Graph Partitioning: Beyond Worst-Case Analysis

Rakesh Venkat

(Indian Institute of Technology, Hyderabad)

CALDAM Pre-Conference School, IIT Hyderabad.

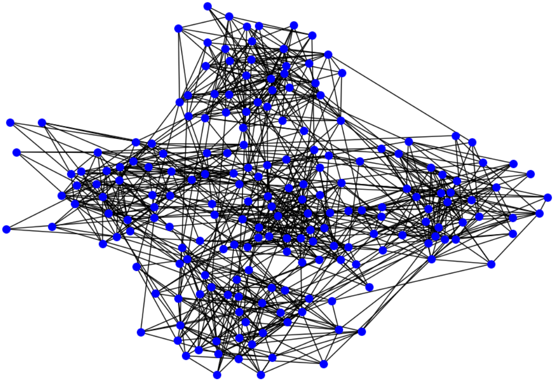
Feb 2020.

Overview

Outline

Graph Partitioning

Aim: Break an input graph $G = (V, E)$ into two or more parts, while optimizing some function that measures the partition quality.



Practical Applications

1. Community detection
2. Routing network flows, e.g. traffic
3. Image Processing and Graphics
4. Biological Networks, e.g. protein-protein interactions
5. Detecting influential/anomalous nodes in Social Networks
6. Epidemic spreading

Practical Applications

1. Community detection
2. Routing network flows, e.g. traffic
3. Image Processing and Graphics
4. Biological Networks, e.g. protein-protein interactions
5. Detecting influential/anomalous nodes in Social Networks
6. Epidemic spreading

Many more applications..

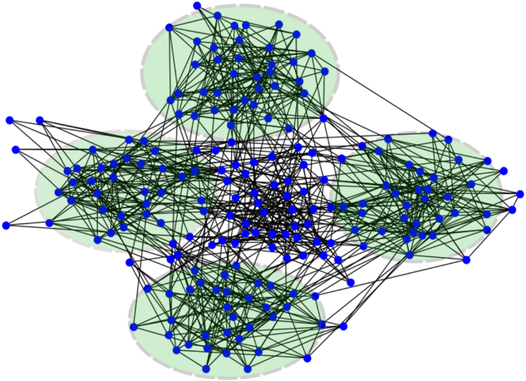
Various objectives

Given the vast number of applications, there are many different objectives one could consider:

- Min-Bisection
- Max-Bisection
- Sparsest Cut/Edge Expansion.
- Sparsest Vertex Cut/Vertex Expansion.
- Multiway Cut
- Approximate Coloring
- .. (Many variants of the above)..

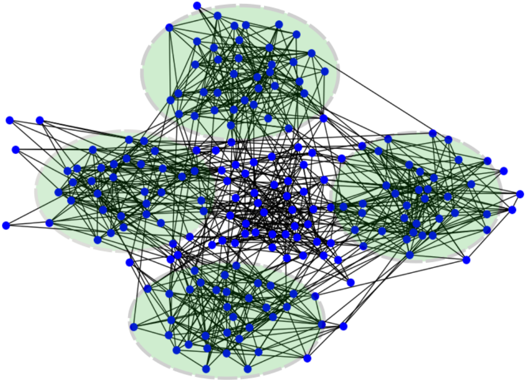
Solving graph partitioning

Most of these problems are NP-hard to compute exactly, or even approximate well in general. However, inputs in practice are not worst-case.



Solving graph partitioning

Most of these problems are NP-hard to compute exactly, or even approximate well in general. However, inputs in practice are not worst-case.



Understanding these classes also gives us insights into the general case.

Worst-Case Analysis

- Consider a minimization objective that is NP-hard (e.g. Min-Bisection).
- Design an algorithm such that:

$$\boxed{\mathrm{ALG}(G) \;\leq\; C \cdot \mathrm{OPT}(G) \qquad \text{for every graph } G}$$

- Would like as small a value for C as possible (Ideal: $C = 1$).

Pros:

Worst-Case Analysis

- Consider a minimization objective that is NP-hard (e.g. Min-Bisection).
- Design an algorithm such that:

$$\boxed{\mathrm{ALG}(G) \;\le\; C \cdot \mathrm{OPT}(G) \qquad \text{for every graph } G}$$

- Would like as small a value for C as possible (Ideal: $C = 1$).

Pros:

- Many clever algorithms have been designed in this framework.
- Often the algorithms work well in practice too.

Worst-Case Analysis

- Consider a minimization objective that is NP-hard (e.g. Min-Bisection).
- Design an algorithm such that:

$$\boxed{\mathrm{ALG}(G) \;\leq\; C \cdot \mathrm{OPT}(G) \qquad \text{for every graph } G}$$

- Would like as small a value for C as possible (Ideal: $C = 1$).

Pros:

- Many clever algorithms have been designed in this framework.
- Often the algorithms work well in practice too.

Cons:

- Pessimistic estimates on algorithm's performance.
- Do not know why the algorithms work well in practice. In many real-life cases, simpler algorithms perform better.
- How do we account for data (e.g. Machine-Learning applications like clustering?)

Beyond Worst-Case Analysis

- Come up with a description of a class of instances that arise in practice.
- Design new algorithms, or analyze known ones on such a class.
    - Expect that these will give better guarantees than the worst-case.

Beyond Worst-Case Analysis

- Come up with a description of a class of instances that arise in practice.
- Design new algorithms, or analyze known ones on such a class.
  - Expect that these will give better guarantees than the worst-case.
- Clearly, no single description will cover all applications. Many models have been explored.

Beyond Worst-Case Analysis: Scenarios

1. Stability of Instances
   - Clustering (Approximation Stability)[BBG09]
   - Bilu-Linial Stability (Max-Cut, Multiway Cut) [BL10, MMV14]

BBG09: Balcan-Blum-Gupta, MMV*: Makarychev-Makarychev-Vijayaraghavan

FK00: Feige-Krauthgamer, BCLS92: Bui-Chaudhari-Leighton-Sipser, FK01: Feige-Kilian, ABH15: Abbe-Bandeira-Hall, BS95: Blum-Spencer

ST01: Spielman-Teng, AV06: Arthur-Vassilvitskii, AMR11: Arthur-Manthey-Roglin

Beyond Worst-Case Analysis: Scenarios

1. Stability of Instances
   - Clustering (Approximation Stability)[BBG09]
   - Bilu-Linial Stability (Max-Cut, Multiway Cut) [BL10, MMV14]
2. Random/Semi-Random and Planted Models
   - Planted Clique [FK00]
   - Graph Bisection [BCLS92, FK01, McSherry01, ABH15 ...]
   - Edge Expansion [BS95, MMV12, MMV14]

BBG09: Balcan-Blum-Gupta, MMV*: Makarychev-Makarychev-Vijayaraghavan

FK00: Feige-Krauthgamer, BCLS92: Bui-Chaudhari-Leighton-Sipser, FK01: Feige-Kilian, ABH15: Abbe-Bandeira-Hall, BS95: Blum-Spencer

ST01: Spielman-Teng, AV06: Arthur-Vassilvitskii, AMR11: Arthur-Manthey-Roglin

Beyond Worst-Case Analysis: Scenarios

1. Stability of Instances
    - Clustering (Approximation Stability)[BBG09]
    - Bilu-Linial Stability (Max-Cut, Multiway Cut) [BL10, MMV14]
2. Random/Semi-Random and Planted Models
    - Planted Clique [FK00]
    - Graph Bisection [BCLS92, FK01, McSherry01, ABH15 ...]
    - Edge Expansion [BS95, MMV12, MMV14]
3. Smoothed Analysis
    - Simplex Method for LPs [ST01]
    - Local Search [AV06, AMR11, ...]

