(Hyper)-Graph Problems via Global Queries

STOC 2026 Workshop, Salt Lake City, Utah

Organizers:
Deeparnab Chakrabarty (Dartmouth)
Sagnik Mukhopadhyay (Birmingham)

Speakers

Louie Putterman

Robi Krauthgamer

Sanjeev Khanna

Amir Nayyeri

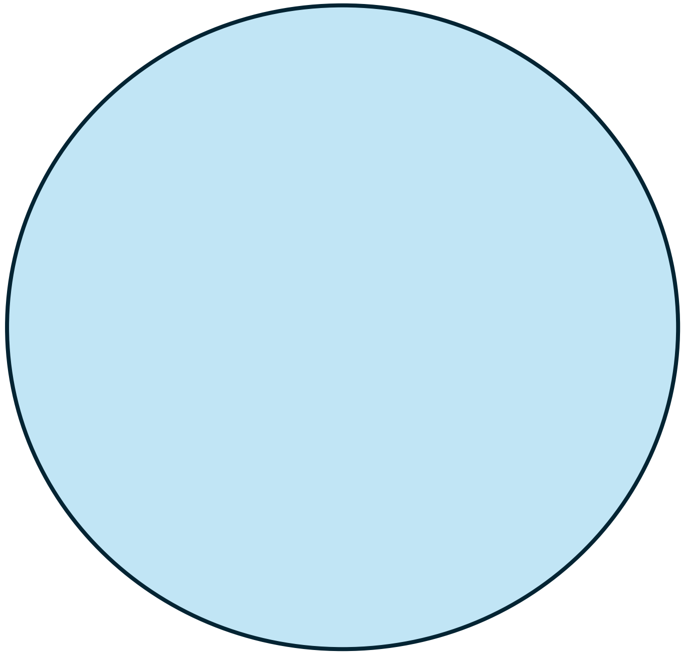
Troy Lee

David Woodruff

Sepehr Assadi

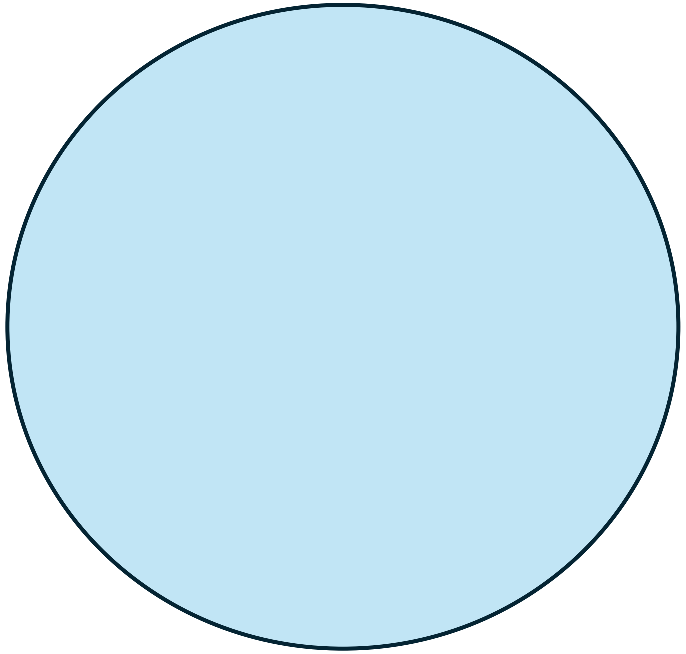
Query Access Model (QAM) for algorithms

- Access object via certain queries
- Solve algorithmic prob, usually appx
- Usually focus on query complexity
- Upper bounds / Lower bounds



