Perturbation-Stable Maximum Cut

Yuchong Pan

UBC Beyond Worst-Case Analysis Reading Group
(Based on Tim Roughgarden's Notes for Stanford CS264)

June 30, 2020

Maximum Cut

Problem (Maximum Cut)

Input: An undirected graph $G = (V, E)$ with edge weights $w_e > 0$ for each $e \in E$.

Goal: A cut (A, B) that maximizes the weight of the **crossing** edges.

Maximum Cut

Problem (Maximum Cut)

Input: An undirected graph $G = (V, E)$ with edge weights $w_e > 0$ for each $e \in E$.

Goal: A cut (A, B) that maximizes the weight of the **crossing** edges.

- Maximum Cut is a type of 2-clustering problem (e.g. weights measure dissimilarities).

Maximum Cut Is *NP*-Hard

Problem (Maximum Cut, Decision Version)

Input: An undirected graph $G = (V, E)$ with edge weights $w_e > 0$ for each $e \in E$, and a positive integer W.

Output: Yes iff. there is a set $S \subseteq V$ such that the weight of the **crossing** edges is at least W.

Maximum Cut Is *NP*-Hard

Problem (Maximum Cut, Decision Version)

Input: An undirected graph $G = (V, E)$ with edge weights $w_e > 0$ for each $e \in E$, and a positive integer W.

Output: Yes iff. there is a set $S \subseteq V$ such that the weight of the **crossing** edges is at least W.

Problem (Partition, Decision Version)

Input: $(c_1, \ldots, c_n) \in \mathbb{Z}^n$.

Output: Yes iff. there is $I \subseteq [n]$ such that $\sum_{i \in I} c_i = \sum_{i \notin I} c_i$.

MAXIMUM CUT Is NP-Hard

Problem (MAXIMUM CUT, Decision Version)

Input: An undirected graph $G = (V, E)$ with edge weights $w_e > 0$ for each $e \in E$, and a positive integer W.

Output: Yes iff. there is a set $S \subseteq V$ such that the weight of the **crossing** edges is at least W.

Problem (PARTITION, Decision Version)

Input: $(c_1, \ldots, c_n) \in \mathbb{Z}^n$.

Output: Yes iff. there is $I \subseteq [n]$ such that $\sum_{i \in I} c_i = \sum_{i \notin I} c_i$.

Proof Sketch (PARTITION $\leq_P$ MAXIMUM CUT)

- $G = K_n$.
- $w_{ij} = c_i c_j$ for all $i, j \in V, i \neq j$.
- $W = \lceil \frac{1}{4} \sum c_i^2 \rceil$.

MAXIMUM CUT Is *NP*-Hard

- MINIMUM CUT is **not** *NP*-hard and can be solved by the Maximum-Flow Minimum-Cut Theorem.

Maximum Cut Is *NP*-Hard

- Minimum Cut is **not** *NP*-hard and can be solved by the Maximum-Flow Minimum-Cut Theorem.
- **Question:** Can't we negate the edge weights, yielding a Minimum Cut instance?

Maximum Cut Is *NP*-Hard

- Minimum Cut is **not** *NP*-hard and can be solved by the Maximum-Flow Minimum-Cut Theorem.
- **Question:** Can't we negate the edge weights, yielding a Minimum Cut instance?
- No! Polynomial-time algorithms solving Minimum Cut require nonnegative edge weights.

Exact Recovery

- **Theme:** To recover the optimal solution in polynomial time in γ-**perturbation-stable** instances, where γ is as small as possible.

Exact Recovery

- **Theme:** To recover the optimal solution in polynomial time in γ-**perturbation-stable** instances, where γ is as small as possible.

Definition (γ-Perturbation-Stability)

For $\gamma \geq 1$, an instance of $\textsc{Maximum Cut}$ is γ-**perturbation-stable** if a cut (A, B) is the **unique** optimal solution to all γ-**perturbations**, where each original edge weight w_e is replaced with an edge weight $w'_e \in [\frac{1}{\gamma} w_e, w_e]$.

LP Relaxation, Take 1

- **Question:** Can we use an LP relaxation similar to the one for Minimum Cut, i.e.

$$
\begin{aligned}
\max \quad & \sum_{e \in E} w_e x_e \\
\text{s.t.} \quad & x_e \geq |d_u - d_v|, && \forall e = uv \in E, \\
& x_e \in [0,1], && \forall e \in E, \\
& d_v \in [0,1], && \forall v \in V.
\end{aligned}
$$

LP Relaxation, Take 1

- **Question:** Can we use an LP relaxation similar to the one for MINIMUM CUT, i.e.

$$
\begin{aligned}
\max \quad & \sum_{e \in E} w_e x_e \\
\text{s.t.} \quad & x_e \geq |d_u - d_v|, && \forall e = uv \in E, \\
& x_e \in [0, 1], && \forall e \in E, \\
& d_v \in [0, 1], && \forall v \in V.
\end{aligned}
$$

- No! $x_e = 1$ for each $e \in E$ is a feasible solution and maximizes the objective value.

LP Relaxation, Take 1

- **Question:** Can we use an LP relaxation similar to the one for MINIMUM CUT, i.e.

$$
\begin{aligned}
\max \quad & \sum_{e \in E} w_e x_e & \\
\text{s.t.} \quad & x_e \geq |d_u - d_v|, & \forall e = uv \in E, \\
& x_e \in [0,1], & \forall e \in E, \\
& d_v \in [0,1], & \forall v \in V.
\end{aligned}
$$

- No! $x_e = 1$ for each $e \in E$ is a feasible solution and maximizes the objective value.
- **Question:** What about $x_e \leq d_u - d_v$ and $x_e \leq d_v - d_u$?

LP Relaxation, Take 1

- **Question:** Can we use an LP relaxation similar to the one for MINIMUM CUT, i.e.

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & x_e \geq |d_u - d_v|, && \forall e = uv \in E, \\ & x_e \in [0,1], && \forall e \in E, \\ & d_v \in [0,1], && \forall v \in V. \end{aligned}$$

- No! $x_e = 1$ for each $e \in E$ is a feasible solution and maximizes the objective value.
- **Question:** What about $x_e \leq d_u - d_v$ and $x_e \leq d_v - d_u$?
- This forces $x_e = 0$, instead of $x_e \leq |d_u - d_v|$.

LP Relaxation, Take 2

- Let $x_{ij} \in \{0, 1\}$ denote whether or not i, j are on different sides of the cut, for all distinct $i, j \in V$. We denote by x_{ij} and x_{ji} the same variable.

LP Relaxation, Take 2

- Let $x_{ij} \in \{0, 1\}$ denote whether or not i, j are on different sides of the cut, for all distinct $i, j \in V$. We denote by x_{ij} and x_{ji} the same variable.
- **Intuition:** If i, j are on different sides, and i, k are also on different sides, then j, k must be on the same side.