BBG09: Balcan-Blum-Gupta, MMV*: Makarychev-Makarychev-Vijayaraghavan

FK00: Feige-Krauthgamer, BCLS92: Bui-Chaudhari-Leighton-Sipser, FK01: Feige-Kilian, ABH15: Abbe-Bandeira-Hall, BS95: Blum-Spencer

ST01: Spielman-Teng, AV06: Arthur-Vassilvitskii, AMR11: Arthur-Manthey-Roglin

Beyond Worst-Case Analysis: Scenarios

1. Stability of Instances
   - Clustering (Approximation Stability)[BBG09]
   - Bilu-Linial Stability (Max-Cut, Multiway Cut) [BL10, MMV14]

2. Random/Semi-Random and Planted Models
   - Planted Clique [FK00]
   - Graph Bisection [BCLS92, FK01, McSherry01, ABH15 ...]
   - Edge and Vertex Expansion [BS95, MMV12, MMV14]

3. Smoothed Analysis
   - Simplex Method for LPs [ST01]
   - Local Search [AV06, AMR11, ...]

4. Other Hybrid or Distribution-Free models

BBG09: Balcan-Blum-Gupta, MMV*: Makarychev-Makarychev-Vijayaraghavan

FK00: Feige-Krauthgamer, BCLS92: Bui-Chaudhari-Leighton-Sipser, FK01: Feige-Kilian, ABH15: Abbe-Bandeira-Hall, BS95: Blum-Spencer

ST01: Spielman-Teng, AV06: Arthur-Vassilvitskii, AMR11: Arthur-Manthey-Roglin

Planted and Semi-Random Models

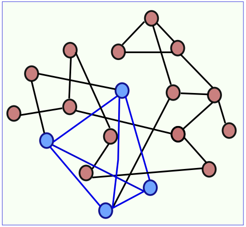
- Semi-Random Models generate inputs via a combination of randomness and adversarial changes.
  - The algorithm designer may know the model of generation of inputs. However, the adversarial changes will keep things difficult.
- In a Planted Model, input graphs are promised to have a solution planted (e.g., a small cut or bisection). However, the rest of the graph can be completely adversarial.
- **Goal:** Recover a planted or close-to-optimal solution with high probability over the input distribution, irrespective of adversarial changes.
- Well-studied problems in such models: (2-way) Edge expansion, Coloring, Planted Clique. [(BS '95), (FK '01), (MMV '12), (MMV '14)]

[BS95]:Blum-Spencer, [FK01]:Feige-Kilian, [MMV*]:Makarychev-Makarychev-Vijayaraghavan.

Outline

The Maximum Clique Problem

- The Maximum Clique Problem: Given a undirected graph $G = (V, E)$, find the largest clique present in G.



- Extremely Hard to solve: for any $\varepsilon > 0$, getting a $n^{1-\varepsilon}$ approximation is NP-Hard!

The Planted Clique Problem: Model

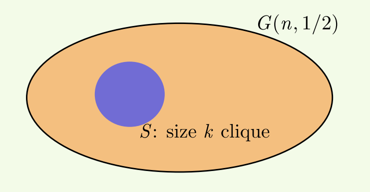
1. What happens in an Erdős-Rényi graph $G(n, \frac{1}{2})$?
   - The size of a maximum clique is $2\log_2 n$ with high probability.
   - Proof hack: $\mathbb{E}[\text{No. of cliques of size } k \text{ in } G] \approx \binom{n}{k} 2^{-k^2/2}$. This is 1 when $k \approx 2\log_2 n$.

The Planted Clique Problem: Model

1. What happens in an Erdős-Rényi graph $G(n, \frac{1}{2})$?
   - The size of a maximum clique is $2 \log_2 n$ with high probability.
   - Proof hack: $\mathbb{E}[\text{No. of cliques of size } k \text{ in } G] \approx \binom{n}{k} 2^{-k^2/2}$. This is 1 when $k \approx 2 \log_2 n$.
2. Can we find a clique of size $2 \log_2 n$ w.h.p?
   - Can only find one of size $\approx \log_2 n$. Simple heuristics achieve it.

The Planted Clique Model

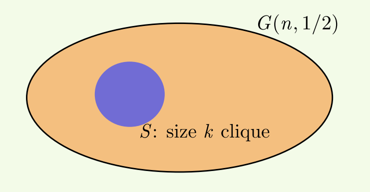
1. What if we plant a clique of size k in this graph: Choose a subset $S \subseteq V$, and add all edges within S to the graph? Remaining part of the graph is generated according to $G(n, 0.5)$.



2. If $k < 2\log_2 n$, then S is not the max-clique, so we can not expect to find it.

The Planted Clique Model

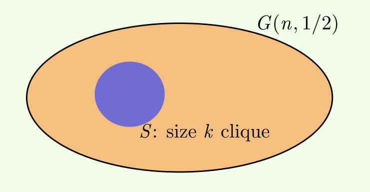
1. What if we plant a clique of size k in this graph: Choose a subset $S \subseteq V$, and add all edges within S to the graph? Remaining part of the graph is generated according to $G(n, 0.5)$.



2. If $k < 2\log_2 n$, then S is not the max-clique, so we can not expect to find it.

The Planted Clique Model

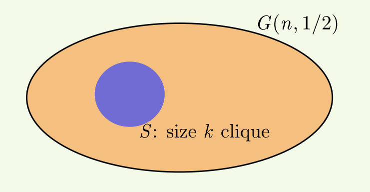
1. What if we plant a clique of size k in this graph: Choose a subset $S \subseteq V$, and add all edges within S to the graph? Remaining part of the graph is generated according to $G(n, 0.5)$.



2. If $k < 2\log_2 n$, then S is not the max-clique, so we can not expect to find it.
3. If $k > \sqrt{n \log_2 n}$, then can find S w.h.p.
   - If $v \notin S$: $\deg(v) \in [n/2 - c\sqrt{n \log_2 n}, n/2 + c\sqrt{n \log_2 n}]$, with high probability.

The Planted Clique Model

1. What if we plant a clique of size k in this graph: Choose a subset $S \subseteq V$, and add all edges within S to the graph? Remaining part of the graph is generated according to $G(n, 0.5)$.



