Sparsification: Graphs, Codes, CSPs

Madhu Sudan
Harvard University

Joint work with
Sanjeev Khanna (Penn) and Aaron (Louie) Putterman (Harvard)

Sparsification

- Lossy compression $\leq$ Sparsification $\leq$ Compression

- Compression: $X \mapsto Comp(X) \mapsto X$

- Noisy compression: $X \mapsto NC(X) \mapsto \tilde{X}$ s.t. $\delta\left(X, \tilde{X}\right) \to 0$
  - Preserves most of $poly(|X|)$ queries

- Sparsification (for class C of queries):
$$X \mapsto Sparse(X) \mapsto \{(1 \pm \epsilon)q(X)\}_{q \in C}$$
  - Approximately preserves all of $|C|$ queries (usually exponentially many)

Benczur-Karger Cut Sparsification

- Thm [Karger 94, BK97]: Every graph on n vertices can be sparsified to $\tilde{O}(n)$ bits while estimating all (2^{n-1}) cuts to within $1 \pm \epsilon$
  - (Note – full information = $O(n^2)$ bits).
- Key ingredient: Karger's cut counting bound
- Lemma [K]: in unweighted graph G

$$\#\{\text{cuts of size} \leq \alpha \cdot \backslash\text{mincut}(G)\} \leq n^{2\alpha}$$

- Random sample of $\tilde{O}\left(\frac{m}{c}\right)$ edges suffices.
- [BK] Non-uniform sampling reduces to $\tilde{O}(n)$ samples (How?)

What else can be sparsified?

- "Structure" ≔ data + set of queries …
- What other structures can be sparsified?
  - Graph Laplacians wrt quadratic form queries
    - Data = L_G ; Query = $x \in R^n$; Ans: $x^T L_G x$
  - Hypergraph Cut Sparsifiers
    - Data = (V,E) ; Query = $S \subseteq V$; Ans: $\left|E\left(S,\overline{S}\right)\right|$
  - SAT sparsifier
    - Data = Sat formula; Query = assignment ; Ans = # clauses satisfied by assignment.
  - CSP(P) sparsifier? [Kogan-Krauthgamer]
    - Data = P constraints on n vars ; Query = assignment …
    - [FK, BZ]: Classification of binary predicates with near linear sparsifiers
  - XOR-SAT sparsifier?
    - Data = XOR-SAT formula ….

This talk:

- Code sparsification (more generally – additive codes over abelian groups):
  - Data = (generator matrix of) linear code.
  - Query = message
  - Ans = (Hamming) weight of its encoding.
- Motivation: Generalizes graph- and hypergraph-sparsification.
- Applications to CSP sparsification:
  - Classification of ternary Boolean CSPs
  - Classification of all symmetric Boolean CSPs
  - Classification* of all Boolean CSPs with non-trivial sparsification

Some theorems

- Thm 1: Every linear code $E: \mathbb{F}_q^k \to \mathbb{F}_q^n$ can be sparsified to $\tilde{O}(k^2 \log^2 q)$ bits.
  - More specifically, ∃ weighted sample of $\tilde{O}(k \log q)$ coordinates s.t. weighted hamming weights in sampled coordinates approximate original weight.
- Thm 2: Every code $E: \mathbb{Z}^k \to G^n$ can be sparsified to $\tilde{O}(k \log^2 |G|)$ coordinates, ∀ abelian group G.
- Thm 3: Every degree t poly function $E: \mathbb{Z}^k \to G^n$ can be sparsified to $\tilde{O}(k^t \log^2 |G|)$ coordinates

Some CSP theorems

- Thm 4: $\forall P: \{0,1\}^3 \to \{0,1\}$ CSP(P) is $\tilde{O}(n^t)$-sparsifiable iff P does not project to AND_{t+1}
  - $P: \{0,1\}^r \to \{0,1\}$ projects to $Q: \{0,1\}^s \to \{0,1\}$ if $\exists \Pi: [r] \to \{Y_1 \dots Y_s\} \cup \{\overline{Y}_1 \dots \overline{Y}_s\} \cup \{0,1\}$ s.t.
  $$Q(Y_1 \dots Y_s) = P\big(\Pi(1) \dots \Pi(r)\big)$$
- Thm 5: $\forall$ symmetric $P: \{0,1\}^r \to \{0,1\}$ CSP(P) is near-linear sparsifiable iff $wt\big(P^{-1}(0)\big)$ form arithmetic progression.
- Thm 6: $\forall P: \{0,1\}^r \to \{0,1\}$ CSP(P) is sparsifiable to $\tilde{O}(n^{r-1})$ constraints iff $|P^{-1}(1)| \geq 2$

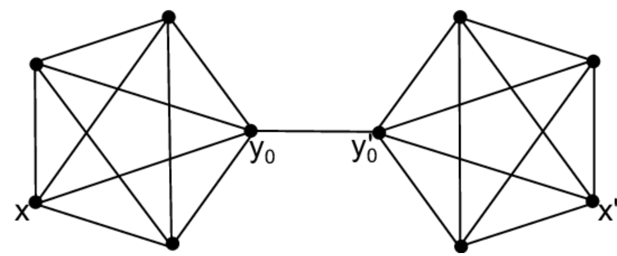
Proofs

Graph Sparsification

Why does a random sample not work?