LP Relaxation, Take 2

- Let $x_{ij} \in \{0, 1\}$ denote whether or not i, j are on different sides of the cut, for all distinct $i, j \in V$. We denote by x_{ij} and x_{ji} the same variable.
- **Intuition:** If i, j are on different sides, and i, k are also on different sides, then j, k must be on the same side.
- For any distinct $i, j, k \in V$, at most two of x_{ij}, x_{ik}, x_{jk} are 1.

LP Relaxation, Take 2

- Let $x_{ij} \in \{0, 1\}$ denote whether or not i, j are on different sides of the cut, for all distinct $i, j \in V$. We denote by x_{ij} and x_{ji} the same variable.
- **Intuition:** If i, j are on different sides, and i, k are also on different sides, then j, k must be on the same side.
- For any distinct $i, j, k \in V$, at most two of x_{ij}, x_{ik}, x_{jk} are 1.

$$x_{ij} + x_{ik} + x_{jk} \leq 2, \qquad \forall i, j, k \in V \text{ distinct.}$$

LP Relaxation, Take 2

- Let $x_{ij} \in \{0, 1\}$ denote whether or not i, j are on different sides of the cut, for all distinct $i, j \in V$. We denote by x_{ij} and x_{ji} the same variable.
- **Intuition:** If i, j are on different sides, and i, k are also on different sides, then j, k must be on the same side.
- For any distinct $i, j, k \in V$, at most two of x_{ij}, x_{ik}, x_{jk} are 1.

$$x_{ij} + x_{ik} + x_{jk} \leq 2, \qquad \forall i, j, k \in V \text{ distinct.}$$

- **Intuition:** If i, j are on the same side, and i, k are on the same side, then j, k are on the same side.

LP Relaxation, Take 2

- Let $x_{ij} \in \{0,1\}$ denote whether or not i, j are on different sides of the cut, for all distinct $i, j \in V$. We denote by x_{ij} and x_{ji} the same variable.
- **Intuition:** If i, j are on different sides, and i, k are also on different sides, then j, k must be on the same side.
- For any distinct $i, j, k \in V$, at most two of x_{ij}, x_{ik}, x_{jk} are 1.

$$x_{ij} + x_{ik} + x_{jk} \leq 2, \qquad \forall i, j, k \in V \text{ distinct.}$$

- **Intuition:** If i, j are on the same side, and i, k are on the same side, then j, k are on the same side.
- For any distinct $i, j, k \in V$, $x_{ij} = x_{ik} = 0$ implies $x_{jk} = 0$.

LP Relaxation, Take 2

- Let $x_{ij} \in \{0, 1\}$ denote whether or not i, j are on different sides of the cut, for all distinct $i, j \in V$. We denote by x_{ij} and x_{ji} the same variable.
- **Intuition:** If i, j are on different sides, and i, k are also on different sides, then j, k must be on the same side.
- For any distinct $i, j, k \in V$, at most two of x_{ij}, x_{ik}, x_{jk} are 1.

$$x_{ij} + x_{ik} + x_{jk} \leq 2, \qquad \forall i, j, k \in V \text{ distinct.}$$

- **Intuition:** If i, j are on the same side, and i, k are on the same side, then j, k are on the same side.
- For any distinct $i, j, k \in V$, $x_{ij} = x_{ik} = 0$ implies $x_{jk} = 0$.

$$x_{jk} \leq x_{ij} + x_{ik}, \qquad \forall i, j, k \in V \text{ distinct.}$$

LP Relaxation, Take 2

- Hence we obtain the LP relaxation (LP-MaxCut):

$$\begin{aligned}
\max \quad & \sum_{(i,j)\in E} w_{ij} x_{ij} & \\
\text{s.t.} \quad & x_{ij} + x_{ik} + x_{jk} \le 2, & \forall i,j,k \in V \text{ distinct}, \\
& x_{jk} \le x_{ij} + x_{ik}, & \forall i,j,k \in V \text{ distinct}, \\
& x_{ij} \in [0,1], & \forall i,j \in V \text{ distinct}.
\end{aligned}$$

Main Theorem

Theorem

There is a constant $c > 0$ such that in every $(c \log n)$-perturbation-stable instance of $\textsc{Maximum Cut}$ *with n vertices,* $(\text{LP-}\textsc{MaxCut})$ *solves to integers.*

Main Theorem

Theorem

There is a constant $c > 0$ such that in every $(c \log n)$-perturbation-stable instance of Maximum Cut *with n vertices,* (LP-MaxCut) *solves to integers.*

- Recall the proofs of exact recovery by LP in 1-perturbation-stable Minimum s-t Cut instances and in 4-perturbation-stable Minimum Multiway Cut instances.

Main Theorem

Theorem

There is a constant $c > 0$ such that in every $(c \log n)$-perturbation-stable instance of MAXIMUM CUT *with n vertices,* (LP-MAXCUT) *solves to integers.*

- Recall the proofs of exact recovery by LP in 1-perturbation-stable MINIMUM s-t CUT instances and in 4-perturbation-stable MINIMUM MULTIWAY CUT instances.
- In each of the two proofs we design a **randomized rounding algorithm** that outputs a (random) cut such that the probability of an edge being cut is approximately the same as the value of the corresponding decision variable.

Main Theorem

Theorem

There is a constant $c > 0$ such that in every $(c \log n)$-perturbation-stable instance of MAXIMUM CUT *with n vertices,* (LP-MAXCUT) *solves to integers.*

- Recall the proofs of exact recovery by LP in 1-perturbation-stable MINIMUM s-t CUT instances and in 4-perturbation-stable MINIMUM MULTIWAY CUT instances.
- In each of the two proofs we design a **randomized rounding algorithm** that outputs a (random) cut such that the probability of an edge being cut is approximately the same as the value of the corresponding decision variable.
- MINIMUM s-t CUT: $A = \{v \in V : \hat{d}_v \leq r\}$ and $B = V \setminus A$, where $r \sim \text{Uniform}(0, 1)$.

Main Theorem

Theorem

There is a constant $c > 0$ such that in every $(c \log n)$-perturbation-stable instance of Maximum Cut *with n vertices,* (LP-MaxCut) *solves to integers.*

- Recall the proofs of exact recovery by LP in 1-perturbation-stable Minimum s-t Cut instances and in 4-perturbation-stable Minimum Multiway Cut instances.
- In each of the two proofs we design a **randomized rounding algorithm** that outputs a (random) cut such that the probability of an edge being cut is approximately the same as the value of the corresponding decision variable.
- Minimum s-t Cut: $A = \{v \in V : \hat{d}_v \leq r\}$ and $B = V \setminus A$, where $r \sim \mathrm{Uniform}(0, 1)$.
- Minimum Multiway Cut: For each iteration, a group and a threshold are chosen uniformly randomly.