2. If $k < 2\log_2 n$, then S is not the max-clique, so we can not expect to find it.
3. If $k > \sqrt{n \log_2 n}$, then can find S w.h.p.
   - If $v \notin S$: $\deg(v) \in [n/2 - c\sqrt{n \log_2 n}, n/2 + c\sqrt{n \log_2 n}]$, with high probability.
   - If $v \in S$, $\deg(v) \approx n/2 + k$. If $k \geq 4c\sqrt{n \lg n}$, then the highest degree vertices will contain S w.h.p.

Planted Clique: $k = \Theta(\sqrt{n})$

1. Above degree counting does not work when $k = \Theta(\sqrt{n})$ (why?)
2. We have to resort to more involved techniques: a Spectral Algorithm.
   - Use linear algebraic properties of the adjacency matrix of G.
3. Consider the adjacency matrix of G, compute the second eigenvector v. Let A be the largest k coordinates of v. Return $B = \{i \in V : |N_A(i)| \geq 3k/4\}$.

Theorem ([AKS98])

When $k \geq \sqrt{n}$, the above algorithm recovers S exactly w.h.p.

AKS98: Alon-Kriveilevich-Sudakov98

Planted Clique: $k = \Theta(\sqrt{n})$: Key Idea

1. Since G is random, its adjacency matrix is random.
2. Key Idea: The expected adjacency matrix of G looks like:

$$\mathbb{E}[A] = \left(\begin{array}{cc|c} 1 & 1 & \multirow{2}{*}{\mathbf{0.5}} \\ 1 & \ddots & \\ \hline \multicolumn{2}{c|}{\mathbf{0.5}} & \mathbf{0.5} \end{array}\right)$$

Planted Clique: $k = \Theta(\sqrt{n})$: Key Idea

1. Since G is random, its adjacency matrix is random.
2. Key Idea: The expected adjacency matrix of G looks like:

$$\mathbb{E}[A] = \left(\begin{array}{c|c} \begin{matrix} 1 & 1 \\ 1 & \ddots \end{matrix} & \mathbf{0.5} \\ \hline \mathbf{0.5} & \mathbf{0.5} \end{array}\right)$$

3. The second eigenvector of this matrix is approximately:
$(\underbrace{n-k, n-k, \ldots, n-k}_{k \text{ times}}, -k, -k, \ldots - k)$.

Planted Clique: $k = \Theta(\sqrt{n})$: Key Idea

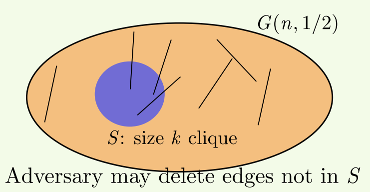
1. Since G is random, its adjacency matrix is random.
2. Key Idea: The expected adjacency matrix of G looks like:

$$\mathbb{E}[A] = \left(\begin{array}{cc|c} 1 & 1 & \multirow{2}{*}{\textbf{0.5}} \\ 1 & \ddots & \\ \hline \textbf{0.5} & & \textbf{0.5} \end{array}\right)$$

3. The second eigenvector of this matrix is approximately: $(\underbrace{n-k, n-k, \ldots, n-k}_{k \text{ times}}, -k, -k, \ldots -k)$.
4. Using random matrix theory, show that the eigenvector of the actual adjacency matrix is not far from this ideal w.h.p.

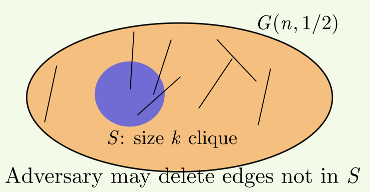
Planted Clique: with monotone adversary

1. Suppose an adversary comes along and:
   - Only deletes some edges that are not completely within S (adversarially).



Planted Clique: with monotone adversary

1. Suppose an adversary comes along and:
   - Only deletes some edges that are not completely within S (adversarially).



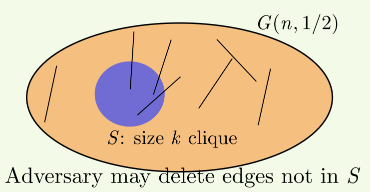
2. Intuitively the problem is now easier, as the clique stands out more.

3. However, we cannot use the expected adjacency matrix anymore for a spectral algorithm!

FK00: Feige-Krauthgamer

Planted Clique: with monotone adversary

1. Suppose an adversary comes along and:
   - Only deletes some edges that are not completely within S (adversarially).



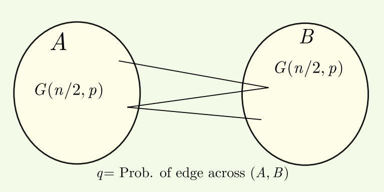
2. Intuitively the problem is now easier, as the clique stands out more.

3. However, we cannot use the expected adjacency matrix anymore for a spectral algorithm!

4. Use Semidefinite Programming Relaxations [FK00]. These are even more 'robust' then spectral algorithms.

FK00: Feige-Krauthgamer

Planted Bisection (Stochastic Block Model)



Assume: $p > q$.

- When $p - q = \Omega(1)$: Degree counting works.
- Say $p = a \log n/n$, and $q = b \log n/n$. If $(\sqrt{a} - \sqrt{b}) \geq \sqrt{2}$, can use spectral (for purely random) or SDP (for semi-random) algorithms for recovery. [..., ABH14, MNS14, WXH15, Ban15].
- Not recoverable if $(\sqrt{a} - \sqrt{b}) \leq \sqrt{2}$.

ABH14: Abbe-Bandeira-Hall, MNS14:Mossel-Neeman-Sly, WXH15: Wu-Xu-Hajek, Ban15: Bandeira

Outline

Problems we consider

1. *k-way Edge Expansion*: Partition an input graph into exactly k parts, while minimizing the maximum edge-expansion.

2. *k-way Vertex Expansion*: Partition an input graph into exactly k parts, while minimizing the maximum vertex-expansion.

- Edge and vertex expansion are qualitatively different problems. Less work on vertex expansion.

Planted Models

- Planted Models assume that the input graphs come with a planted solution:
  - G is guaranteed to have a k-way partition with low k-way edge (or vertex) expansion.
- **Goal:** Recover solution guaranteed to be a good approximation of the planted solution.

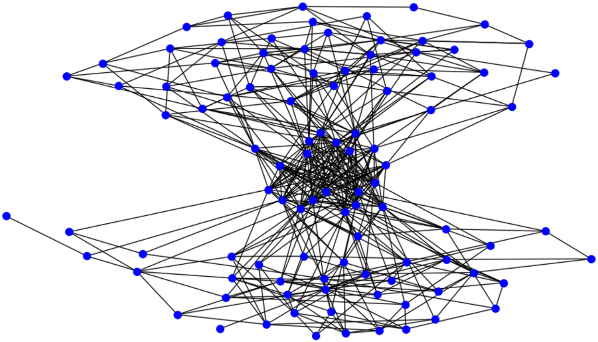
This Talk: k-way Vertex-Expansion objective.

(Results mentioned in this section are based on joint work with Anand Louis, IISc Bangalore)

Sparse Vertex-Cuts

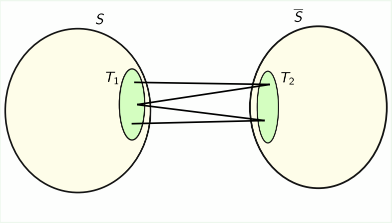
Sparse Vertex-Cuts

- Edge density across a cut alone may not always be the right indicator of partition sparsity.