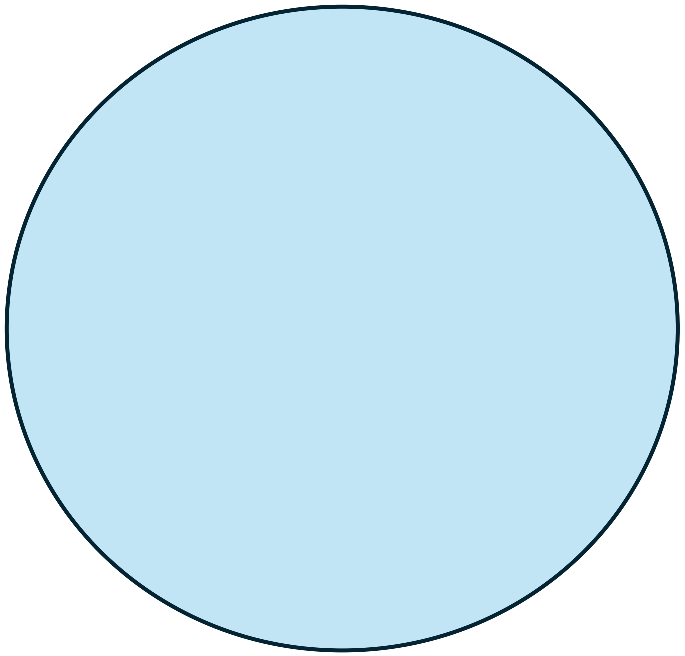
“Local” Queries: Sublinear Algorithms

- Random Access (vertex / edge)
- Get “local” inf about object
  - degree
  - random neighbor



“Global” Queries

- Vertex Set known
- **Query** info about **subset** of vertices
  - Is the set an independent set?
  - How many edges are in the cut set?
  - What’s the shortest path length between s & t?



Why?

- Sometimes that is the way to access
- Teases out complexity of a problem
- Special cases of a general problem (eg, Submod. Func. Min.)
- Connections with other areas: sketching, streaming, etc

Considerations

- **Qu**ery Complexity
- **A**daptivity / Round complexity
- **R**andomized / Deterministic
- **T**ime Efficiency (after getting query answers)

Rest of this talk

- An **incomplete** survey
- Classics [40s to 60s]
- Graph Learning [90s to 10s]
- Graph properties [recent-er]

Classics [40s to 60s]

Reconstructing vectors

- Unknown object: n-dimensional vector $\boldsymbol{x} \in \{0,1\}^n$
- Two Query Models: both take input $S \subseteq [n]$
  - (1) Hit(S) = YES if $x_i = 1$ for some $i \in S$, NO otherwise
  - (2) Sum(S) = $\sum_{i \in S} x_i$

Group Testing Dorfman '43

Coin Weighing Shapiro '60

Goal: reconstruct $\boldsymbol{x}$

Group Testing

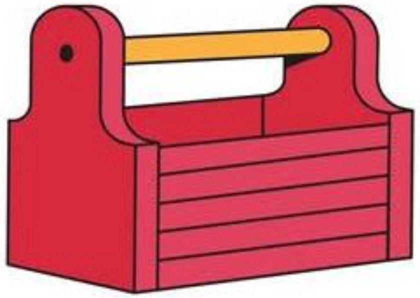
- If nothing else is known, n-queries needed

every query gives only one bit of info

- Usually focus on at most d ones case

need $\log\binom{n}{d} \approx d \log n$ queries

Group Testing (d=1)



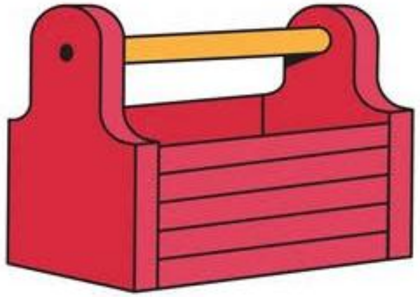
Tool: Binary Search

- Take arbitrary half A of [n] and query Hit(A)
- if YES, recurse on A; o/w recurse on [n]\A

- Optimal in Query Complexity

- Adaptive

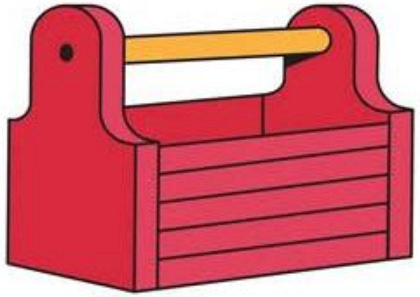
Group Testing (d=1)



Tool: "non-adaptive" binary search

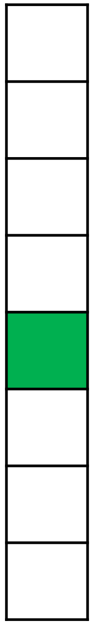
0	0	0	1	1	1	1
0	1	1	0	0	1	1
1	0	1	0	1	0	1

Group Testing (d=1)



Tool: “non-adaptive” binary search

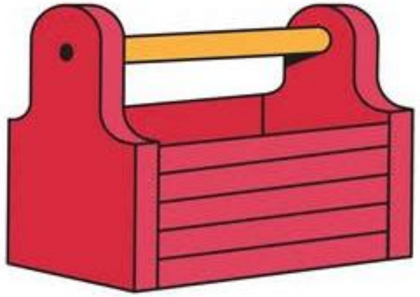
Y	0	0	0	1	1	1	1
N	0	1	1	0	0	1	1
Y	1	0	1	0	1	0	1



Group Testing (general d)

- If adaptivity not an issue, $O(d \log n)$ repeated binary search
- Non-adaptive?

