On the Cut-Query Complexity of Approximating Max-Cut

Orestis Plevrakis, Seyoon Ragavan, Matt Weinberg

July 8, 2024

Max-Cut

- Undirected weighted graph G on $[n]$, edge weights $w_{i,j} \geq 0$
- Task: (c-approximately) find

$$\max_S v(S) = \max_S \sum_{i \in S} \sum_{j \in [n] \setminus S} w_{i,j}$$

Cut Query/Value Oracle Model

- Algorithm does not have direct access to the graph (e.g. an adjacency matrix)
- Instead: an oracle that takes $S \subseteq [n]$ as queries and informs the algorithm of $v(S)$

Cut Query/Value Oracle Model

- Algorithm does not have direct access to the graph (e.g. an adjacency matrix)
- Instead: an oracle that takes $S \subseteq [n]$ as queries and informs the algorithm of $v(S)$
- Algorithm's cost is the number of queries made; *it can perform unlimited local computation without penalty*

Comparison with the Standard Model

- Standard model:
  - [GW95]: poly(n) time algorithm achieving a 0.878-approximation
  - [KKMO07]: achieving a better approximation is NP-hard, assuming the Unique Games Conjecture

Comparison with the Standard Model

- Standard model:
  - [GW95]: poly(n) time algorithm achieving a 0.878-approximation
  - [KKMO07]: achieving a better approximation is NP-hard, assuming the Unique Games Conjecture
- In this model, $O(n^2)$ queries suffice even to exactly find the max cut!

Comparison with the Standard Model

- Standard model:
  - [GW95]: poly(n) time algorithm achieving a 0.878-approximation
  - [KKMO07]: achieving a better approximation is NP-hard, assuming the Unique Games Conjecture
- In this model, $O(n^2)$ queries suffice even to exactly find the max cut!
  - Observation:

$$w_{i,j} = \frac{1}{2}(v(\{i\}) + v(\{j\}) - v(\{i,j\})).$$

  - Algorithm: query all singletons and sets of size 2 to learn the graph, then brute force

Comparison with the Standard Model

- Standard model:
  - [GW95]: poly(n) time algorithm achieving a 0.878-approximation
  - [KKMO07]: achieving a better approximation is NP-hard, assuming the Unique Games Conjecture
- In this model, $O(n^2)$ queries suffice even to exactly find the max cut!
  - Observation:

$$w_{i,j} = \frac{1}{2}(v(\{i\}) + v(\{j\}) - v(\{i,j\})).$$

  - Algorithm: query all singletons and sets of size 2 to learn the graph, then brute force
- [GW95] does not imply any algorithm with $o(n^2)$ queries; needs access to the entire graph

Why This Model?

- Cut function is submodular $\Rightarrow$ special case of submodular function maximization

Why This Model?

- Cut function is submodular $\Rightarrow$ special case of submodular function maximization
- Many other graph problems have been studied in this model and other limited-access query models:

Why This Model?

- Cut function is submodular $\Rightarrow$ special case of submodular function maximization
- Many other graph problems have been studied in this model and other limited-access query models:
  - Problems: connectivity, bipartiteness, triangle detection ...
  - Models: linear measurements [AGM12, ACK21], matrix-vector products [SWYZ19], OR/XOR/AND queries [BvdBE+22]

Previous Work in the Cut-Query Model

- Learning the entire graph: $\Theta(m \log n / \log m)$ queries are necessary and sufficient [CK08, BM12, Cho13, BM15]

Previous Work in the Cut-Query Model

- Learning the entire graph: $\Theta(m \log n / \log m)$ queries are necessary and sufficient [CK08, BM12, Cho13, BM15]
- Exact min-cut:
  - [RSW18]: $\widetilde{O}(n)$ queries suffice in *unweighted* graphs
    - Proceeds by constructing a cut sparsifier of G, which our work extends to weighted graphs

Previous Work in the Cut-Query Model

- Learning the entire graph: $\Theta(m \log n / \log m)$ queries are necessary and sufficient [CK08, BM12, Cho13, BM15]
- Exact min-cut:
  - [RSW18]: $\widetilde{O}(n)$ queries suffice in *unweighted* graphs
  - [AEG+22]: improved this to $O(n)$