Main Theorem

- A randomized rounding algorithm implies the exact recovery theorem since:

Main Theorem

- A randomized rounding algorithm implies the exact recovery theorem since:
  1. The optimal fractional solution $\hat{x}$ can only be better than the optimal integral solution C^*;

Main Theorem

- A randomized rounding algorithm implies the exact recovery theorem since:
  1. The optimal fractional solution $\hat{x}$ can only be better than the optimal integral solution C^*;
  2. The randomized rounding algorithm gives a distribution over s-t cuts that is as good, on average, as C^*;

Main Theorem

- A randomized rounding algorithm implies the exact recovery theorem since:
  1. The optimal fractional solution $\hat{x}$ can only be better than the optimal integral solution C^*;
  2. The randomized rounding algorithm gives a distribution over s-t cuts that is as good, on average, as C^*;
  3. Hence the distribution must be a point mass on C^*.

Main Theorem

- A randomized rounding algorithm implies the exact recovery theorem since:
  1. The optimal fractional solution $\hat{x}$ can only be better than the optimal integral solution C^*;
  2. The randomized rounding algorithm gives a distribution over s-t cuts that is as good, on average, as C^*;
  3. Hence the distribution must be a point mass on C^*.
- Formally, we define $\Delta(C)$ to be the total cost of C that exceeds that of C^* and $\Delta(\hat{x})$ to be total cost of C^* that exceeds the objective function value of $\hat{x}$.

Main Theorem

- A randomized rounding algorithm implies the exact recovery theorem since:
  1. The optimal fractional solution $\hat{\mathrm{x}}$ can only be better than the optimal integral solution C^*;
  2. The randomized rounding algorithm gives a distribution over s-t cuts that is as good, on average, as C^*;
  3. Hence the distribution must be a point mass on C^*.
- Formally, we define $\Delta(C)$ to be the total cost of C that exceeds that of C^* and $\Delta(\hat{\mathrm{x}})$ to be total cost of C^* that exceeds the objective function value of $\hat{\mathrm{x}}$.
- We show that $\mathbb{E}[\Delta(C)] \leq 0$ by the probability properties of the cut generated by the randomized rounding algorithm.

Main Theorem

- A randomized rounding algorithm implies the exact recovery theorem since:
  1. The optimal fractional solution $\hat{x}$ can only be better than the optimal integral solution C^*;
  2. The randomized rounding algorithm gives a distribution over s-t cuts that is as good, on average, as C^*;
  3. Hence the distribution must be a point mass on C^*.
- Formally, we define $\Delta(C)$ to be the total cost of C that exceeds that of C^* and $\Delta(\hat{x})$ to be total cost of C^* that exceeds the objective function value of $\hat{x}$.
- We show that $\mathbb{E}[\Delta(C)] \leq 0$ by the probability properties of the cut generated by the randomized rounding algorithm.
- Since $\Delta(C) \geq 0$ and since the equality holds iff. C is an optimal cut, it follows that the randomized rounding algorithm outputs an optimal cut w.p.1.

Randomized Rounding Algorithm

Lemma

Fix an instance of the $\textsc{Maximum Cut}$ *problem, with* F^* *the edges in the optimal cut, and* $\hat{x}$ *the optimal solution to* $(\text{LP-}\textsc{MaxCut})$. *Then there exists a randomized algorithm that generates a random cut* (A, B) *and a scaling parameter* $\sigma > 0$ *such that:*

1. *For every edge* $e = ij \notin F^*$,

$$\mathbb{P}[e \textit{ cut by } (A, B)] \geq \sigma \cdot \frac{\hat{x}_{ij}}{\alpha},$$

   where $\alpha = \Theta(\log n)$;

2. *For every edge* $e = ij \in F^*$,

$$\mathbb{P}[e \textit{ not cut by } (A, B)] \leq \sigma \cdot (1 - \hat{x}_{ij});$$

3. *The rounding algorithm is deterministic iff.* $\hat{x}$ *is integral.*

Randomized Rounding Algorithm, Roadmap

- **Exercise:** Show that this lemma implies the main theorem (outlined above, Homework #4).

Randomized Rounding Algorithm, Roadmap

- **Exercise:** Show that this lemma implies the main theorem (outlined above, Homework #4).

Proposition

Fix an instance of Maximum Cut, a cut C, and a feasible solution $\hat{x}$ to (LP-MaxCut). For distinct $i, j \in V$, define

$$\hat{y}_{ij} = \begin{cases} \hat{x}_{ij}, & \text{if } i, j \text{ are on the same side of } C, \\ 1 - \hat{x}_{ij}, & \text{if } i, j \text{ are on different sides of } C. \end{cases}$$

Then $\hat{y}$ satisfies the triangle inequality:

$$\hat{y}_{jk} \le \hat{y}_{ij} + \hat{y}_{ik}$$

for every $i, j, k \in V$.

Randomized Rounding Algorithm, Roadmap

- **Exercise:** Show that this lemma implies the main theorem (outlined above, Homework #4).

Proposition

Fix an instance of Maximum Cut, a cut C, and a feasible solution $\hat{x}$ to (LP-MaxCut). For distinct $i, j \in V$, define

$$\hat{y}_{ij} = \begin{cases} \hat{x}_{ij}, & \text{if } i,j \text{ are on the same side of } C, \\ 1 - \hat{x}_{ij}, & \text{if } i,j \text{ are on different sides of } C. \end{cases}$$

Then $\hat{y}$ satisfies the triangle inequality:

$$\hat{y}_{jk} \leq \hat{y}_{ij} + \hat{y}_{ik}$$

for every $i, j, k \in V$.

- That is, $\hat{x}, \hat{y}$ are both **pseudometrics** (i.e. metrics except that distinct points may have zero distances).

Randomized Rounding Algorithm, Roadmap

Theorem (Bourgain's Theorem)

For every n-point **pseudometric** *space* (X, d), *there exists a randomized algorithm that generates a random partition* (A, B) *of* X *and a scaling parameter* $\sigma > 0$ *such that, for all distinct* $i, j \in X$,

$$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{d(i,j)}{\alpha}, d(i,j)\right],$$

where $\alpha = \Theta(\log n)$.

Randomized Rounding Algorithm, Roadmap

Theorem (Bourgain's Theorem)

For every n-point **pseudometric** *space (X, d), there exists a randomized algorithm that generates a random partition (A, B) of X and a scaling parameter $\sigma > 0$ such that, for all distinct $i, j \in X$,*

$$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{d(i,j)}{\alpha}, d(i,j)\right],$$

where $\alpha = \Theta(\log n)$.

- That is, every n-point metric space admits a randomized partitioning algorithm so that the sepration probabilities between pairs of points are **proportional** to the distances, up to a $\Theta(\log n)$ factor.

Randomized Rounding Algorithm, Roadmap

Theorem (Bourgain's Theorem)

For every n-point **pseudometric** *space (X, d), there exists a randomized algorithm that generates a random partition (A, B) of X and a scaling parameter $\sigma > 0$ such that, for all distinct $i, j \in X$,*

$$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{d(i,j)}{\alpha}, d(i,j)\right],$$

where $\alpha = \Theta(\log n)$.