- Pick $\tilde{O}(n)$ constraints uniformly at random
- Output $\frac{m}{n}.(\#"sampled \,\&\, satisfied"\ constraints)$

- Gives additive ($\pm\epsilon m$) approximation;
- … but not multiplicative $(1 \pm \epsilon)$ approximation



Cut counting bound

- Fix a cut S w. $\leq \alpha \cdot c$ edges
- Contract $n - 1 \leq \frac{2m}{c}$ random edges (till #vertices = 2)
- $\Pr[i\text{th edge from end crosses } S] \leq \frac{2\ \alpha \cdot c}{i.c} = \frac{2\alpha}{i}$
- $\Pr[\text{no edge crosses } S] \geq \prod_i \left(1\ - \frac{2\alpha}{i}\right) \geq n^{-2\alpha}$
- Pr[S final cut] $\geq n^{-2\alpha}$
- $\Rightarrow$ # {cuts w. $\leq \alpha c$ edges} $\leq n^{2\alpha}$

Graph Sparsifiers from c.c. bound

- [K]: Sample $\frac{10m}{c}\log n$ edges …
  - Pr [cut S of size αc not sampled well] $\leq n^{-10\alpha}$
  - Pr [∃ cut of size αc not sampled well] $\leq n^{-8\alpha}$
  - Now union over α
- [BK] Define strength of edges ; sample edges w.p. prop. to strength …
- [Our simpler proof (loses log factors)]:
  - Given G, let G_0 be union of cuts of size $\sqrt{\frac{m}{cn}}$; $G_1 \dots G_t$ be c.c.s of the rest;
  - $m(G_0) \leq \sqrt{\frac{mn}{c}}$; $min - cut(G_i) \geq \sqrt{\frac{m}{cn}}$; $\frac{m}{c}$ better in all!
  - Recurse+weight appropriately (by mincut)!

Code Sparsification

Code Sparsification

- Need an analog of cut counting bound …
  - "In every code C of min dist d,
    $\#\{\text{codewords of wt} \leq \alpha d\} \leq k^{\alpha}$" ?
- Patently false: Asymptotically good code has $d, k = \Omega(n)$, and so $2^{\Omega(n)}$ words of weight $O(d)$
  (Aside: Hypergraph cut counting bound also fails similarly!! Obstacle to prior work.)
- But asymptotically good code is already sparsified! So not obstacle to sparsification.
- Needs a modified "cut counting bound"

Code counting Lemma

- Informally, every code has a good subcode supported on few coordinates, or satisfies Karger-style counting bound.

- Lemma: $\forall t \in \mathbb{Z}^+, C \subseteq \mathbb{F}_q^n$ we have:
  1. $\forall \alpha$ #{codewords of wt $\leq \alpha \cdot t\} \leq q^\alpha \binom{n}{\alpha}$ OR
  2. $\exists C' \leq C$, s.t. $|\mathrm{supp}(C')| \leq \dim(C') \cdot t$

- Corollary: $\forall t \in \mathbb{Z}^+, C \subseteq \mathbb{F}_q^n, \exists S \subseteq [n], |S| \leq \dim(C).t$ s.t.

  $\forall \alpha$ #{codewords of $C|_{\bar{S}}$ of wt $\leq \alpha \cdot t\} \leq q^\alpha \binom{n}{\alpha}$

Code Counting ⇒ Sparsification

- $\mathrm{Sparsify}(C)$
  - Let $t = \sqrt{\frac{n}{k}}$ where $k = \dim C$
  - Apply Corollary and let $C_1 = C|_S$ and $C_2 = C|_{\overline{S}}$
  - Return $\mathrm{Sparsify}(C_1) \cup \sqrt{t} \cdot \mathrm{Sparsify}(C_2)$
  - QED

Proof of Code Counting

- Contract(C, t):
  - If $|\mathrm{supp}(C)| \leq t \cdot \dim(C)$ stop "Case 2";
  - If $\dim C > \alpha$
    - Pick random coord. $j \in [n]$ s.t. $C|_{\{j\}} \neq 0$
    - $C' = C - \{c \in C$ s.t. $c_j \neq 0\}$
    - Contract(C', t)
  - Else, output "Case 1" + random codeword of C
- Fix word $c \in C$ of weight $\leq \alpha t$
- $\Pr[c \notin C'] \leq \frac{\alpha}{\dim(C)}$
- $\Rightarrow$Pr[c survives and output at end] $\geq \binom{n}{\alpha}^{-1} q^{-\alpha} \cdot$

Implications + Extensions

Hypergraph Sparsification

- Hypergraph: Say r-uniform hypergraph on n vertices. Edge e cut by $(S, \bar{S})$ if $e \cap S, e \cap \bar{S} \neq \phi$.
- Q: $\exists\, \tilde{O}(n)$ hypergraph cut sparsifiers?
  - [KK'15]: $\tilde{O}(nr)$- sparsifiers exist
  - [CKN'20] Improve to $\tilde{O}(n)$
- Our proof:
  - Let $q \approx n$ prime, map edge e to row vector in $\mathbb{F}_q^n$ with nnz entries $(1,1,1, \ldots - (r-1))$
  - Consider code generated by columns of matrix with a row for each edge.
  - Sparsifying code sparsifies hypergraph!

Variations

- Can sparsify codes $E: \mathbb{Z}^k \to G^n$, for finite abelian group G, to $\tilde{O}(k \log G)$ rows.
  - Proof: Some linear algebra breaks down. Replace dimension etc with actual counts, Gaussian elimination with HNF.
- Can sparsify degree t maps $P: \mathbb{Z}^k \to G^n$ to $\tilde{O}(k^t \log G)$ coordinates.
- Applications:
  - Classify all symmetric Boolean CSPs with near linear sparsification
  - Classify all r-ary Boolean CSPs with $o(n^r)$-sparsification.

Open Questions

- Sparsification results non-constructive!
  - Open: Construct polytime algorithm to find sparsification?
  - Given code C and integer t find support of a high-rate subcode C'?
- CSP Classification:
  - Only upper bound tool: our group-based-polynomial sparsifier
  - Only lower bound tool: Projection to t-AND.
  - The two don't meet ☹
  - New ideas?

Thank You!