Previous Work in the Cut-Query Model

- Learning the entire graph: $\Theta(m \log n / \log m)$ queries are necessary and sufficient [CK08, BM12, Cho13, BM15]
- Exact min-cut:
  - [RSW18]: $\widetilde{O}(n)$ queries suffice in *unweighted* graphs
  - [AEG+22]: improved this to $O(n)$
  - [MN20]: $\widetilde{O}(n)$ queries suffice in weighted graphs

Previous Work in the Cut-Query Model

- Learning the entire graph: $\Theta(m \log n / \log m)$ queries are necessary and sufficient [CK08, BM12, Cho13, BM15]
- Exact min-cut:
  - [RSW18]: $\widetilde{O}(n)$ queries suffice in *unweighted* graphs
  - [AEG+22]: improved this to $O(n)$
  - [MN20]: $\widetilde{O}(n)$ queries suffice in weighted graphs
  - [GPRW20]: $\Omega(n)$ queries are necessary for weighted graphs
    - Uses a linear algebraic *cut dimension* argument, which we extend in one of our lower bounds

Our Results for Max-Cut

- Resolve most variants of this question up to factors of polylog(n)
- Three regimes: $c < 1/2$, $1/2 < c < 1$, and $c = 1$ (exact)

Our Results for Max-Cut

- Resolve most variants of this question up to factors of polylog(n)
- Three regimes: $c < 1/2$, $1/2 < c < 1$, and $c = 1$ (exact)
- Deterministic case with $c \in (1/2, 1)$ and randomised case with $c = 1$ are unresolved

c	Deterministic	Randomised
1	n^2	(n, n^2)
$(1/2, 1)$	(n, n^2)	n
$(0, 1/2)$	$\log n$	1

Our Results for Max-Cut

- Resolve most variants of this question up to factors of polylog(n)
- Three regimes: $c < 1/2$, $1/2 < c < 1$, and $c = 1$ (exact)
- Deterministic case with $c \in (1/2, 1)$ and randomised case with $c = 1$ are unresolved

c	Deterministic	Randomised
1	n^2	(n, n^2)
$(1/2, 1)$	(n, n^2)	n
$(0, 1/2)$	$\log n$	1

- Main results:
  1. $\Omega(n)$-query deterministic lower bound for $c > 1/2$
  2. $\widetilde{O}(n)$-query randomised algorithm for $c < 1$

Warm-Up: Cut Dimension Lower Bound for $c = 1$

Theorem

A deterministic algorithm requires at least $\Omega(n^2)$ queries to exactly find the value of the max cut.

- Directly adapts the cut dimension lower bound for min-cut in the query model by [GPRW20]

Warm-Up: Cut Dimension Lower Bound for $c = 1$

Theorem

A deterministic algorithm requires at least $\Omega(n^2)$ queries to exactly find the value of the max cut.

- Directly adapts the cut dimension lower bound for min-cut in the query model by [GPRW20]
- Adversary answers all queries as if the graph is K_n (all edge weights 1)

Warm-Up: Cut Dimension Lower Bound for $c = 1$

Theorem

A deterministic algorithm requires at least $\Omega(n^2)$ queries to exactly find the value of the max cut.

- Directly adapts the cut dimension lower bound for min-cut in the query model by [GPRW20]
- Adversary answers all queries as if the graph is K_n (all edge weights 1)
- Idea: at the end, there will be another graph that is consistent with all queries but has different max cut value to K_n

Cut Dimension Lower Bound: Setup

- Assign each cut S a *query vector* $v_S \in \mathbb{R}^{\binom{n}{2}}$ indexed over unordered pairs (i, j) of distinct vertices:

$$(v_S)_{i,j} = \begin{cases} 1, & (i,j) \text{ crosses the cut defined by } S, \\ 0, & \text{otherwise.} \end{cases}$$

Cut Dimension Lower Bound: Setup

- Assign each cut S a *query vector* $v_S \in \mathbb{R}^{\binom{n}{2}}$ indexed over unordered pairs (i, j) of distinct vertices:

$$(v_S)_{i,j} = \begin{cases} 1, & (i,j) \text{ crosses the cut defined by } S, \\ 0, & \text{otherwise.} \end{cases}$$

- Let $w \in \mathbb{R}^{\binom{n}{2}}$ be the vector containing all edge weights. Then:

$$v(S) = w^T v_S.$$

Cut Dimension Lower Bound: Setup

- $\mathbf{1} \in \mathbb{R}^{\binom{n}{2}}$ with all entries 1: the weight vector of K_n
- Let the other graph have weights $\mathbf{1} + z \in \mathbb{R}^{\binom{n}{2}}$. Non-negative weights $\Rightarrow$ require $z \geq -\mathbf{1}$.