- That is, every n-point metric space admits a randomized partitioning algorithm so that the sepration probabilities between pairs of points are **proportional** to the distances, up to a $\Theta(\log n)$ factor.
- The $\Theta(\log n)$ approximation factor is the best possible for **arbitrary** pseudometric spaces.

Randomized Rounding Algorithm, Roadmap

Proof (Proposition & Bourgain's Theorem $\implies$ Lemma).

- Fix an instance of Maximum Cut. Let C^* denote an optimal cut, cutting the edges F^*.

Randomized Rounding Algorithm, Roadmap

Proof (Proposition & Bourgain's Theorem $\implies$ Lemma).

- Fix an instance of $\textsc{Maximum Cut}$. Let C^* denote an optimal cut, cutting the edges F^*.
- Let $\hat{x}$ be an optimal solution to $(\text{LP-}\textsc{MaxCut})$. Define $\hat{y}$ as in Proposition (with C^* being the cut).

Randomized Rounding Algorithm, Roadmap

Proof (Proposition & Bourgain's Theorem $\implies$ Lemma).

- Fix an instance of $\textsc{Maximum Cut}$. Let C^* denote an optimal cut, cutting the edges F^*.
- Let $\hat{x}$ be an optimal solution to $(\text{LP-}\textsc{MaxCut})$. Define $\hat{y}$ as in Proposition (with C^* being the cut).
- By Proposition, $\hat{y}$ is a pseudometric.

Randomized Rounding Algorithm, Roadmap

Proof (Proposition & Bourgain's Theorem $\implies$ Lemma).

- Fix an instance of MAXIMUM CUT. Let C^* denote an optimal cut, cutting the edges F^*.
- Let $\hat{x}$ be an optimal solution to (LP-MAXCUT). Define $\hat{y}$ as in Proposition (with C^* being the cut).
- By Proposition, $\hat{y}$ is a pseudometric.
- By Bourgain's Theorem, there is a randomized algorithm that outputs a partition (A, B) and $\sigma > 0$ such that

$$\mathbb{P}[i, j \text{ on different sides of } (A, B)] = \sigma \cdot \left[\frac{\hat{y}_{ij}}{\alpha}, \hat{y}_{ij}\right],$$

where $\alpha = \Theta(\log n)$.

Randomized Rounding Algorithm, Roadmap

Proof (Proposition & Bourgain's Theorem $\implies$ Lemma).

- By the definition of $\hat{y}$,
  1. If i, j are on the same side of C^*, then
  $$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{\hat{x}_{ij}}{\alpha}, \hat{x}_{ij}\right].$$
  2. If i, j are on different sides of C^*, then
  $$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{1 - \hat{x}_{ij}}{\alpha}, 1 - \hat{x}_{ij}\right].$$

Randomized Rounding Algorithm, Roadmap

Proof (Proposition & Bourgain's Theorem $\implies$ Lemma).

- By the definition of $\hat{y}$,
  1. If i, j are on the same side of C^*, then
  $$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{\hat{x}_{ij}}{\alpha}, \hat{x}_{ij}\right].$$
  2. If i, j are on different sides of C^*, then
  $$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{1 - \hat{x}_{ij}}{\alpha}, 1 - \hat{x}_{ij}\right].$$
- (3) follows from the description of the randomized algorithm in Bourgain's Theorem.

□

Randomized Rounding Algorithm, Roadmap

Proof (Proposition & Bourgain's Theorem $\implies$ Lemma).

- By the definition of $\hat{y}$,
  1. If i, j are on the same side of C^*, then
  $$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{\hat{x}_{ij}}{\alpha}, \hat{x}_{ij}\right].$$
  2. If i, j are on different sides of C^*, then
  $$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{1 - \hat{x}_{ij}}{\alpha}, 1 - \hat{x}_{ij}\right].$$
- (3) follows from the description of the randomized algorithm in Bourgain's Theorem.

$\square$

- **Exercise:** Prove Proposition and Bourgain's Theorem (Homework #4).

Dimensionality Reduction

Definition (α-Embeddings)

Let $(X, d_X), (Y, d_Y)$ be metric spaces. We say that $\phi : X \to Y$ is an α-**embedding** if there exists $r > 0$ such that

$$r \cdot d_X(u, v) \leq d_Y(\phi(u), \phi(v)) \leq r \cdot \alpha \cdot d_X(u, v).$$

for all $u, v \in X$.

Dimensionality Reduction

Definition (α-Embeddings)

Let $(X, d_X), (Y, d_Y)$ be metric spaces. We say that $\phi : X \to Y$ is an α-**embedding** if there exists $r > 0$ such that

$$r \cdot d_X(u, v) \leq d_Y(\phi(u), \phi(v)) \leq r \cdot \alpha \cdot d_X(u, v).$$

for all $u, v \in X$.

- **Dimensionality reduction** is the process of mapping a high dimensional dataset to a lower dimensional space, while preserving much of the important structure.

Dimensionality Reduction

Definition (α-Embeddings)

Let $(X, d_X), (Y, d_Y)$ be metric spaces. We say that $\phi : X \to Y$ is an α-**embedding** if there exists $r > 0$ such that

$$r \cdot d_X(u, v) \leq d_Y(\phi(u), \phi(v)) \leq r \cdot \alpha \cdot d_X(u, v).$$

for all $u, v \in X$.

- **Dimensionality reduction** is the process of mapping a high dimensional dataset to a lower dimensional space, while preserving much of the important structure.
- For instance, let $X \subseteq \mathbb{R}^d$ and $Y = \mathbb{R}^t$ with $t < d$ and d_X, d_Y being the Euclidean distance.

Dimensionality Reduction

Theorem (Johnson-Lindenstrauss, 1984)

Let $x_1, \ldots, x_n \in \mathbb{R}^d$*. Let* $\epsilon \in (0,1)$*. Then for some* $t = O(\frac{\log(n)}{\epsilon^2})$*, there exist* $y_1, \ldots, y_n \in \mathbb{R}^t$ *such that*

$$\begin{array}{rcccll} (1-\epsilon)\left\|x_j\right\| & \leq & \left\|y_j\right\| & \leq & (1+\epsilon)\left\|x_j\right\|, & \forall j \in [n], \\ (1-\epsilon)\left\|x_j - x_{j'}\right\| & \leq & \left\|y_j - y_{j'}\right\| & \leq & (1+\epsilon)\left\|x_j - x_{j'}\right\|, & \forall j, j' \in [n]. \end{array}$$

Notation: $\|v\| = \sqrt{\sum_{i=1}^n v_i^2}$.

