edges is given weight $\frac{\alpha-1}{2^r}$. The minimum cut value in this graph is 1, since every cut contains at least one of the r-hyperedges. However, all $\Omega(m \cdot 2^r)$ cuts given by the 2^r bipartitions of a single r-hyperedge, are of weight at most α.

3.2 Proof Of Theorem 3.1

We prove Theorem 3.1 by closely following the proof in the original setting of graphs in [5], and thus we refrain from repeating the full details. Instead, we present an outline of the proof (following the presentation in [6]) while emphasizing the reasons it translates to the hypergraph setting and handling the differences that require a separate treatment.

The main tool used by Benczúr and Karger is random sampling: each edge e is included in the sparsifier with probability p_e, and given weight w_e/p_e if it is included. It is immediate that every cut in the sparsifier preserves its weight in expectation. The main task is thus to carefully select the sampling probabilities p_e in order to both obtain the required number of edges in the sparsifier, and guarantee the required concentration bounds.

As a rough sketch, to guarantee concentration, one needs to apply a Chernoff bound to estimate the probability that the weight of a specific cut (which is a sum of the independent samples of the edges it contains) deviates from its expectation. Subsequently, a union bound over all cuts is used to show the concentration of *all* cuts. A priori it is unclear whether the Chernoff bound is strong enough to handle the exponentially many different cuts in the union bound. The remedy comes from Theorem 3.2 that counts the number of cuts of each weight. It is still unclear how should the random sampling be tuned to handle both the small and large cuts simultaneously. If we are to chose the sampling probability to be small enough to handle the exponentially many large cuts, we run into trouble of small cuts having large variance. On the other hand, increasing the sampling probability imposes a risk of ending up with too many edges in the sparsifier.

Following Benczúr and Karger, we now show that when no edge carries a large portion of the weight in any of the cuts, the cut-counting theorem is sufficient to obtain concentration.

THEOREM 3.9. *Let $H = (V, E, w)$ be a r-uniform hypergraph on n vertices, let $\varepsilon > 0$ be an error parameter, and fix $d \geq 1$. If $H' = (V, E', w')$ is a random subhypergraph of H where the weights w' are independent random variables distributed arbitrarily (and not necessarily identically) in the interval $[0, 1]$, and the expected weight of every cut in H' exceeds $\rho_\varepsilon = \frac{3}{\varepsilon^2}(r + (d+2)\ln n)$, then with probability at least $1 - n^{-d}$, every cut in H' has weight within $1 \pm \varepsilon$ of its expectation.*

One can verify that the proof of the analogous theorem for graphs, as appears in [19], easily extends to the hypergraph setting. Indeed, for the sake of this proof, a cut is merely a sum of independently sampled edges/hyperedges. The lower bound on the weight of the minimum expected cut $\hat{w}$ allows one to show that probability of a cut of weight $\alpha\hat{w}$ to deviate from its expectation is at most $n^{-\alpha(d+2)} \cdot e^{-\alpha r}$ which trades-off nicely with the bound on the number of cuts given by Theorem 3.2.

Informally, Theorem 3.9 implies that in order to obtain the desired concentration bound in the general case, the sampling probability of an edge must be inversely proportional to the size of the largest cut that contains that edge. This motivates the following definitions, and the theorem that follows them.

DEFINITION 3.10. *A hypergraph H is* k-connected *if the weight of each cut in H is at least k.*

DEFINITION 3.11. *A* k-strong component *of H is a maximal k-connected vertex-induced subhypergraph of H.*

DEFINITION 3.12. *The* strong connectivity *of hyperedge e, denoted k_e, is the maximum value of k such that a k-strong component contains (all endpoints of) e.*

Note that one can compute the strong connectivities of all hyperedges in a hypergraph in polynomial time as follows. Compute the global minimum cut, and then proceed recursively into each of the two subhypergraphs induced by the minimum cut. The strong connectivity of an edge would then be the maximum among the minimum cuts of all the subhypergraphs it has been a part of throughout the recursion. The minimum cut in a hypergraph was shown by [22] to be computable in $O(n^2 \log n + mn)$ time. Note that since the total number of subhypergraphs considered throughout the recursion is at most n, there are at most n different strong-connectivity values in any hypergraph.