Group Testing (general d)



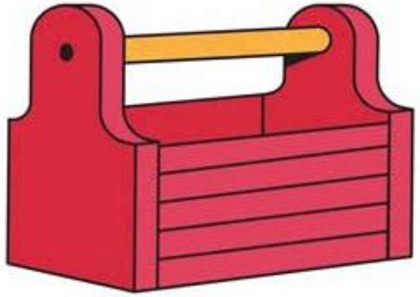
Tool: random sampling

- Sample K subsets picking each i with probability $1/d$
- Zero all coordinates which participate in a NO query
- Rest all 1

- If x_i = 1, we report 1 on i
- If x_i = 0 Pr[a sample R zeros i] = Pr[$i \in R$] * Pr[no 1 in R]

$\frac{1}{d}$ $\left(1 - \frac{1}{d}\right)^{d} \geq \frac{1}{4}$

Group Testing (general d)

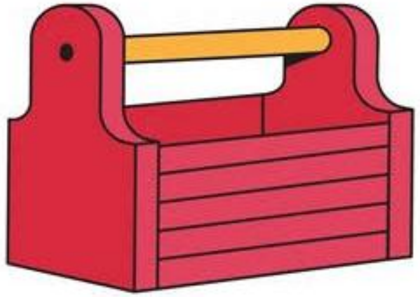


Tool: random sampling

- Sample K subsets picking each i with probability $1/d$
- Zero all coordinates which participate in a NO query
- Rest all 1

- If x_i = 1, we report 1 on i
- If x_i = 0   Pr[a sample R zeros i] $\geq 1/4d$

Group Testing (general d)



Tool: random sampling

- Sample K subsets picking each i with probability $1/d$
- Zero all coordinates which participate in a NO query
- Rest all 1

- If x_i = 1, we report 1 on i
- If x_i = 0 Pr[we don't report 0] $\leq \left(1 - \frac{1}{4d}\right)^K$

If $K = 4d\ln\left(\frac{n}{\delta}\right)$, then algo correct with prob $1 - \delta$

Group Testing (general d)

- There is an $O\left(d\log\left(\frac{n}{\delta}\right)\right)$ query *non-adaptive* randomized algo which on any x succeeds with prob $1-\delta$

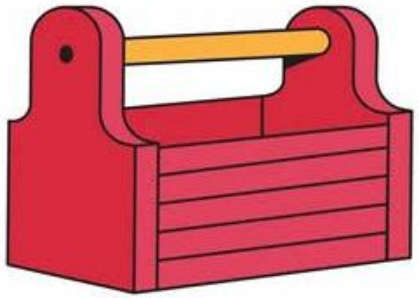
- If $\delta < \frac{1}{\binom{n}{d}} \sim \frac{1}{n^d}$, then by union bound there *exists* $O\left(d\log\left(\frac{n}{\delta}\right)\right) \sim O(d^2 \log n)$ non-adaptive queries whose answers determine x

These can be made explicit using connections to error correcting codes

Coin Weighing

- Unknown Boolean vector $\boldsymbol{x} \in \{0,1\}^n$
- SUM Query(S) $\mapsto \sum_{i \in S} \boldsymbol{x}_i$
- *Question:* Learn $\boldsymbol{x}$

Need at least $\frac{n}{\log_2 n}$ queries



$\mathbf{A} \in \{\mathbf{0}, \mathbf{1}\}^{\ell \times n}$ $\quad \boldsymbol{x} \quad = \quad \boldsymbol{Ax}$

$\ell = \frac{2n+o(n)}{\log_2 n}$ suffices!