Dimensionality Reduction

Theorem (Johnson-Lindenstrauss, 1984)

Let $x_1, \ldots, x_n \in \mathbb{R}^d$. *Let* $\epsilon \in (0,1)$. *Then for some* $t = O(\frac{\log(n)}{\epsilon^2})$, *there exist* $y_1, \ldots, y_n \in \mathbb{R}^t$ *such that*

$$\begin{array}{rcccll} (1-\epsilon)\left\|x_j\right\| & \leq & \left\|y_j\right\| & \leq & (1+\epsilon)\left\|x_j\right\|, & \forall j \in [n], \\ (1-\epsilon)\left\|x_j - x_{j'}\right\| & \leq & \left\|y_j - y_{j'}\right\| & \leq & (1+\epsilon)\left\|x_j - x_{j'}\right\|, & \forall j, j' \in [n]. \end{array}$$

Notation: $\|v\| = \sqrt{\sum_{i=1}^n v_i^2}$.

- **Remark:** There is a **random linear map** such that for any $x_1, \ldots, x_n$ the above condition holds with probability at least $\frac{1}{2n}$. This linear map is **oblivious**: it does not depend on $x_1, \ldots, x_n$ at all! In fact, the linear map is just a matrix whose entries are independent Gaussians.

Bourgain's Theorem & Sparsest Cut

Theorem (Bourgain's Metric Embedding Theorem)

For any metric space (V, d), there exists an $O(\log n)$-embedding into $\mathbb{R}^{O(\log^2 n)}$ with the ℓ_1-norm that is computable with high probability by a randomized polynomial-time algorithm.

Bourgain's Theorem & SPARSEST CUT

Theorem (Bourgain's Metric Embedding Theorem)

For any metric space (V, d), there exists an $O(\log n)$-embedding into $\mathbb{R}^{O(\log^2 n)}$ with the ℓ_1-norm that is computable with high probability by a randomized polynomial-time algorithm.

- This result is the best possible; i.e., there exists a metric that cannot be embedded into ℓ_1 with distortion less than $\Omega(\log n)$.

Bourgain's Theorem & Sparsest Cut

Definition (Cut Metrics)

A metric (X, d) is a **cut metric** if there exists $S \subseteq X$ such that $d(x, y) = 0$ whenever $x, y \in S$ or $x, y \notin S$, and $d(x, y) = 1$ otherwise.

Bourgain's Theorem & Sparsest Cut

Definition (Cut Metrics)

A metric (X, d) is a **cut metric** if there exists $S \subseteq X$ such that $d(x, y) = 0$ whenever $x, y \in S$ or $x, y \notin S$, and $d(x, y) = 1$ otherwise.

Lemma

A metric (X, d) is an ℓ_1 metric if and only if it is a nonnegative linear combination of cut metrics.

Bourgain's Theorem & Sparsest Cut

Definition (Cut Metrics)

A metric (X, d) is a **cut metric** if there exists $S \subseteq X$ such that $d(x, y) = 0$ whenever $x, y \in S$ or $x, y \notin S$, and $d(x, y) = 1$ otherwise.

Lemma

A metric (X, d) is an ℓ_1 metric if and only if it is a nonnegative linear combination of cut metrics.

- **Exercise:** Prove Lemma.

Bourgain's Theorem & Sparsest Cut

Definition (Cut Metrics)

A metric (X, d) is a **cut metric** if there exists $S \subseteq X$ such that $d(x, y) = 0$ whenever $x, y \in S$ or $x, y \notin S$, and $d(x, y) = 1$ otherwise.

Lemma

A metric (X, d) is an ℓ_1 metric if and only if it is a nonnegative linear combination of cut metrics.

- **Exercise:** Prove Lemma.
- Lemma and Bourgain's Metric Embedding Theorem imply Bourgain's Theorem in the proof of the main theorem.

Bourgain's Theorem & Sparsest Cut

Problem (Sparsest Cut)

Input: An undirected graph $G = (V, E)$ with edge weights $w_e > 0$ for each $e \in E$, and k pairs of vertices (s_i, t_i) each with demand d_i.

Goal: A set of vertices S that minimizes

$$\rho(S) \equiv \frac{\sum_{e \in \delta(S)} c_e}{\sum_{i : |S \cap \{s_i, t_i\}| = 1} d_i}.$$

Bourgain's Theorem & Sparsest Cut

Problem (Sparsest Cut)

Input: An undirected graph $G = (V, E)$ with edge weights $w_e > 0$ for each $e \in E$, and k pairs of vertices (s_i, t_i) each with demand d_i.

Goal: A set of vertices S that minimizes

$$\rho(S) \equiv \frac{\sum_{e \in \delta(S)} c_e}{\sum_{i:|S \cap \{s_i, t_i\}|=1} d_i}.$$

Corollary

There is a randomized $O(\log n)$-approximation algorithm for Sparsest Cut.

Tree Metric Embedding

Definition (Tree Metrics)

A metric (X, d) is a **tree metric** if there exists a tree $T = (V, E)$ with edge costs c_e for each $e \in E$ such that $d(u, v)$ is the cost of the unique path from u to v in T.

Tree Metric Embedding

Definition (Tree Metrics)

A metric (X, d) is a **tree metric** if there exists a tree $T = (V, E)$ with edge costs c_e for each $e \in E$ such that $d(u, v)$ is the cost of the unique path from u to v in T.

Theorem (Fakcharoenphol-Rao-Talwar)

For any metric (V, d) such that $d(u, v) \geq 1$ for all $u, v \in V$ with $u \neq v$, there exists a randomized, polynomial-time algorithm that produces a tree metric (V', T), $V \subseteq V'$ such that for all $u, v \in V$, we have $d(u, v) \leq T_{uv}$ and $\mathbb{E}[T_{uv}] \leq O(\log n) d(u, v)$.

Tree Metric Embedding

Definition (Tree Metrics)

A metric (X, d) is a **tree metric** if there exists a tree $T = (V, E)$ with edge costs c_e for each $e \in E$ such that $d(u, v)$ is the cost of the unique path from u to v in T.

Theorem (Fakcharoenphol-Rao-Talwar)

For any metric (V, d) such that $d(u, v) \geq 1$ for all $u, v \in V$ with $u \neq v$, there exists a randomized, polynomial-time algorithm that produces a tree metric $(V', T), V \subseteq V'$ such that for all $u, v \in V$, we have $d(u, v) \leq T_{uv}$ and $\mathbb{E}[T_{uv}] \leq O(\log n) d(u, v)$.

- The above result is obtained via the method of **hierarchical tree decomposition**.

Semidefinite Programming

Definition (Positive Semidefinite Matrices)

A matrix $X \in \mathbb{R}^{n\times n}$ is **positive semidefinite** (or **psd**), denoted $X \succeq 0$, if $y^T X y \geq 0$ for all $y \in \mathbb{R}^n$.