- Graph communities may interact heavily, but via just a small number of influential nodes.
  - For example, these may be hubs in the network.

The Vertex-Expansion objective



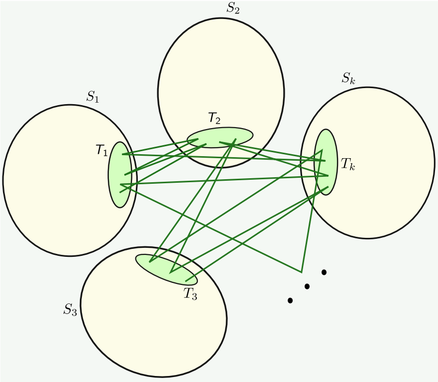
- Φ^V measures sparsity via the number of vertices on the boundary of a cut $(S, \overline{S})$

$$\Phi^V(S) = |V| \frac{|N(S)| + |N(\overline{S})|}{|S| \, |\overline{S}|}$$

  - In the above figure, if $|S| = n/2$ and $|T_i| = \varepsilon n/2$, then $\Phi^V(S) = 4\varepsilon$.
- Vertex Expansion of G:

$$\boxed{\Phi^V(G) = \min_{S \subseteq V} \Phi^V(S)}$$

k-way Vertex-Expansion objective



$$\Phi^{V,k}(G) := \min_{\{S_1,\ldots,S_k\}\in\mathcal{P}_k} \max_{i\in[k]} \Phi^V(S_i)$$

- Above, $\mathcal{P}_k$ is the set of all k-partitions of the vertex set V.
- In the figure, if $|T_i| = \varepsilon n/k$, and $|S| = n/k$, then $\Phi^V(S_i) = \varepsilon k/(1-1/k) \leq 2\varepsilon k$.

Known Results for Sparse Vertex-Cuts

Vertex Expansion/Cuts less well-understood as compared to Edge Expansion/Sparsest Cut.

Algorithms:

- ($k = 2$) [FHL '08] : $O(\sqrt{\log n})$-approximation algorithm, using ℓ_1 line embeddings.
- ($k = 2$) [LRV '13]: $O(\sqrt{\log d/\mathrm{OPT}})$-approximation algorithm, where d is max-degree.
- ($k \geq 2$) Can infer from [CLTZ '18, LM '16]: $O(\sqrt{\log n} \cdot OPT \cdot f(k))$.

[FHL08]: Feige-Hajiaghayi-Lee, [LRV13]: Louis-Raghavendra-Vempala, [AMS07]: Ambühl- Mastrolilli-Svensson, [CLTZ18]:Chan-Louis-Tang-Zhang, [LM16]:Louis-Makarychev.

Known Results for Sparse Vertex-Cuts

Vertex Expansion/Cuts less well-understood as compared to Edge Expansion/Sparsest Cut.

Algorithms:

- $(k=2)$ [FHL '08] : $O(\sqrt{\log n})$-approximation algorithm, using ℓ_1 line embeddings.
- $(k=2)$ [LRV '13]: $O(\sqrt{\log d/\mathrm{OPT}})$-approximation algorithm, where d is max-degree.
- $(k \geq 2)$ Can infer from [CLTZ '18, LM '16]: $O(\sqrt{\log n} \cdot OPT \cdot f(k))$.

Lower bounds $(k=2)$**:**

- [AMS '07]: No PTAS unless SAT has sub-exponential time algorithms.
- [LRV '13]: No constant-factor approximation algorithm, assuming Small-Set Expansion Hypothesis.

[FHL08]: Feige-Hajiaghayi-Lee, [LRV13]: Louis-Raghavendra-Vempala, [AMS07]: Ambühl- Mastrolilli-Svensson, [CLTZ18]:Chan-Louis-Tang-Zhang, [LM16]:Louis-Makarychev.

Known results for k-way Edge expansion

$$\Phi(G) = \min_{S_1,\ldots,S_k} \max_{i\in[k]} \frac{\left|E(S_i,\overline{S_i})\right|}{|S_i||\overline{S_i}|}$$

Better-studied

- Best known approximations are of the form: $O\left(\text{OPT}\sqrt{\log n}\cdot f_1(k)\right)$ or $O\left(\sqrt{\text{OPT}}\cdot f_2(k)\right)$.
  - [LM '14] $f_1(k) = \mathrm{poly}(k)$, to get exactly k-partition, if $|S_i|$'s are not known.
  - [BFK+ '11] Bi-criteria guarantee, with $f_1(k) = O(\sqrt{\log k})$, if the optimal S_i's are all of size n/k.
  - [LRTV '12, LGT '14] Spectral guarantees: $O(\sqrt{\lambda_k}\cdot\mathrm{poly}(k))$.

[LM14]: Louis-Makarychev, [BFK+11]:Bansal-Feige-Krauthgamer-Nagarajan-Naor-Schwartz, [LRTV12]:Louis-Raghavendra-Tetali-Vempala, [LGT14]:Lee-Gharan-Trevisan

The model k-Part (vertex)

Motivation: Keep well-connected within every part, only few vertices connect outside.

The model k-Part (vertex)

Motivation: Keep well-connected within every part, only few vertices connect outside.

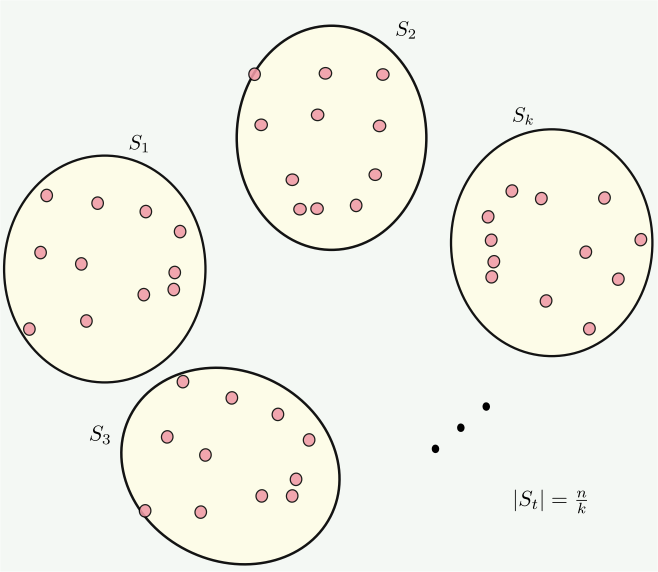
- Partition V into k sets $S_1, S_2, \ldots S_k$, with $|S_t| = n/k$ for every $t \in [k]$.
- Add edges within each each S_t to make it a spectral expander of degree (roughly) d and spectral gap $\geq \lambda$.