Cut Dimension Lower Bound: Setup

- $\mathbf{1} \in \mathbb{R}^{\binom{n}{2}}$ with all entries 1: the weight vector of K_n
- Let the other graph have weights $\mathbf{1} + z \in \mathbb{R}^{\binom{n}{2}}$. Non-negative weights $\Rightarrow$ require $z \geq -\mathbf{1}$.
- Graphs defined by $\mathbf{1}, \mathbf{1} + z$ agree on a query Q when:

$$\mathbf{1}^T v_Q = (\mathbf{1} + z)^T v_Q \Leftrightarrow z^T v_Q = 0.$$

Cut Dimension Lower Bound: Setup

- $\mathbf{1} \in \mathbb{R}^{\binom{n}{2}}$ with all entries 1: the weight vector of K_n
- Let the other graph have weights $\mathbf{1} + z \in \mathbb{R}^{\binom{n}{2}}$. Non-negative weights $\Rightarrow$ require $z \geq -\mathbf{1}$.
- Graphs defined by $\mathbf{1}, \mathbf{1} + z$ agree on a query Q when:

$$\mathbf{1}^T v_Q = (\mathbf{1} + z)^T v_Q \Leftrightarrow z^T v_Q = 0.$$

  - Hence to be consistent with $\mathbf{1}$ on queries $Q_1, Q_2, \ldots, Q_q$, want z orthogonal to span$(v_{Q_1}, v_{Q_2}, \ldots, v_{Q_q})$

Cut Dimension Lower Bound: Setup

- $\mathbf{1} \in \mathbb{R}^{\binom{n}{2}}$ with all entries 1: the weight vector of K_n
- Let the other graph have weights $\mathbf{1} + z \in \mathbb{R}^{\binom{n}{2}}$. Non-negative weights $\Rightarrow$ require $z \geq -\mathbf{1}$.
- Graphs defined by $\mathbf{1}, \mathbf{1} + z$ agree on a query Q when:

$$\mathbf{1}^T v_Q = (\mathbf{1} + z)^T v_Q \Leftrightarrow z^T v_Q = 0.$$

  - Hence to be consistent with $\mathbf{1}$ on queries $Q_1, Q_2, \ldots, Q_q$, want z orthogonal to $\text{span}(v_{Q_1}, v_{Q_2}, \ldots, v_{Q_q})$
- Finally: need $\mathbf{1}, \mathbf{1} + z$ to have different max cut values

Cut Dimension Lower Bound: Setup

- $\mathbf{1} \in \mathbb{R}^{\binom{n}{2}}$ with all entries 1: the weight vector of K_n
- Let the other graph have weights $\mathbf{1} + z \in \mathbb{R}^{\binom{n}{2}}$. Non-negative weights $\Rightarrow$ require $z \geq -\mathbf{1}$.
- Graphs defined by $\mathbf{1}, \mathbf{1} + z$ agree on a query Q when:

$$\mathbf{1}^T v_Q = (\mathbf{1} + z)^T v_Q \Leftrightarrow z^T v_Q = 0.$$

  - Hence to be consistent with $\mathbf{1}$ on queries $Q_1, Q_2, \ldots, Q_q$, want z orthogonal to $\text{span}(v_{Q_1}, v_{Q_2}, \ldots, v_{Q_q})$
- Finally: need $\mathbf{1}, \mathbf{1} + z$ to have different max cut values
  - Suffices for there to exist a max cut C of K_n such that $(\mathbf{1} + z)^T v_C > \mathbf{1}^T v_C \Leftrightarrow z^T v_C > 0$.