Fact

If $X \in \mathbb{R}^{n\times n}$ is a symmetric matrix, then the following statements are equivalent:

1. X *is psd;*
2. X *has nonnegative eigenvalues;*
3. $X = V^T V$ *for some* $V \in \mathbb{R}^{m\times n}$ *where* $m \leq n$*;*
4. $X = \sum_{i=1}^n \lambda_i w_i w_i^T$ *for some* $\lambda_i \geq 0$ *and* $w_i \in \mathbb{R}^n$ *such that* $w_i^T w_i = 1$ *and* $w_i^T w_j = 0$ *for all* $i \neq j$*.*

Semidefinite Programming

Definition (Semidefinite Programming, SDP)

A **semidefinite program**, or **SDP**, is a mathematical program with real-valued variables, a linear objective function, linear constraints, and a square symmetric matrix of variables constrained to be psd.

Semidefinite Programming

Definition (Semidefinite Programming, SDP)

A **semidefinite program**, or **SDP**, is a mathematical program with real-valued variables, a linear objective function, linear constraints, and a square symmetric matrix of variables constrained to be psd.

- Below is an example of SDP with variables x_{ij} for $i, j \in [n]$:

$$
\begin{aligned}
\text{max or min} \quad & \sum_{i,j\in[n]} c_{ij} x_{ij} && (1) \\
\text{s.t.} \quad & \sum_{i,j\in[n]} a_{ijk} x_{ij} = b_k, && \forall k \in [n], \\
& x_{ij} = x_{ji}, && \forall i, j \in [n], \\
& X = (x_{ij}) \succeq 0.
\end{aligned}
$$

Semidefinite Programming

Definition (Semidefinite Programming, SDP)

A **semidefinite program**, or **SDP**, is a mathematical program with real-valued variables, a linear objective function, linear constraints, and a square symmetric matrix of variables constrained to be psd.

- Below is an example of SDP with variables x_{ij} for $i, j \in [n]$:

$$\begin{aligned} \text{max or min} \quad & \sum_{i,j \in [n]} c_{ij} x_{ij} && \qquad (1) \\ \text{s.t.} \quad & \sum_{i,j \in [n]} a_{ijk} x_{ij} = b_k, && \forall k \in [n], \\ & x_{ij} = x_{ji}, && \forall i, j \in [n], \\ & X = (x_{ij}) \succeq 0. \end{aligned}$$

- SDP can be solved to within an additive error of ϵ in polynomial time in the size of the input and $\log(\frac{1}{\epsilon})$.

Equivalence of SDP and Vector Programming

Definition (Vector Programming)

A **vector program** is a mathematical program with variables $v_i \in \mathbb{R}^n$, where n is the number of vectors, and an objective function and constraints linear in the inner products of the vectors.

Equivalence of SDP and Vector Programming

Definition (Vector Programming)

A **vector program** is a mathematical program with variables $\mathrm{v}_i \in \mathbb{R}^n$, where n is the number of vectors, and an objective function and constraints linear in the inner products of the vectors.

- Below is an example of vector programming with variables $\mathrm{v}_i \in \mathbb{R}^n$ for $i \in [n]$:

$$
\begin{aligned}
\text{max or min} \quad & \sum_{i,j \in [n]} c_{ij} \left(v_i \cdot v_j \right) && && (2) \\
\text{s.t.} \quad & \sum_{i,j \in [n]} a_{ijk} \left(v_i \cdot v_j \right) = b_k, && \forall k \in [n], \\
& \mathrm{v}_i \in \mathbb{R}^n, && \forall i \in [n].
\end{aligned}
$$

Equivalence of SDP and Vector Programming

Theorem (Equivalence of SDP and Vector Programming)

The SDP (1) *and the vector program* (2) *are equivalent.*

Equivalence of SDP and Vector Programming

Theorem (Equivalence of SDP and Vector Programming)

The SDP (1) *and the vector program* (2) *are equivalent.*

Proof.

- ($\Longrightarrow$) Given a solution X to (1), compute a matrix V such that $\mathsf{X} = \mathsf{V}^T\mathsf{V}$ (within small error), and set v_i to be the i^{th} column of V.

Equivalence of SDP and Vector Programming

Theorem (Equivalence of SDP and Vector Programming)

The SDP (1) *and the vector program* (2) *are equivalent.*

Proof.

- ($\Longrightarrow$) Given a solution X to (1), compute a matrix V such that $\mathrm{X} = \mathrm{V}^T\mathrm{V}$ (within small error), and set v_i to be the i^{th} column of V.
- Then $x_{ij} = \mathrm{v}_i^T \mathrm{v}_j = \mathrm{v}_i \cdot \mathrm{v}_j$.

Equivalence of SDP and Vector Programming

Theorem (Equivalence of SDP and Vector Programming)

The SDP (1) *and the vector program* (2) *are equivalent.*

Proof.

- ($\Longrightarrow$) Given a solution X to (1), compute a matrix V such that $\mathrm{X} = \mathrm{V}^T\mathrm{V}$ (within small error), and set v_i to be the i^{th} column of V.
- Then $x_{ij} = \mathrm{v}_i^T\mathrm{v}_j = \mathrm{v}_i \cdot \mathrm{v}_j$.
- Hence v_i's is a feasible solution to (2) with the same value.

Equivalence of SDP and Vector Programming

Theorem (Equivalence of SDP and Vector Programming)

The SDP (1) *and the vector program* (2) *are equivalent.*

Proof.

- ($\Longrightarrow$) Given a solution X to (1), compute a matrix V such that $\mathrm{X} = \mathrm{V}^T\mathrm{V}$ (within small error), and set v_i to be the i^{th} column of V.
- Then $x_{ij} = \mathrm{v}_i^T \mathrm{v}_j = \mathrm{v}_i \cdot \mathrm{v}_j$.
- Hence v_i's is a feasible solution to (2) with the same value.
- ($\Longleftarrow$) Given a solution v_i's to (2), construct a matrix V with i^{th} column v_i, and let $\mathrm{X} = \mathrm{V}^T\mathrm{V}$.

Equivalence of SDP and Vector Programming

Theorem (Equivalence of SDP and Vector Programming)

The SDP (1) *and the vector program* (2) *are equivalent.*

Proof.

- ($\Longrightarrow$) Given a solution X to (1), compute a matrix V such that $\mathrm{X} = \mathrm{V}^T\mathrm{V}$ (within small error), and set v_i to be the i^{th} column of V.
- Then $x_{ij} = \mathrm{v}_i^T\mathrm{v}_j = \mathrm{v}_i \cdot \mathrm{v}_j$.
- Hence v_i's is a feasible solution to (2) with the same value.
- ($\Longleftarrow$) Given a solution v_i's to (2), construct a matrix V with i^{th} column v_i, and let $\mathrm{X} = \mathrm{V}^T\mathrm{V}$.
- Then X is symmetric and psd by Fact, with $x_{ij} = \mathrm{v}_i \cdot \mathrm{v}_j$.