The model k-Part (vertex)

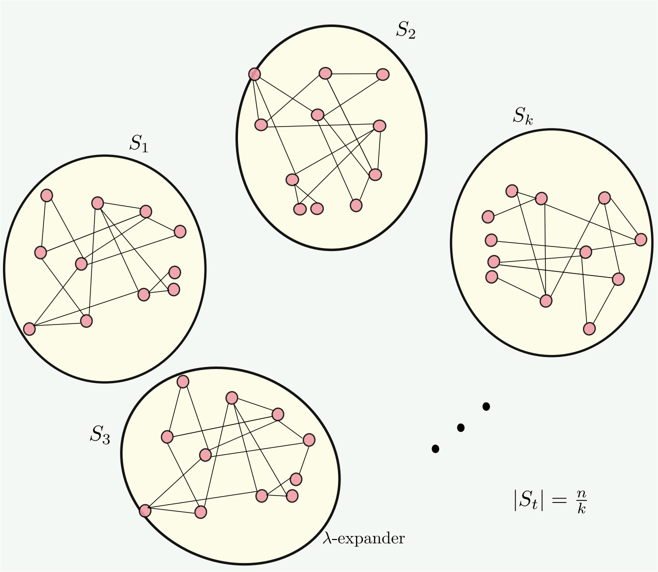
Motivation: Keep well-connected within every part, only few vertices connect outside.

- Partition V into k sets $S_1, S_2, ...S_k$, with $|S_t| = n/k$ for every $t \in [k]$.
- Add edges within each each S_t to make it a spectral expander of degree (roughly) d and spectral gap $\geq \lambda$.
- For each $t \in [k]$: Choose boundary vertices $T_t \subset S_t$ with $|T_t| \leq \varepsilon n/k$. Add arbitrary edges across T_t's
- Monotone adversary: Add edges arbitrarily within every S_t.

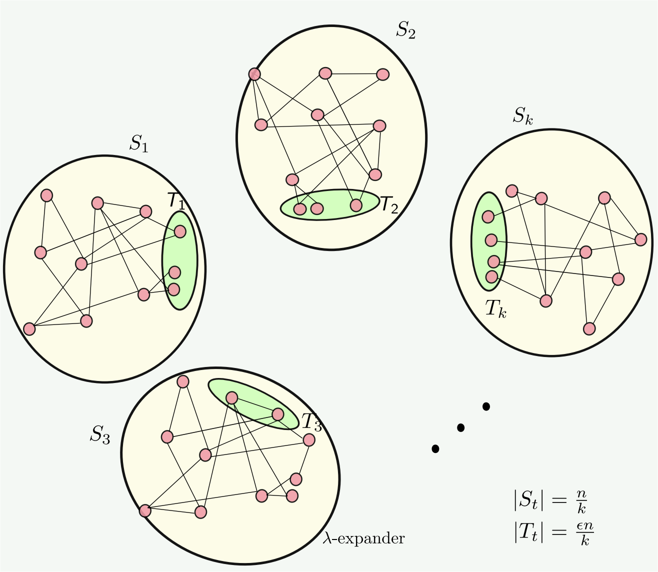
The model k-Part (vertex)



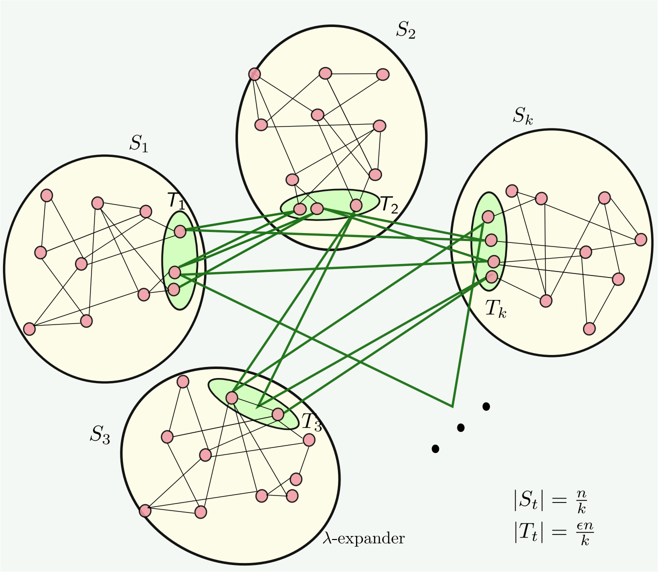
The model k-Part (vertex)



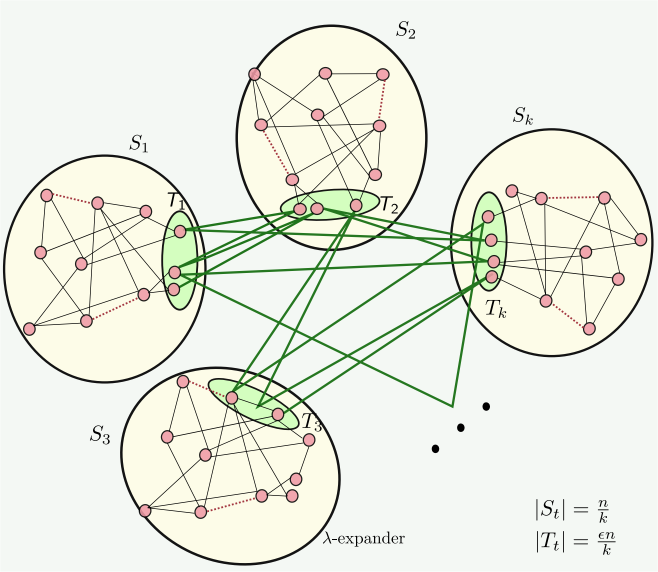
The model k-Part (vertex)



The model k-Part (vertex)



The model k-Part (vertex)



Main Result

Theorem (Louis-V.19)

For a graph from k-Part satisfying $\varepsilon \leq \lambda/800k$*, there is a polytime algorithm that outputs a k-partition* $\mathcal{P} = \{P_1, \ldots, P_k\}$ *of* V *such that:*

1. *For each* $i \in [k]$, $|P_i| \geq \Omega(n/k)$,
2. *For each* $i \in [k]$, $\Phi^V(P_i) \leq O(k^2)\mathrm{OPT}$

Main Result

Theorem (Louis-V.19)

For a graph from k-Part satisfying $\varepsilon \le \lambda/800k$, there is a polytime algorithm that outputs a k-partition $\mathcal{P} = \{P_1, \dots, P_k\}$ of V such that:

1. *For each $i \in [k]$, $|P_i| \ge \Omega(n/k)$,*
2. *For each $i \in [k]$, $\Phi^V(P_i) \le O(k^2)\mathrm{OPT}$*

- Above, OPT is the optimal balanced k-partition value.
  - Due to planted solution, $\mathrm{OPT} \le 2\varepsilon k$.
- Final approximation ratio is independent of n.
- Algorithm runs in time polynomial in both n, k.