Cut Dimension Lower Bound: Setup

- $\mathbf{1} \in \mathbb{R}^{\binom{n}{2}}$ with all entries 1: the weight vector of K_n
- Let the other graph have weights $\mathbf{1} + z \in \mathbb{R}^{\binom{n}{2}}$. Non-negative weights $\Rightarrow$ require $z \geq -\mathbf{1}$.
- Graphs defined by $\mathbf{1}, \mathbf{1} + z$ agree on a query Q when:

$$\mathbf{1}^T v_Q = (\mathbf{1} + z)^T v_Q \Leftrightarrow z^T v_Q = 0.$$

  - Hence to be consistent with $\mathbf{1}$ on queries $Q_1, Q_2, \ldots, Q_q$, want z orthogonal to $\text{span}(v_{Q_1}, v_{Q_2}, \ldots, v_{Q_q})$
- Finally: need $\mathbf{1}, \mathbf{1} + z$ to have different max cut values
  - Suffices for there to exist a max cut C of K_n such that $(\mathbf{1} + z)^T v_C > \mathbf{1}^T v_C \Leftrightarrow z^T v_C > 0$.
- If there exists $z \in \mathbb{R}^{\binom{n}{2}}$ satisfying all highlighted constraints, a deterministic algorithm cannot exactly identify the value of the max cut.

Defining the Cut Dimension

Definition

Define the *max cut space* of a graph G to be:

$$\text{span}(\{v_U : U \text{ a max cut of } G\})$$

The *max cut dimension* of G is the dimension of its max cut space.

Cut Dimension Lower Bound: Proof

Lemma ([GPRW20])

A deterministic algorithm that exactly finds the max cut needs at least as many queries as the max cut dimension d of K_n.

Proof.

- Assume $q < d$. Then we can find nonzero z in the max cut space of K_n that is orthogonal to $\text{span}(v_{Q_1}, \ldots, v_{Q_q})$. There must exist a max cut C such that $z^T v_C \neq 0$.

Cut Dimension Lower Bound: Proof

Lemma ([GPRW20])

A deterministic algorithm that exactly finds the max cut needs at least as many queries as the max cut dimension d of K_n.

Proof.

- Assume $q < d$. Then we can find nonzero z in the max cut space of K_n that is orthogonal to $\text{span}(v_{Q_1}, \ldots, v_{Q_q})$. There must exist a max cut C such that $z^T v_C \neq 0$.
- Scale z so that $z \geq -\mathbf{1}$ and $z^T v_C > 0$.

Cut Dimension Lower Bound: Proof

Lemma ([GPRW20])

A deterministic algorithm that exactly finds the max cut needs at least as many queries as the max cut dimension d of K_n.

Proof.

- Assume $q < d$. Then we can find nonzero z in the max cut space of K_n that is orthogonal to $\text{span}(v_{Q_1}, \ldots, v_{Q_q})$. There must exist a max cut C such that $z^T v_C \neq 0$.
- Scale z so that $z \geq -\mathbf{1}$ and $z^T v_C > 0$.

□

Max cut dimension of K_n is $\Omega(n^2) \Rightarrow$ lower bound follows.

Extending Cut Dimension Argument to $c > 1/2$

Theorem

For $c > 1/2$, a deterministic algorithm that estimates the value of the max cut within a factor of c requires at least $\Omega(n)$ queries.

- Again, adversary answers according to K_n

Extending Cut Dimension Argument to $c > 1/2$

Theorem

For $c > 1/2$, a deterministic algorithm that estimates the value of the max cut within a factor of c requires at least $\Omega(n)$ queries.

- Again, adversary answers according to K_n
- Want to find a perturbation z such that $\mathbf{1} + z$ has max cut value at least a factor of $1/c$ more than K_n.

$c > 1/2$ Deterministic Lower Bound: Constraints on z

- $\mathbf{1} + z$ must have non-negative edge weights:

$$z \geq -\mathbf{1}.$$

- $\mathbf{1} + z$ must agree with $\mathbf{1}$ on all queries:

$$z^T v_{Q_i} = 0 \forall i.$$

$c > 1/2$ Deterministic Lower Bound: Constraints on z

- $\mathbf{1} + z$ must have non-negative edge weights:

$$z \geq -\mathbf{1}.$$

- $\mathbf{1} + z$ must agree with $\mathbf{1}$ on all queries:

$$z^T v_{Q_i} = 0 \forall i.$$

- There must be a max cut (or just a *near max cut* where each side has $n/2 + o(n)$ vertices) C such that $(\mathbf{1} + z)^T v_C$ is much larger than $\mathbf{1}^T v_C$:

$$z^T v_C \geq \frac{\delta^2 n^2}{4},$$

where $\delta \in (0, 1)$ depends on c.