THEOREM 3.13. *Let H be an r-uniform hypergraph, and let $\varepsilon > 0$ be an error parameter. Consider the hypergraph H_ε obtained by sampling each hyperedge e in H independently with probability $p_e = \frac{3(r+(d+2)\ln n)}{k_e \varepsilon^2}$, giving it weight w_e/p_e if it is included. Then with probability at least $1 - O(n^{-d})$*

1. *The hypergraph H_ε has $O\left(\frac{n}{\varepsilon^2}(r + \log n)\right)$ edges.*
2. *Every cut in H_ε has weight between $(1-\varepsilon)$ and $(1+\varepsilon)$ times its weight in H.*

The proof of the theorem is again identical to the proof of [6, Theorem 2.6] for the graph setting. This includes a bound on the total number of edges in H_ε that follows from the property that $\sum_{e \in E} w_e/k_e \leq n-1$ (see [6, Lemma 2.7]). The only thing that needs verifying is that strong-connectivity induces a recursive partitioning of the vertices of the hypergraph, just as it does when dealing with graphs. This is in fact the case, mainly because the components considered in the definitions are vertex-induced, and therefore the cardinality of the hyperedges plays no part. One can then decompose the hyperedges of the hypergraph to "layers", based on their strong-connectivity, and apply Theorem 3.9 to each layer separately.

As to the running time, it is dominated by the time required to compute the strong connectivities of all the edges in the hypergraph, which as mentioned above, can be done by running the $O(n^2 \log n + mn)$ min-cut algorithm at most n times. Therefore, the total running time required to compute H_ε is $O(n^3 \log n + mn^2)$. Since we may assume that $m = \Omega(n \log n)$ (as there is no point to construct the sparsifier otherwise), the second term dominates and thus the running time is simply $O(mn^2)$.

To complete our discussion we bring the reader's attention to a couple of places where the cardinality of the hyperedges has played part:

- The modified parameter $p_e = \frac{3(r+(d+3)\ln n)}{k_e \varepsilon^2}$ counters the number of cuts from Theorem 3.2 (at most $O(2^{\alpha r} n^{2\alpha})$ cuts of weight $\alpha\hat{w}$) and the number of distinct edge-connectivity values, which is at most n.[10]
- The number of edges in the sparsifier is (with high probability) $O\left(\frac{n}{\varepsilon^2}(r + \log n)\right)$ since the sampling probability is also linear in r.

3.3 SAT Sparsification

LEMMA 3.14. *For every r-CNF formula Φ with n variables and m clauses, there exists an $(r+1)$-uniform hypergraph H with $2n+1$ vertices, and a mapping $\Pi : \{0,1\}^n \to \{0,1\}^{2n+1}$ (from assignments to Φ, to cuts in H), such that for every assignment φ, it holds that $val_\Phi(\varphi) = val_H(\Pi(\varphi))$.*

PROOF. Consider an r-CNF formula Φ with variables $\{x_i\}_{i \in [n]}$. We construct the weighted hypergraph H whose vertices are $\{x_i, \neg x_i\}_{i \in [n]}$ and a special vertex F. For each clause $\ell_{i_1} \vee \ell_{i_2} \vee \cdots \vee \ell_{i_r}$, we add a hyperedge $\{\ell_{i_1}, \ldots, \ell_{i_r}, F\}$. Moreover, let Π be the mapping that maps an assignment to Φ to the cut in H obtained by placing all vertices corresponding to true literals on one side, and the F vertex together with all vertices corresponding to false literals on the other side.

For an assignment φ to Φ, it is clear that a hyperedge is contained in the cut $\Pi(\varphi)$ if and only if at least one of the vertices it contains is on the opposite side of F. Therefore the weight of $\Phi(\varphi)$ is exactly the value of φ. $\square$

Theorem 3.4 follows from Lemma 3.14 and Theorem 3.1. The running time for constructing the sketch of the CNF formula is dominated by the running time of constructing the hypergraph sparsifier, which is $O(mn^2)$, where m is the number of clauses in the original CNF formula.