Main Result

Theorem (Louis-V.19)

For a graph from k-Part satisfying $\varepsilon \leq \lambda/800k$, there is a polytime algorithm that outputs a k-partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of V such that:

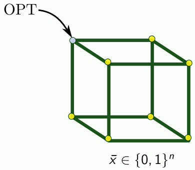
1. *For each $i \in [k]$, $|P_i| \geq \Omega(n/k)$,*
2. *For each $i \in [k]$, $\Phi^V(P_i) \leq O(k^2)\mathrm{OPT}$*

- Above, OPT is the optimal balanced k-partition value.
  - Due to planted solution, $\mathrm{OPT} \leq 2\varepsilon k$.
- Final approximation ratio is independent of n.
- Algorithm runs in time polynomial in both n, k.
- Similar guarantee holds for the edge expansion version.

General Framework

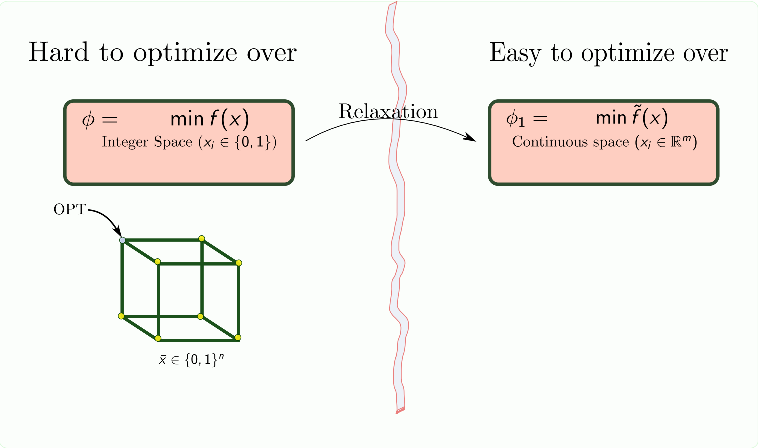
Hard to optimize over

$$\phi = \min_{\text{Integer Space } (x_i \in \{0,1\})} f(x)$$



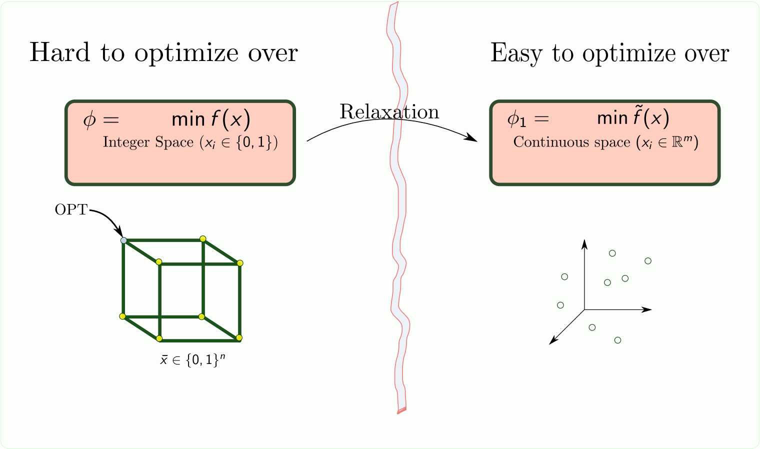
Easy to optimize over

General Framework



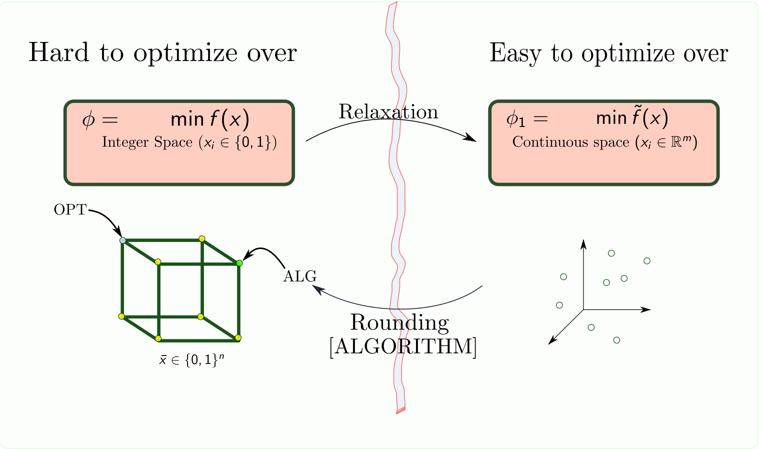
- The continuous space contains the discrete one, and therefore, $\phi_1 \leq \phi$.
- The relaxation step is generally well-understood.

General Framework



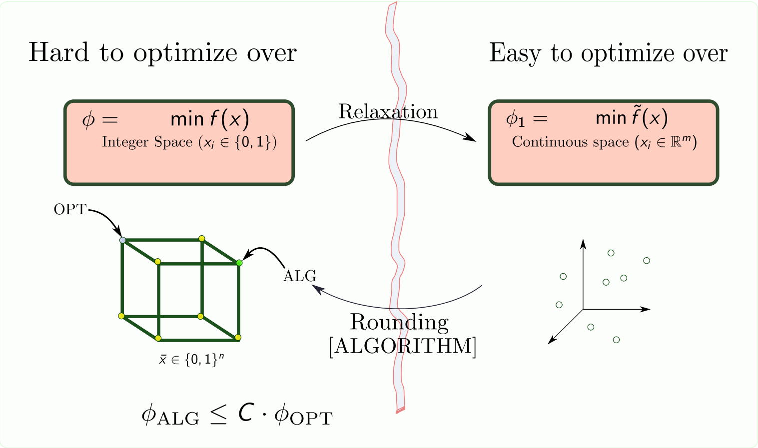
- The continuous space contains the discrete one, and therefore, $\phi_1 \leq \phi$.
- The relaxation step is generally well-understood.

General Framework



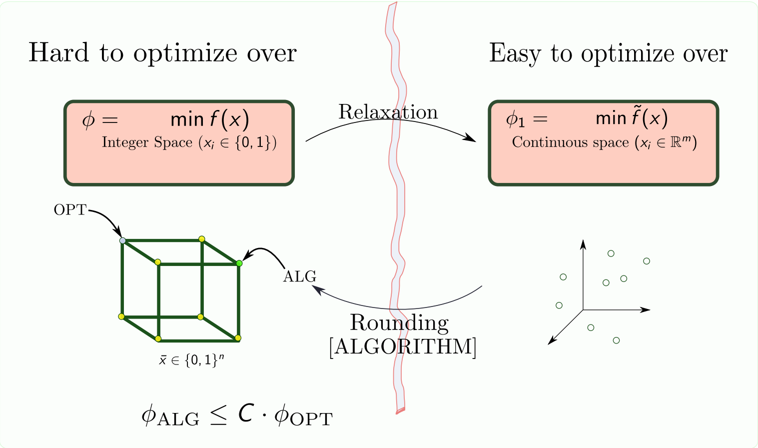
- The continuous space contains the discrete one, and therefore, $\phi_1 \leq \phi$.
- The relaxation step is generally well-understood.

General Framework



- The continuous space contains the discrete one, and therefore, $\phi_1 \leq \phi$.
- The relaxation step is generally well-understood.

General Framework



- Rounding step is usually the difficult part. Yields a solution with $\Phi_{\mathsf{ALG}} \leq C \cdot \phi_1 \leq C \cdot \Phi_{\mathsf{OPT}}$, for some $C \geq 1$.