Equivalence of SDP and Vector Programming

Theorem (Equivalence of SDP and Vector Programming)

The SDP (1) *and the vector program* (2) *are equivalent.*

Proof.

- ($\Longrightarrow$) Given a solution X to (1), compute a matrix V such that $\mathrm{X} = \mathrm{V}^T\mathrm{V}$ (within small error), and set v_i to be the i^{th} column of V.
- Then $x_{ij} = \mathrm{v}_i^T\mathrm{v}_j = \mathrm{v}_i \cdot \mathrm{v}_j$.
- Hence v_i's is a feasible solution to (2) with the same value.
- ($\Longleftarrow$) Given a solution v_i's to (2), construct a matrix V with i^{th} column v_i, and let $\mathrm{X} = \mathrm{V}^T\mathrm{V}$.
- Then X is symmetric and psd by Fact, with $x_{ij} = \mathrm{v}_i \cdot \mathrm{v}_j$.
- Hence X is a feasible solution to (1) with the same value.

$\square$

Quadratic Programming Formulation

- **Recall:** We obtained exact recovery in $\Theta(\log n)$-perturbation-stable instances of MAXIMUM CUT.

Quadratic Programming Formulation

- **Recall:** We obtained exact recovery in $\Theta(\log n)$-perturbation-stable instances of Maximum Cut.
- **Goal:** To get exact recovery for smaller values of γ by SDP.

Quadratic Programming Formulation

- **Recall:** We obtained exact recovery in $\Theta(\log n)$-perturbation-stable instances of MAXIMUM CUT.
- **Goal:** To get exact recovery for smaller values of γ by SDP.
- We start with a quadratic programming formulation, which suggests a vector programming relaxation since a vector program contains **inner products** of the vectors.

Quadratic Programming Formulation

- **Recall:** We obtained exact recovery in $\Theta(\log n)$-perturbation-stable instances of Maximum Cut.
- **Goal:** To get exact recovery for smaller values of γ by SDP.
- We start with a quadratic programming formulation, which suggests a vector programming relaxation since a vector program contains **inner products** of the vectors.
- Let $z_i \in \{-1, +1\}$ for each $i \in V$ indicate which side of the cut i belongs to.

Quadratic Programming Formulation

- **Recall:** We obtained exact recovery in $\Theta(\log n)$-perturbation-stable instances of MAXIMUM CUT.
- **Goal:** To get exact recovery for smaller values of γ by SDP.
- We start with a quadratic programming formulation, which suggests a vector programming relaxation since a vector program contains **inner products** of the vectors.
- Let $z_i \in \{-1, +1\}$ for each $i \in V$ indicate which side of the cut i belongs to.
- Hence, $z_i - z_j$ is 0 if i, j are on the same side of the cut, and ± 2 otherwise.

Quadratic Programming Formulation

- **Recall:** We obtained exact recovery in $\Theta(\log n)$-perturbation-stable instances of $\textsc{Maximum Cut}$.
- **Goal:** To get exact recovery for smaller values of γ by SDP.
- We start with a quadratic programming formulation, which suggests a vector programming relaxation since a vector program contains **inner products** of the vectors.
- Let $z_i \in \{-1, +1\}$ for each $i \in V$ indicate which side of the cut i belongs to.
- Hence, $z_i - z_j$ is 0 if i, j are on the same side of the cut, and ± 2 otherwise.
- Equivalently, $(z_i - z_j)^2$ is 0 if i, j are on the same side of the cut, and 4 otherwise.

Quadratic Programming Formulation

- Hence we obtain the **exact** quadratic programming formulation of Maximum Cut:

$$\begin{aligned} \max \quad & \frac{1}{4}\sum_{ij\in E} w_{ij}(z_i - z_j)^2 \\ \text{s.t.} \quad & z_i \in \{-1,+1\}, \qquad \forall i \in V. \end{aligned}$$

Quadratic Programming Formulation

- Hence we obtain the **exact** quadratic programming formulation of MAXIMUM CUT:

$$\begin{array}{lll} \max & \frac{1}{4}\sum_{ij\in E} w_{ij}(z_i - z_j)^2 & \\ \text{s.t.} & z_i \in \{-1,+1\}, & \forall i \in V. \end{array}$$

- This quadratic program is **equivalent** to MAXIMUM CUT. Hence optimizing this program is *NP*-hard.

Vector Programming Relaxation

- **Recall:** Vector programming and SDP are equivalent and hence can be solved to within small error in polynomial time.

Vector Programming Relaxation

- **Recall:** Vector programming and SDP are equivalent and hence can be solved to within small error in polynomial time.
- Hence relaxing each $z_i \in \{-1, 1\}$ to a **unit vector** $\mathrm{v}_i \in \mathbb{R}^n$ and therefore replacing quadratic terms with inner products yield a vector program that is computationally tractable:

$$
\begin{aligned}
\max \quad & \frac{1}{4} \sum_{ij \in E} w_{ij} \left\| \mathrm{v}_i - \mathrm{v}_j \right\|^2 \\
\text{s.t.} \quad & \left\| \mathrm{v}_i \right\|^2 = 1, \qquad \forall i \in V, \\
& \mathrm{v}_i \in \mathbb{R}^n, \qquad \forall i \in V.
\end{aligned}
$$

Vector Programming Relaxation

- **Recall:** Vector programming and SDP are equivalent and hence can be solved to within small error in polynomial time.
- Hence relaxing each $z_i \in \{-1, 1\}$ to a **unit vector** $\mathrm{v}_i \in \mathbb{R}^n$ and therefore replacing quadratic terms with inner products yield a vector program that is computationally tractable:

$$
\begin{array}{lrl}
\max & \frac{1}{4} \sum_{ij \in E} w_{ij} \|\mathrm{v}_i - \mathrm{v}_j\|^2 & \\
 & \|\mathrm{v}_i\|^2 = 1, & \forall i \in V, \\
\text{s.t.} & \mathrm{v}_i \in \mathbb{R}^n, & \forall i \in V.
\end{array}
$$

- The quadratic terms $\|\mathrm{v}_i - \mathrm{v}_j\|^2$ and $\|\mathrm{v}_i\|^2$ expands to sums of inner products $\mathrm{v}_i \cdot \mathrm{v}_i - 2 \cdot \mathrm{v}_i \cdot \mathrm{v}_j + \mathrm{v}_j \cdot \mathrm{v}_j$ and $\mathrm{v}_i \cdot \mathrm{v}_i$, respectively.