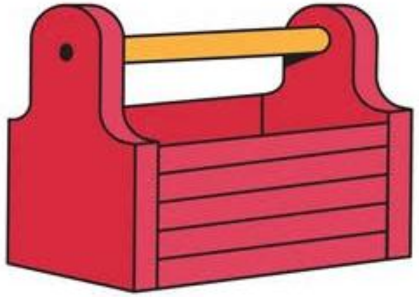
$\boldsymbol{Ax}$ determines $\boldsymbol{x}$

Soderbergh, Shapiro (1960)
Erdos, Renyi (1963)
Cantor, Mills (1966)
Lindstrom (1965)

Coin Weighing

- Unknown Boolean vector $\boldsymbol{x} \in \{0,1\}^n$
- SUM Query(S) $\mapsto \sum_{i \in S} \boldsymbol{x}_i$
- *Question:* Learn $\boldsymbol{x}$

Need at least $\frac{n}{\log_2 n}$ queries



- If x has $\leq d$ ones, there *exists* $O(\frac{d \log\left(\frac{n}{d}\right)}{\log d})$ query non-adaptive algorithms
- $O(\frac{d \log(n/d)}{\log d})$ query *efficient* adaptive algorithms

Bshouty (2009), Bshouty-Mazzawi (2011)

- If x is weighted, non-negative with supp $\leq d$, there is $O(\frac{d \log(n/d)}{\log d})$ query randomized adaptive algo

Choi (2013)

Graph Learning [90s to 10s]

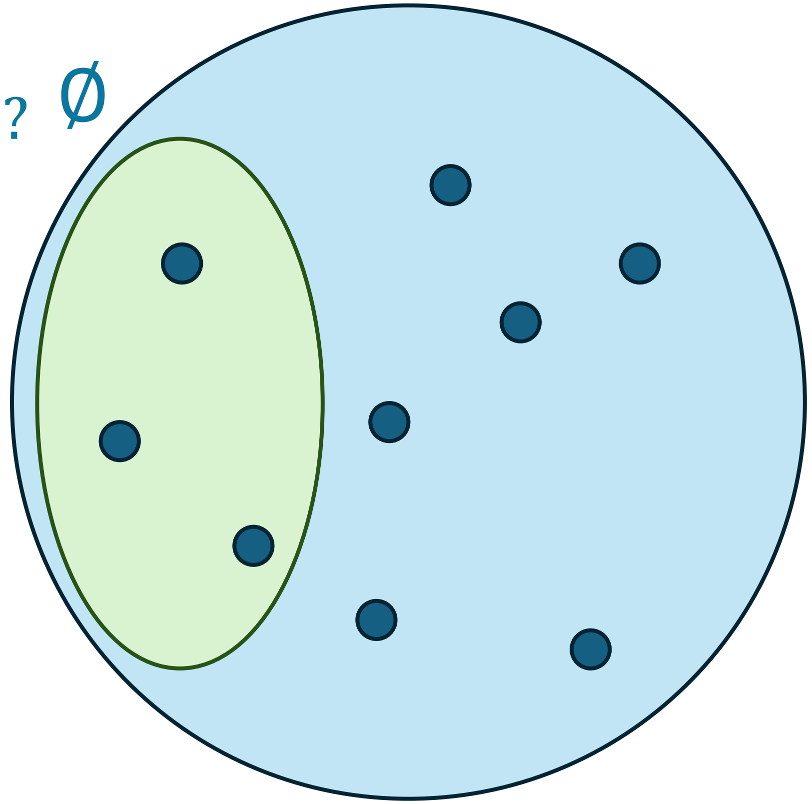
$E[S] := \{uv \in E : u \in S, v \in S\}$

$\partial S := \{uv \in E : u \in S, v \notin S\}$

Undirected Graphs

- Unknown object: undirected graph with known V
  - Edge Detecting / Ind Set $S \mapsto E[S] =_? \emptyset$
  - Additive $S \mapsto |E[S]|$
  - CUT $S \mapsto |\partial S|$

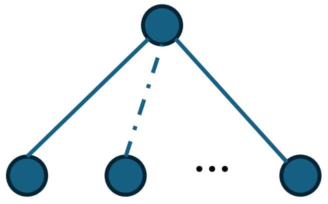
Submodular function



- Goal: reconstruct the edges of the graph

Grebinski-Kucherov (1997)

Observations

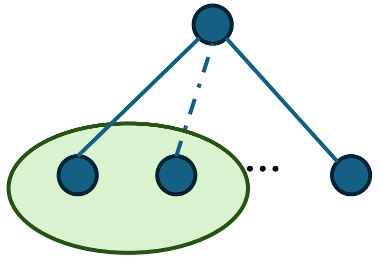


- Independent Set Queries generalizes group testing
- Additive Queries generalizes coin weighing
- CUT queries $\equiv$ Additive queries after degrees

$$\sum_{v \in S} \deg(v) = 2|E[S]| + |\partial S|$$

- Graph can be learnt in $O(n^2 / \log n)$ ADD/CUT queries...