Outline

Relaxation for 2-way vertex expansion

Relaxation for 2-way vertex expansion

$$\Phi_G^V = \min_S n \frac{|N(S) \cup N(\bar{S})|}{|S||\bar{S}|}$$

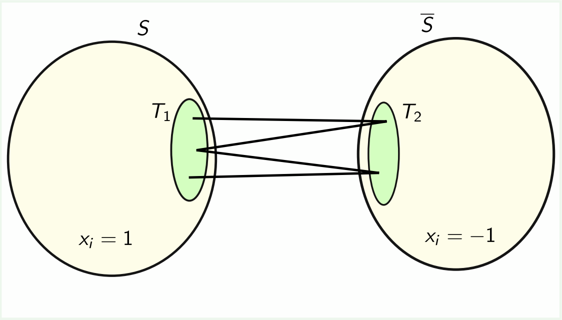
- Original Objective

Relaxation for 2-way vertex expansion

$$\Phi_G^V = \min_S n \frac{|N(S) \cup N(\bar{S})|}{|S||\bar{S}|}$$

$$\Phi_G^V = n \min_{x_i \in \{-1,1\}} \frac{\sum_i \max_{j \in N(i)} (x_i - x_j)^2}{\sum_{ij \in V \times V} (x_i - x_j)^2}$$

- Original Objective
- Where
  $x_i = 1$ if $i \in S$,
  $x_i = -1$ if $i \in \overline{S}$



SDP relaxation: 2-way vertex expansion

Relaxation: Assign a vector $u_i \in \mathbb{R}^d$ for every $i \in V$:

$$\Phi_{SDP}^{V} = \frac{1}{n} \cdot \min_{u_i \in \mathbb{R}^d} \sum_{i \in V} \max_{j \in N(i)} \|u_i - u_j\|^2$$

subject to:

$$\|u_i\|^2 = 1 \qquad \forall i \in V$$

$$\sum_{i \in V} \sum_{j \in V} \|u_i - u_j\|^2 = n^2$$

SDP relaxation: 2-way vertex expansion

Relaxation: Assign a vector $u_i \in \mathbb{R}^d$ for every $i \in V$:

$$\Phi^V_{SDP} = \frac{1}{n} \cdot \min_{u_i \in \mathbb{R}^d} \sum_{i \in V} \max_{j \in N(i)} \|u_i - u_j\|^2$$

subject to:

$$\|u_i\|^2 = 1 \qquad \forall i \in V$$

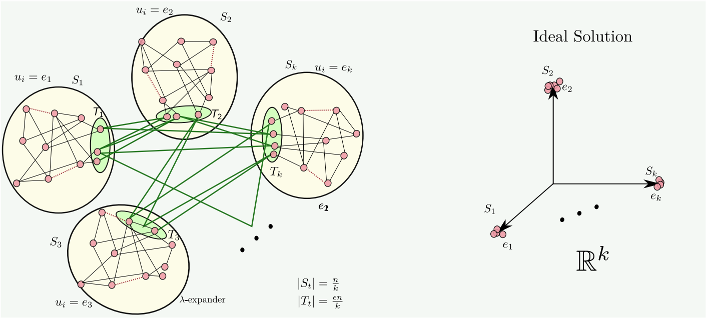
$$\sum_{i \in V} \sum_{j \in V} \|u_i - u_j\|^2 = n^2$$

- Ideal solution is $u_i \in \mathbb{R}$, with $u_i = 1$, if $i \in S_1$, and $u_i = -1$, if $i \in V \setminus S_1$
- This is indeed a relaxation, and therefore $\Phi^V_{SDP} \leq 4\varepsilon$ on k-part with $k = 2$.
- Note: An edge expansion objective would have the numerator as:

$$\sum_{i \in V} \sum_{j \in N(i)} \|u_i - u_j\|^2$$

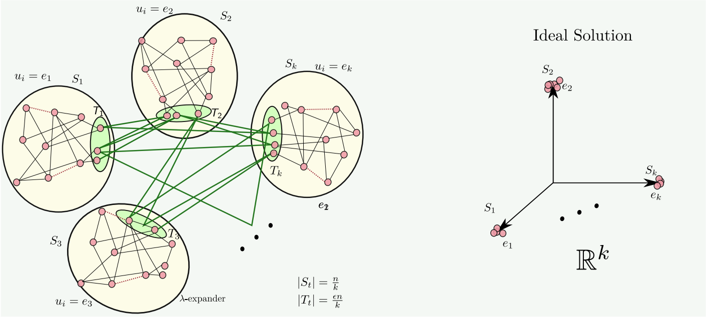
Relaxation for k-way expansion

- As before, assign one vector u_i for each $i \in V$.
- In the ideal solution, each vector is k-dimensional.
  - If $i \in S_t$, the intended solution is $u_i = e_t$, the unit vector along the t-th coordinate.



Relaxation for k-way expansion

- As before, assign one vector u_i for each $i \in V$.
- In the ideal solution, each vector is k-dimensional.
  - If $i \in S_t$, the intended solution is $u_i = e_t$, the unit vector along the t-th coordinate.
- The constraints are adjusted accordingly. We also add in additional ℓ_2^2 triangle inequality constraints.



SDP Relaxation for k-way vertex expansion

$$\Phi_{SDP}^{\mathrm{V,k}} := \min_{U} \sum_{i \in V} \eta_i$$

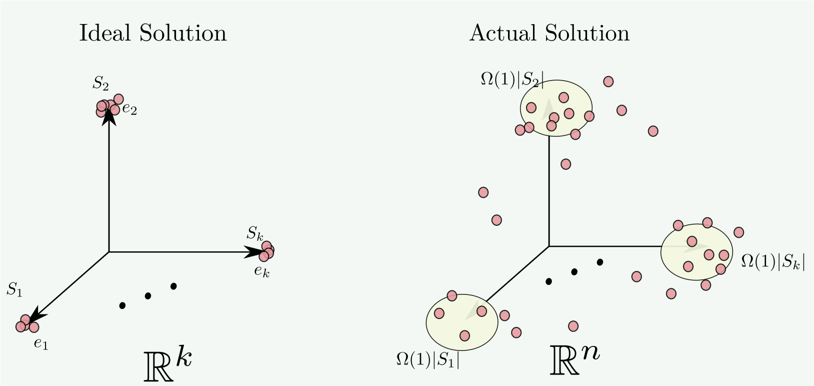
s.t.

$$\begin{aligned}
\eta_i &\geq \|u_i - u_j\|^2 && \forall i, \forall j \in N(i) \\
\|u_i\|^2 &= 1 && \forall i \in V \\
u_i^T u_j &\geq 0 && \forall i, j \in V \\
\sum_j u_i^T u_j &= n/k && \forall i \in V \\
\|u_i - u_j\|^2 + \|u_j - u_k\|^2 &\geq \|u_i - u_j\|^2 && \forall i, j, k \in V
\end{aligned}$$

$$\boxed{\Phi_{SDP}^{\mathrm{V,k}} \leq 2\varepsilon n}$$

Main Structure Lemma

The actual solution is "close" to the ideal solution for k-part instances



Main Structure Lemma

Lemma

Let $\{u_i\}_{i \in V}$ *be the optimal solution to the SDP for an instance* G *from* k*-Part*-vertex, *with* $\varepsilon \leq \lambda/800k$. *For each* $t \in [k]$, *let* $\mu_t = \mathbb{E}_{i \in S_t}[u_i]$. *The following holds:*