[10] In their analysis [6] take a union bound over n^2 distinct edge-connectivity values. For hypergraphs using the stronger linear bound (instead of the trivial n^r) is crucial.

4. FUTURE DIRECTIONS

Our results raise several questions that deserve further work.

Sketching Max-Cut.

Our results and the results of [15] make progress on the *streaming* complexity of approximating MAX-CUT, showing polynomial space lower bounds. To fully resolve this problem, one still needs to determine whether $\Omega(n)$ space is necessary for any non-trivial approximation (i.e., strictly better than $1/2$), or whether there is a sublinear-space streaming algorithm that beats the $1/2$-approximation barrier.

Also of interest is the *communication complexity* of approximating MAX-CUT in the *multi-round two-party* model, and even a multi-round analogue of BOOLEAN HIDDEN HYPERMATCHING.

Sketching Cuts in Hypergraphs.

Can one improve over the linear dependence on r in hypergraph sparsification (Theorem 3.1)? Or perhaps prove a matching lower bound? Such a refinement could be especially significant when the hyperedge cardinality is unbounded, in which case the known upper bound is $O(n^2/\varepsilon^2)$.

General CSPs.

Do all CSPs admit sketches of size (in bits or in machine words) $o(n^r)$, or even $\tilde{O}(n)$, that preserve the values of all assignments? From the direction of lower bounds, we may even restrict ourselves to sketches that are *sub-instances*, and ask whether there exist CSPs where such sketches require size $\Omega(nr)$ or even $n^{\Omega(r)}$?

5. ACKNOWLEDGEMENTS

We thank Alexandr Andoni and David Woodruff for useful discussions at early stages of this work. Work supported in part by a US-Israel BSF grant #2010418, Israel Science Foundation grant #897/13, and by the Citi Foundation.

6. REFERENCES

[1] K. J. Ahn and S. Guha. Graph sparsification in the semi-streaming model. In *36th International Colloquium on Automata, Languages and Programming: Part II*, ICALP '09, pages 328–338. Springer-Verlag, 2009. `arXiv:0902.0140`, `doi:10.1007/978-3-642-02930-1_27`.

[2] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467. SIAM, 2012. `doi:10.1137/1.9781611973099.40`.

[3] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of the 31st Symposium on Principles of Database Systems*, PODS '12, pages 5–14. ACM, 2012. `doi:10.1145/2213556.2213560`.

[4] A. Andoni, R. Krauthgamer, and D. P. Woodruff. The sketching complexity of graph cuts. *CoRR*, abs/1403.7058, 2014. `arXiv:1403.7058`.

[5] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, STOC '96, pages 47–55, New York, NY, USA, 1996. ACM. `doi:10.1145/237814.237827`.

[6] A. A. Benczúr and D. R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *CoRR*, cs.DS/0207078, 2002. `arXiv:cs/0207078`.

[7] M. K. de Carli Silva, N. J. A. Harvey, and C. M. Sato. Sparse sums of positive semidefinite matrices. *CoRR*, abs/1107.0088, 2011. `arXiv:1107.0088`.

[8] H. Dell and D. van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, STOC '10, pages 251–260. ACM, 2010. `doi:10.1145/1806689.1806725`.

[9] E. A. Dinitz, A. V. Karzanov, and M. V. Lomonosov. On the structure of the system of minimum edge cuts in a graph. *Issledovaniya po Diskretnoi Optimizatsii*, pages 290–306, 1976. URL: `http://alexander-karzanov.net/ScannedOld/76_cactus_transl.pdf`.

[10] A. Goel, M. Kapralov, and I. Post. Single pass sparsification in the streaming model with edge deletions. *CoRR*, abs/1203.4900, 2012. `arXiv:1203.4900`.

[11] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, Nov. 1995. `doi:10.1145/227683.227684`.

[12] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9:551–570, 1961. `doi:10.1137/0109047`.

[13] M. Henzinger and D. P. Williamson. On the number of small cuts in a graph. *Information Processing Letters*, 59(1):41 – 44, 1996. `doi:10.1016/0020-0190(96)00079-8`.