$c > 1/2$ Deterministic Lower Bound: Simplified LP

- Work over $\mathbb{R}^n$, indexed by vertices. Assign to each cut S a vector $u_S \in \mathbb{R}^n$ that is 1 when $i \in S$ and -1 otherwise.

$c > 1/2$ Deterministic Lower Bound: Simplified LP

- Work over $\mathbb{R}^n$, indexed by vertices. Assign to each cut S a vector $u_S \in \mathbb{R}^n$ that is 1 when $i \in S$ and -1 otherwise.
- It turns out that it suffices to find a near max cut C and $y \in \mathbb{R}^n$ satisfying the following:
  - $-\mathbf{1} \leq y \leq \mathbf{1}$
  - $y^T \mathbf{1} = 0$ and $y^T u_{Q_i} = 0$ for all i
  - $y^T u_C \geq \delta n$

$c > 1/2$ Deterministic Lower Bound: Simplified LP

- Work over $\mathbb{R}^n$, indexed by vertices. Assign to each cut S a vector $u_S \in \mathbb{R}^n$ that is 1 when $i \in S$ and -1 otherwise.
- It turns out that it suffices to find a near max cut C and $y \in \mathbb{R}^n$ satisfying the following:
  - $-\mathbf{1} \leq y \leq \mathbf{1}$
  - $y^T \mathbf{1} = 0$ and $y^T u_{Q_i} = 0$ for all i
  - $y^T u_C \geq \delta n$
- Take the dual of this LP $\Rightarrow$ we need to show that there exists a near max cut C such that

$$\min_{u \in V} ||u - u_C||_1 \geq \delta n$$

where $V = \text{span}(\mathbf{1}, u_{Q_1}, \ldots, u_{Q_q})$.

$c > 1/2$ Deterministic Lower Bound: Simplified LP

- Work over $\mathbb{R}^n$, indexed by vertices. Assign to each cut S a vector $u_S \in \mathbb{R}^n$ that is 1 when $i \in S$ and -1 otherwise.
- It turns out that it suffices to find a near max cut C and $y \in \mathbb{R}^n$ satisfying the following:
  - $-\mathbf{1} \leq y \leq \mathbf{1}$
  - $y^T \mathbf{1} = 0$ and $y^T u_{Q_i} = 0$ for all i
  - $y^T u_C \geq \delta n$
- Take the dual of this LP $\Rightarrow$ we need to show that there exists a near max cut C such that

$$\min_{u \in V} ||u - u_C||_1 \geq \delta n$$

  where $V = \text{span}(\mathbf{1}, u_{Q_1}, \ldots, u_{Q_q})$.
- Proof idea: consider an ϵ-net of points in V with respect to the ℓ_1 metric.

Randomised Algorithm for $c < 1$

We will use *cut sparsifiers:*

Definition

Given weighted graphs G, H on the same set of n vertices with non-negative weights, we say H is an ϵ-sparsifier of G if all of the following conditions hold:

- H has $\widetilde{O}_\epsilon(n)$ edges with nonzero weight.
- For any cut $S \subseteq [n]$, we have $(1-\epsilon)v(S;G) \leq v(S;H) \leq (1+\epsilon)v(S;G)$.

Here $v(S;G)$ denotes the value of the cut defined by S on the graph G.

Algorithm: compute an ϵ-sparsifier, then find the max cut of the sparsifier by running brute force locally.

Sparsification in the Cut Query Model: Results

Theorem ([RSW18], based on ideas by [BK15])

For unweighted G, we can construct an ϵ-sparsifier in $\widetilde{O}(n/\epsilon^2)$ queries.

Sparsification in the Cut Query Model: Results

Theorem ([RSW18], based on ideas by [BK15])

For unweighted G, we can construct an ϵ-sparsifier in $\widetilde{O}(n/\epsilon^2)$ queries.

Theorem (This work, directly adapting [RSW18] to weighted graphs)

For weighted G with weights in $[1, W]$, we can construct an ϵ-sparsifier in $\widetilde{O}(n \log W/\epsilon^2)$ queries.

Sparsification in the Cut Query Model: Results

Theorem ([RSW18], based on ideas by [BK15])

For unweighted G, we can construct an ϵ-sparsifier in $\widetilde{O}(n/\epsilon^2)$ queries.