(a) $\forall t \in [k] : \quad \mathbb{E}_{j \in S_t}[\|\mu_t - u_j\|^2] \leq 1/800$

(b) $\forall t \in [k] : \quad 1 \geq \|\mu_t\|^2 \geq \Omega(1)$

(c) $\forall t \neq t' \quad \mu_t^T \mu_{t'} \leq 1/800$

Main Structure Lemma

Lemma

*Let $\{u_i\}_{i \in V}$ be the optimal solution to the SDP for an instance G from k-Part-*vertex*, with $\varepsilon \leq \lambda/800k$. For each $t \in [k]$, let $\mu_t = \mathbb{E}_{i \in S_t}[u_i]$. The following holds:*

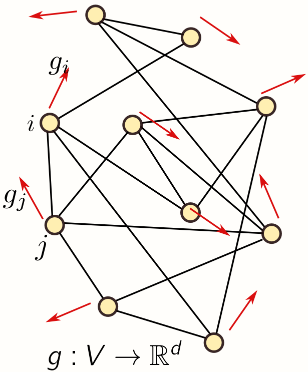
(a) $\forall t \in [k]: \quad \mathbb{E}_{j \in S_t}[\|\mu_t - u_j\|^2] \leq 1/800$

(b) $\forall t \in [k]: \quad 1 \geq \|\mu_t\|^2 \geq \Omega(1)$

(c) $\forall t \neq t' \quad \mu_t^T \mu_{t'} \leq 1/800$

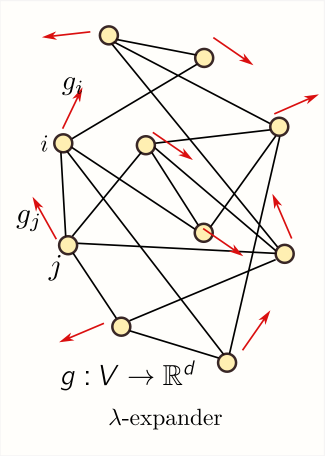
- Above, μ_t is the centroid of the vectors corresponding to S_t.
- The centroids are far apart, and almost orthonormal.
- Can greedily extract out k disjoint sets of size n/k using line embeddings.

Key: Local-Global Correlation on λ-expanders



λ-expander

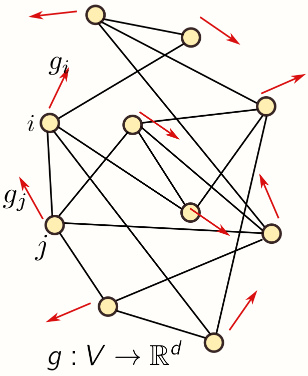
Key: Local-Global Correlation on λ-expanders



- Associate a vector $g : V \to \mathbb{R}^d$ with every vertex.
- **Expansion**: $\mathbb{E}_{e:\{i\sim j\}}[\|g_i - g_j\|^2] \ \leq \ \delta$

  $\implies$.

Key: Local-Global Correlation on λ-expanders



- Associate a vector $g : V \to \mathbb{R}^d$ with every vertex.
- **Expansion**: $\mathbb{E}_{e:\{i \sim j\}}[\|g_i - g_j\|^2] \;\leq\; \delta$
  $\implies \qquad \mathbb{E}_{ij}[\|g_i - g_j\|^2] \leq O(\delta/\lambda)$.
- λ is the second smallest eigenvalue of the Laplacian:

$$L_G \;=\; I \;-\; A/d$$

Here, d is the degree of the expander. .

Main lemma: Proof that S_t's are clustered

Ignore edges added by monotone adversary. The following (still) holds:

Main lemma: Proof that S_t's are clustered

Ignore edges added by monotone adversary. The following (still) holds:

Fix any $t \in [k]$. Since the SDP objective is $\sum_{i \in V} \eta_i \leq 2\varepsilon n$, we have:

$$\sum_{i \in S_t} \eta_i \leq 2\varepsilon n$$

$$\sum_{i \in S_t} \max_{j \in N(i)} \|u_i - u_j\|^2 \leq 2\varepsilon n$$

$$\implies \sum_{i \in S_t} \frac{1}{d} \sum_{j \in N(i) \cap S_t} \|u_i - u_j\|^2 \leq 2\varepsilon n \qquad \ldots \text{ since average} \leq \max$$

$$\implies \mathbb{E}_{\{i,j\} \in E(S_t)} \|u_i - u_j\|^2 \leq \varepsilon k$$

$$\implies \mathbb{E}_{i,j \in S_t} \|u_i - u_j\|^2 \leq \frac{\varepsilon k}{\lambda} \qquad \ldots \text{ using expansion within } S_t$$

Remaining steps in the proof

Following from the Main Lemma, we show:

- There are k disjoint, well-separated sets of vectors (corresponding to subsets of S_t's), each having small diameter and small vertex expansion.

Remaining steps in the proof

Following from the Main Lemma, we show:

- There are k disjoint, well-separated sets of vectors (corresponding to subsets of S_t's), each having small diameter and small vertex expansion.
- Given this structure, we can repeatedly (in a greedy fashion) find a $\Omega(n/k)$-sized set of small ($O(k \cdot \mathrm{OPT})$) vertex expansion using line embeddings.

Remaining steps in the proof

Following from the Main Lemma, we show:

- There are k disjoint, well-separated sets of vectors (corresponding to subsets of S_t's), each having small diameter and small vertex expansion.
- Given this structure, we can repeatedly (in a greedy fashion) find a $\Omega(n/k)$-sized set of small ($O(k \cdot \mathrm{OPT})$) vertex expansion using line embeddings.
- This does not give a true partition yet. However, we can move from k disjoint sets to a k -partition of vertices while incurring a further $O(k)$ approximation factor loss.

Thus, we get a $O(k^2)$-approximation.

Outline

Summary and Further Directions

- Going beyond worst-case analysis: semi-random and planted models, inspired from practical scenarios.
- An $O(k^2)$-approximate recovery result for vertex and edge-expansion.

Summary and Further Directions

- Going beyond worst-case analysis: semi-random and planted models, inspired from practical scenarios.
- An $O(k^2)$-approximate recovery result for vertex and edge-expansion.
- Immediate open questions from expansion objectives:
    - $O(\mathrm{poly} \log(k))$ guarantee? Relaxing expansion criterion?
- Many other problems too can be explored in this framework
    - Densest k-subgraph, Clustering variants, etc.
    - ML applications also provide a rich source of relevant questions
- Do higher order SDP or LP constraints help?
- Other settings such as Online or Streaming algorithms?

Thank You.
Questions?