[14] P. Indyk, A. McGregor, I. Newman, and K. Onak. Open questions in data streams, property testing, and related topics. `http://people.cs.umass.edu/~mcgregor/papers/11-openproblems.pdf`, 2011. See also `http://sublinear.info/45`.

[15] M. Kapralov, S. Khanna, and M. Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2015. To Appear. `arXiv:1409.2138`.

[16] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, 2014. `arXiv:1407.1289`.

[17] D. R. Karger. Global min-cuts in $\mathcal{RNC}$, and other ramifications of a simple min-cut algorithm. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=313559.313605`.

[18] D. R. Karger. Better random sampling algorithms for flows in undirected graphs. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 490–499. SIAM, 1998.

URL:
http://dl.acm.org/citation.cfm?id=314613.314833.

[19] D. R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, 1999. doi:10.1287/moor.24.2.383.

[20] T. Kaufman, D. Kazhdan, and A. Lubotzky. Ramanujan complexes and bounded degree topological expanders. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, 2014. arXiv:1409.1397.

[21] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

[22] R. Klimmek and F. Wagner. A simple hypergraph min cut algorithm. Technical Report B 96-02, Freie Universität Berlin, Fachbereich Mathematik, 1996. URL: http://edocs.fu-berlin.de/docs/servlets/MCRFileNodeServlet/FUDOCS_derivate_000000000297/1996_02.pdf.

[23] F. T. Leighton and A. Moitra. Extensions and limits to vertex sparsification. In *42nd ACM symposium on Theory of computing*, STOC, pages 47–56. ACM, 2010. doi:10.1145/1806689.1806698.

[24] M. V. Lomonosov and V. Polesskii. Lower bound of network reliability. *Problemy Peredachi Informatsii*, 8(2):47–53, 1972. URL: http://www.mathnet.ru/links/36bd620cb75111781cef454d72f0d773/ppi824.pdf.

[25] A. Louis. Hypergraph Markov operators, eigenvalues and approximation algorithms. *CoRR*, abs/1408.2425, 2014. arXiv:1408.2425.

[26] A. Louis and Y. Makarychev. Approximation algorithms for hypergraph small set expansion and small set vertex expansion. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28, pages 339–355, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.339.

[27] A. McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014. doi:10.1145/2627692.2627694.

[28] A. Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *50th Annual Symposium on Foundations of Computer Science*, FOCS, pages 3–12. IEEE, 2009. doi:10.1109/FOCS.2009.28.

[29] H. Nagamochi, K. Nishimura, and T. Ibaraki. Computing all small cuts in an undirected network. *SIAM Journal on Discrete Mathematics*, 10(3):469–481, 1997. doi:10.1137/S0895480194271323.

[30] I. Newman and Y. Rabinovich. On multiplicative λ-approximations and some geometric applications. *SIAM Journal on Computing*, 42(3):855–883, 2013. doi:10.1137/100801809.

[31] D. Peleg and A. A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

[32] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

[33] Y. Yamaguchi, A. Ogawa, A. Takeda, and S. Iwata. Cyber security analysis of power networks by hypergraph cut algorithms. In *Proceedings of the Fifth Annual IEEE International Conference on Smart Grid Communications*, 2014. To appear. URL: http://www.keisu.t.u-tokyo.ac.jp/research/techrep/data/2014/METR14-12.pdf.

[34] W. Yu and E. Verbin. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 11–25, 2011. doi:10.1137/1.9781611973082.2.

[35] M. Zelke. *Algorithms for Streaming Graphs*. PhD thesis, Mathematisch-Naturwissenschaftliche Fakultät II, Humboldt-Universität zu Berlin, 2009. Published at Südwestdeutscher Verlag für Hochschulschriften. URL: http://www.tks.informatik.uni-frankfurt.de/data/doc/diss.pdf.

[36] M. Zelke. Intractability of min- and max-cut in streaming graphs. *Inf. Process. Lett.*, 111(3):145–150, Jan. 2011. doi:10.1016/j.ipl.2010.10.017.

[37] J. Zhang. A survey on streaming algorithms for massive graphs. In C. C. Aggarwal and H. Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 393–420. Springer, 2010. doi:10.1007/978-1-4419-6045-0_13.