Learn every vertex's nbrhood*



- ...this is optimal

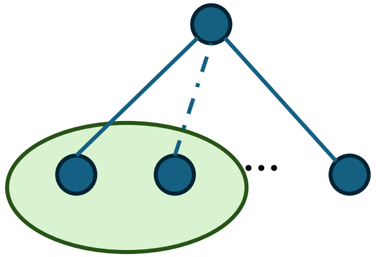
Focus: case when graph has only m edges

$\Omega(m \log n)$ IS

$\Omega(m \log n / \log m)$ CUT/ADD

Independent Set Queries with m edges

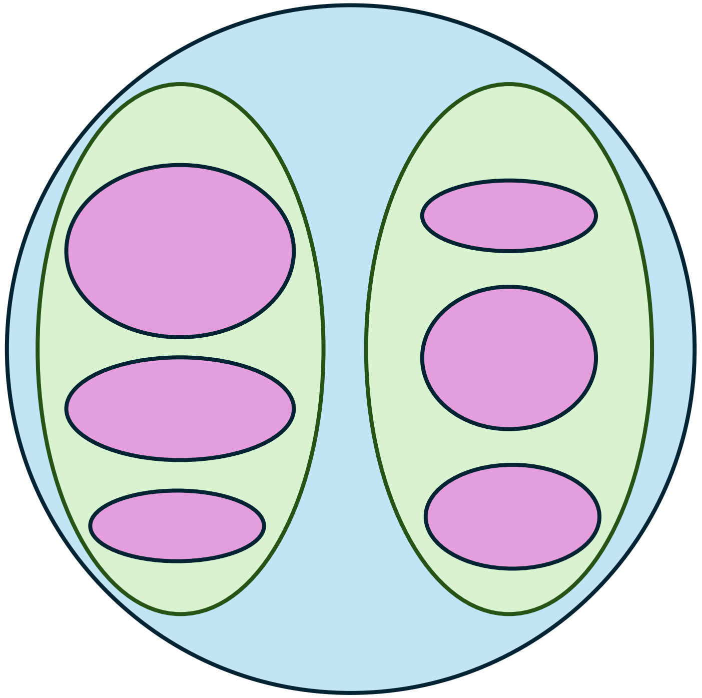
- $\Omega(m \log n)$ is the best one can hope for
- Can't use "learn vrtx nbrhood" trick since "green zone" may not be independent



Angluin-Chen (2004)

Exercise

- Recursively learn two halves
- Each half can be $O(\sqrt{m})$-colored
- Learn edges between Ind Sets using group testing costing $O(\log n)$ per edge

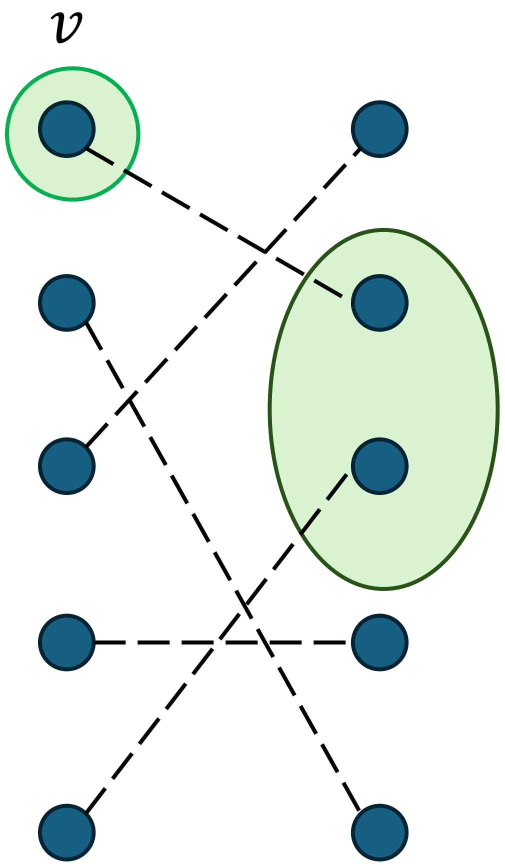


ADD/CUT queries with m edges

- With ADD/CUT queries, lower bound: $\Omega(m \log n / \log m)$
- In particular, for sparse graphs: $\Omega(n)$
- This is also the best known algorithmically!
  - $O\left(\frac{m \log n}{\log m}\right)$ query algorithm for m-edge graphs
  - Even works with weights on edges