Theorem (This work, directly adapting [RSW18] to weighted graphs)

For weighted G with weights in $[1, W]$, we can construct an ϵ-sparsifier in $\widetilde{O}(n \log W/\epsilon^2)$ queries.

Theorem (This work, using additional ideas by [BK15])

For weighted G, we can construct an ϵ-sparsifier in $\widetilde{O}(n/\epsilon^2)$ queries (regardless of the value of W).

Sparsification: Naive Attempts

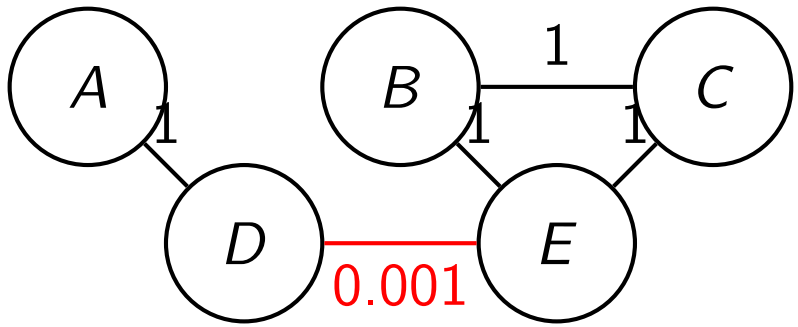
Idea: randomly subsample edges and use a Chernoff bound.

- Attempt 1: randomly sample each edge with probability proportionate to $w_{i,j}$.

Sparsification: Naive Attempts

Idea: randomly subsample edges and use a Chernoff bound.

- Attempt 1: randomly sample each edge with probability proportionate to $w_{i,j}$. Issue: could miss bridges of low weight.



Sparsification: Edge Strength-Based Subsampling [BK15]

For an edge $e = (i, j)$, define the *edge strength* of e as the maximum min cut over all vertex-induced subgraphs containing e:

$$k_e = \max_{S \subseteq V : i,j \in S} \text{MinCut}(G[S]).$$

Sparsification: Edge Strength-Based Subsampling [BK15]

For an edge $e = (i, j)$, define the *edge strength* of e as the maximum min cut over all vertex-induced subgraphs containing e:

$$k_e = \max_{S \subseteq V : i,j \in S} \text{MinCut}(G[S]).$$

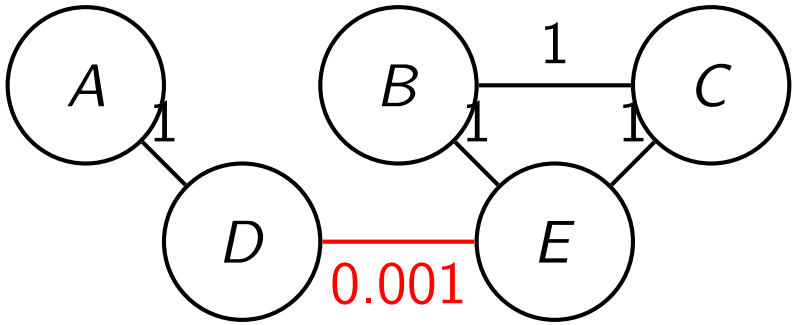
Algorithm by [BK15]: sample e with probability proportionate to w_e/k_e.

Sparsification: Edge Strength-Based Subsampling [BK15]

For an edge $e = (i, j)$, define the *edge strength* of e as the maximum min cut over all vertex-induced subgraphs containing e:

$$k_e = \max_{S \subseteq V : i,j \in S} \text{MinCut}(G[S]).$$

Algorithm by [BK15]: sample e with probability proportionate to w_e / k_e.



Sparsification: Estimating Edge Strengths [RSW18]

- Barrier to using [BK15] in the cut-query model: estimating edge strengths in a graph we don't know!

Sparsification: Estimating Edge Strengths [RSW18]

- Barrier to using [BK15] in the cut-query model: estimating edge strengths in a graph we don't know!
- [RSW18]: shows how to do this by subsampling G at $O(\log W + \log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n \log W/\epsilon^2)$ queries.