Choi, Kim (2008)
Bshouty, Mazzawi (2009)
Mazzawi (2010)
Bshouty, Mazzawi (2011)
Choi (2013)

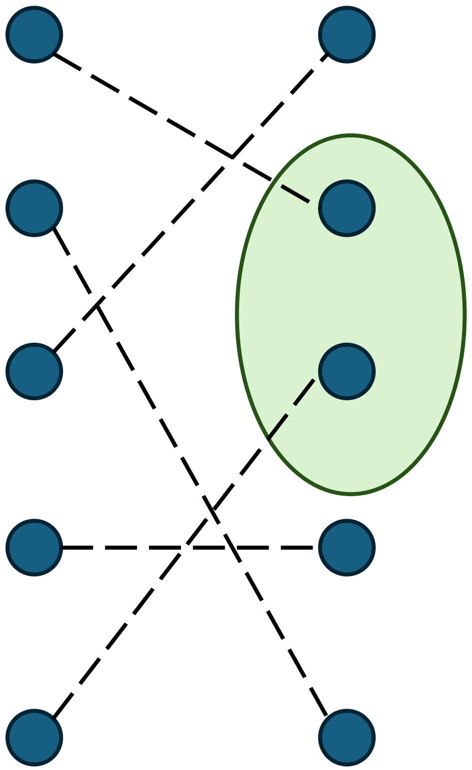
Learning a Hidden Matching



- v needs to learn a Boolean vector with exactly one 1
- There are $\sim\log_2 n$ subsets $S_1, \ldots, S_{\log n}$ such that knowing ADD(v, S_i)'s will reveal edge inc. on v
- Key: *these subsets don't depend on v*
- What we need: for each S_i, which vertices have edge to S_i

Grebinski, Kucherov (2000)

Learning a Hidden Matching



- What we need: for each S_i, which vertices have edge to S_i
- $\boldsymbol{x}^{(i)} \in \{0,1\}^n$ where $\boldsymbol{x}_j^{(i)} = 1$ iff j has an edge to S_i
- Sum query on this unknown $\boldsymbol{x}$ with ADD query
- Can learn $\boldsymbol{x}^{(i)}$ in $\sim \frac{2n}{\log_2 n}$ queries
- Doing this for all $\sim\log_2 n$ sets gives $2n$ query algo

Grebinski, Kucherov (2000)

$E[S]:=\{e \in E : e \subseteq S\}$

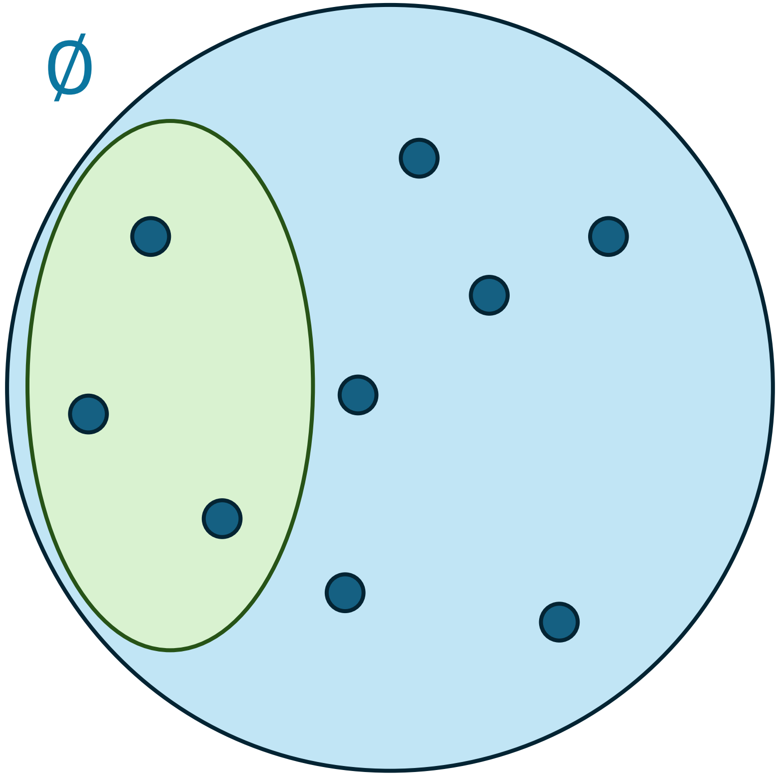
$\partial S:=\{e \in E : e \cap S \neq \emptyset, e \cap \bar{S} \neq \emptyset\}$

Hypergraphs

- Unknown object: undirected **hypergraph** with known V
  - Edge Detecting / Ind Set $S \mapsto E[S] =_? \emptyset$
  - Additive $S \mapsto |E[S]|$
  - CUT $S \mapsto |\partial S|$

Submodular function

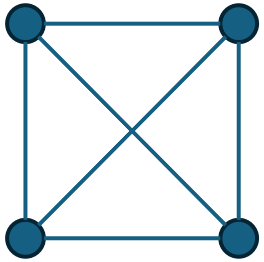
- Goal: reconstruct the hyperedges

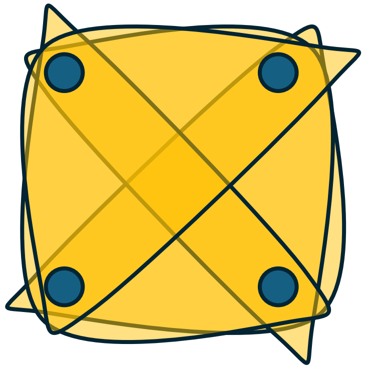


Angluin-Chen (2004)

Observations

- Additive and CUT queries are **not** equivalent
- In fact, CUT queries **cannot** reconstruct hypergraphs!





- On the other hand, can learn m-hyperedge hypergraph in $O(mn \log n)$ ADD queries

Angluin-Chen (2006)

Graph Properties [2018 - …]

Two papers from ITCS 2018

Unknown object: undirected graph with known V

Beame, Har-Peled, Ramamoorthy, Rashtchian, Sinha

Edge Detecting / Ind Set $S \mapsto \ E[S] =_? \ \emptyset$

Goal: Estimate number of edges

Chen, Waingarten, Levi (2020)
Bhattacharya-Bishnu-Ghosh-Mishra (2020)
Assadi-Chakrabarty-Khanna (2022)
Kupfer-Nisan (2022)
Addanki-McGregor-Musco (2022)
Dell-Lapinskas-Meeks (2024)
Adar-Hotam-Levi (2026)…

Rubinstein, Schramm, Weinberg

CUT ≡ ADD queries $S \ \mapsto \ |\partial S|$

Goal: Return minimum cut

Spl. case of (symmetric) submodular function minimization (SFM)

SFM can be solved in $\tilde{O}(n^2)$ queries

Jiang (2021), Chekuri-Quanrud (2021)

Efficiently, in $\tilde{O}(n^3)$ queries

Lee-Sidford-Wong (2015)

Best lower bound: $\tilde{\Omega}(\mathrm{n})$

C.-Graur-Jiang-Sidford (2022)

Very brief lay of the land with CUT queries

	Unweighted Undirected Graphs	Weighted Undirected Graphs	Unweighted Directed Graphs	Unweighted Hypergraphs
Connected? (mincut = 0 or not)	$O(n)$ [Apers et al 2022]	$O(n)$ [Liao, C. 2024]	$O(n^2/\log n)$ [folklore]	$O(n\log n)$ [exercise] $O(n)$ [Liao, C 2026]
(Global) Min-Cut	$O(n)$ [Apers et al 2022]	$\tilde{O}(n)$ [Mukhopadhyay-Nanongkai 2020]	$O(n^2/\log n)$ [folklore]	$\tilde{O}(n^2)$ [symmetric SFM]
(s,t) Min-Cut	$\tilde{O}(n^{3/2})$ [Kenneth-Mordoch, Krauthgamer, 2026]	$O(n^2)$ [learn the whole graph]	$O(n^2/\log n)$ [folklore]	$\tilde{O}(n^2)$ [SFM]

over to Louie and Sagnik…