Sparsification: Estimating Edge Strengths [RSW18]

- Barrier to using [BK15] in the cut-query model: estimating edge strengths in a graph we don't know!
- [RSW18]: shows how to do this by subsampling G at $O(\log W + \log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n \log W/\epsilon^2)$ queries.
  - Intuition: edge strengths take values in $[1, n^2 W]$, and we iterate through all powers of 2 in this range

Sparsification: Estimating Edge Strengths [RSW18]

- Barrier to using [BK15] in the cut-query model: estimating edge strengths in a graph we don't know!
- [RSW18]: shows how to do this by subsampling G at $O(\log W + \log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n \log W/\epsilon^2)$ queries.
  - Intuition: edge strengths take values in $[1, n^2 W]$, and we iterate through all powers of 2 in this range
- This work: only subsample at $O(\log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n/\epsilon^2)$ queries.

Sparsification: Estimating Edge Strengths [RSW18]

- Barrier to using [BK15] in the cut-query model: estimating edge strengths in a graph we don't know!
- [RSW18]: shows how to do this by subsampling G at $O(\log W + \log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n \log W/\epsilon^2)$ queries.
  - Intuition: edge strengths take values in $[1, n^2 W]$, and we iterate through all powers of 2 in this range
- This work: only subsample at $O(\log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n/\epsilon^2)$ queries.
  - Estimate edge strengths within a factor of poly(n) using $\widetilde{O}(n)$ queries
    - Uses a very weak Kruskal-style algorithm and results by [BK15] relating edge strengths to maximum spanning trees

Sparsification: Estimating Edge Strengths [RSW18]

- Barrier to using [BK15] in the cut-query model: estimating edge strengths in a graph we don't know!
- [RSW18]: shows how to do this by subsampling G at $O(\log W + \log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n \log W/\epsilon^2)$ queries.
  - Intuition: edge strengths take values in $[1, n^2 W]$, and we iterate through all powers of 2 in this range
- This work: only subsample at $O(\log n)$ different resolutions $\Rightarrow$ sparsification in $\widetilde{O}(n/\epsilon^2)$ queries.
  - Estimate edge strengths within a factor of $\text{poly}(n)$ using $\widetilde{O}(n)$ queries
  - Incorporate this information into the [RSW18] algorithm $\Rightarrow$ no longer need to search through all $O(\log W + \log n)$ resolutions

Open Direction: Deterministic Sparsifiers?

- Success in designing deterministic sparsification algorithms in the classical model $\Rightarrow$ might hope for these to be adaptable to the cut query model
- This appears to be difficult; the deterministic sparsification algorithm by [BSS14] is linear algebraic and relies heavily on having access to the graph's Laplacian matrix

Thank you!

References I

[ACK21] Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph connectivity and single element recovery via linear and OR queries. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[AEG+22] Simon Apers, Yuval Efron, Pawel Gawrychowski, Troy Lee, Sagnik Mukhopadhyay, and Danupon Nanongkai. Cut query algorithms with star contraction. *CoRR*, abs/2201.05674, 2022.

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012.

[BK15] András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015.

References II

[BM12] Nader H. Bshouty and Hanna Mazzawi. Toward a deterministic polynomial time algorithm with optimal additive query complexity. *Theor. Comput. Sci.*, 417:23–35, 2012.

[BM15] Nader H. Bshouty and Hanna Mazzawi. On parity check (0, 1)-matrix over F_p. *SIAM J. Discret. Math.*, 29(1):631–657, 2015.

[BSS14] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Rev.*, 56(2):315–334, 2014.

[BvdBE+22] Joakim Blikstad, Jan van den Brand, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Nearly optimal communication and query complexity of bipartite matching. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1174–1185. IEEE, 2022.

[Cho13] Sung-Soon Choi. Polynomial time optimal query algorithms for finding graphs with arbitrary real weights. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 797–818, Princeton, NJ, USA, 12–14 Jun 2013. PMLR.

References III

[CK08] Sung-Soon Choi and Jeong Han Kim. Optimal query complexity bounds for finding graphs. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 749–758. ACM, 2008.

[GPRW20] Andrei Graur, Tristan Pollner, Vikram Ramaswamy, and S. Matthew Weinberg. New query lower bounds for submodular function minimization. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 64:1–64:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[GW95] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

References IV

[MN20] Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 496–509. ACM, 2020.

[RSW18] Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[SWYZ19] Xiaoming Sun, David P. Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 94:1–94:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.