Vector Programming Relaxation

- **Recall:** Vector programming and SDP are equivalent and hence can be solved to within small error in polynomial time.
- Hence relaxing each $z_i \in \{-1, 1\}$ to a **unit vector** $\mathrm{v}_i \in \mathbb{R}^n$ and therefore replacing quadratic terms with inner products yield a vector program that is computationally tractable:

$$
\begin{aligned}
\max \quad & \frac{1}{4} \sum_{ij \in E} w_{ij} \|\mathrm{v}_i - \mathrm{v}_j\|^2 \\
\text{s.t.} \quad & \|\mathrm{v}_i\|^2 = 1, \qquad \forall i \in V, \\
& \mathrm{v}_i \in \mathbb{R}^n, \qquad \forall i \in V.
\end{aligned}
$$

- The quadratic terms $\|\mathrm{v}_i - \mathrm{v}_j\|^2$ and $\|\mathrm{v}_i\|^2$ expands to sums of inner products $\mathrm{v}_i \cdot \mathrm{v}_i - 2 \cdot \mathrm{v}_i \cdot \mathrm{v}_j + \mathrm{v}_j \cdot \mathrm{v}_j$ and $\mathrm{v}_i \cdot \mathrm{v}_i$, respectively.
- This vector program is a relaxation of the quadratic program by setting $\mathrm{v}_i = (z_i, 0, \ldots, 0) \in \mathbb{R}^n$.

Goemans-Williamson Approximation Algorithm

- The above vector programming relaxation already leads to the best-known approximation algorithm for MAXIMUM CUT:

Goemans-Williamson Approximation Algorithm

- The above vector programming relaxation already leads to the best-known approximation algorithm for MAXIMUM CUT:

Theorem (Goemans-Williamson Approximation Algorithm)

There is a randomized .878-approximation algorithm for MAXIMUM CUT.

Goemans-Williamson Approximation Algorithm

- The above vector programming relaxation already leads to the best-known approximation algorithm for MAXIMUM CUT:

Theorem (Goemans-Williamson Approximation Algorithm)

There is a randomized .878-approximation algorithm for MAXIMUM CUT*.*

Theorem

Given the unique game conjecture there is no α-approximation for MAXIMUM CUT *with constant $\alpha > .878$ unless $P = NP$.*

Vector Programming Relaxation

- For our purposes we want the vector programming relaxation to generalize the LP relaxation. Hence we want the analogs of the following two sets of constraints:

$$
\begin{aligned}
x_{ij} + x_{ik} + x_{jk} &\leq 2, && \forall i, j, k \in V \text{ distinct}, \\
x_{jk} &\leq x_{ij} + x_{ik}, && \forall i, j, k \in V \text{ distinct}.
\end{aligned}
$$

Vector Programming Relaxation

- For our purposes we want the vector programming relaxation to generalize the LP relaxation. Hence we want the analogs of the following two sets of constraints:

$$
\begin{aligned}
x_{ij} + x_{ik} + x_{jk} &\leq 2, \qquad && \forall i, j, k \in V \text{ distinct},\\
x_{jk} &\leq x_{ij} + x_{ik}, \qquad && \forall i, j, k \in V \text{ distinct}.
\end{aligned}
$$

- This implies the following two sets of constraints in the form of inner products:

$$
\begin{aligned}
&\|\mathrm{v}_i - \mathrm{v}_j\|^2 + \|\mathrm{v}_i - \mathrm{v}_k\|^2 + \|\mathrm{v}_j - \mathrm{v}_k\|^2 \leq 8, && \forall i, j, k \in V \text{ distinct},\\
&\|\mathrm{v}_j - \mathrm{v}_k\|^2 \leq \|\mathrm{v}_i - \mathrm{v}_j\|^2 + \|\mathrm{v}_i - \mathrm{v}_k\|^2, && \forall i, j, k \in V \text{ distinct}.
\end{aligned}
$$

Vector Programming Relaxation

- For our purposes we want the vector programming relaxation to generalize the LP relaxation. Hence we want the analogs of the following two sets of constraints:

$$\begin{aligned} x_{ij} + x_{ik} + x_{jk} &\le 2, && \forall i, j, k \in V \text{ distinct}, \\ x_{jk} &\le x_{ij} + x_{ik}, && \forall i, j, k \in V \text{ distinct}. \end{aligned}$$

- This implies the following two sets of constraints in the form of inner products:

$$\begin{aligned} &\|\mathrm{v}_i - \mathrm{v}_j\|^2 + \|\mathrm{v}_i - \mathrm{v}_k\|^2 + \|\mathrm{v}_j - \mathrm{v}_k\|^2 \le 8, && \forall i, j, k \in V \text{ distinct}, \\ &\|\mathrm{v}_j - \mathrm{v}_k\|^2 \le \|\mathrm{v}_i - \mathrm{v}_j\|^2 + \|\mathrm{v}_i - \mathrm{v}_k\|^2, && \forall i, j, k \in V \text{ distinct}. \end{aligned}$$

- The extended vector program, which we call (VP-MaxCut), is still a relaxation for Maximum Cut by setting v_i to $\pm e_1$ according to i's side.

Improved Exact Recovery, Roadmap

- Recall the roadmap for the proof of the exact recovery for MAXIMUM CUT with $\gamma = \Theta(\log n)$.

Improved Exact Recovery, Roadmap

- Recall the roadmap for the proof of the exact recovery for $\textsc{Maximum Cut}$ with $\gamma = \Theta(\log n)$.
- We first proved a proposition saying that if $\hat{x}$ is a pseudometric, then $\hat{y}$ defined below is also a pseudometric:

$$\hat{y}_{ij} = \begin{cases} \hat{x}_{ij}, & \text{if } i, j \text{ are on the same side of } C, \\ 1 - \hat{x}_{ij}, & \text{if } i, j \text{ are on different sides of } C. \end{cases}$$

Improved Exact Recovery, Roadmap

- Recall the roadmap for the proof of the exact recovery for $\textsc{Maximum Cut}$ with $\gamma = \Theta(\log n)$.
- We first proved a proposition saying that if $\hat{x}$ is a pseudometric, then $\hat{y}$ defined below is also a pseudometric:

$$\hat{y}_{ij} = \begin{cases} \hat{x}_{ij}, & \text{if } i, j \text{ are on the same side of } C, \\ 1 - \hat{x}_{ij}, & \text{if } i, j \text{ are on different sides of } C. \end{cases}$$

- This proposition and Bourgain's Theorem imply the lemma for the existence of randomized rounding algorithm that outputs a (random) cut such that the probability of an edge being cut is approximately the same as the value of the corresponding decision variable.

Improved Exact Recovery, Roadmap

- Recall the roadmap for the proof of the exact recovery for MAXIMUM CUT with $\gamma = \Theta(\log n)$.
- We first proved a proposition saying that if $\hat{x}$ is a pseudometric, then $\hat{y}$ defined below is also a pseudometric:

$$\hat{y}_{ij} = \begin{cases} \hat{x}_{ij}, & \text{if } i,j \text{ are on the same side of } C, \\ 1 - \hat{x}_{ij}, & \text{if } i,j \text{ are on different sides of } C. \end{cases}$$

- This proposition and Bourgain's Theorem imply the lemma for the existence of randomized rounding algorithm that outputs a (random) cut such that the probability of an edge being cut is approximately the same as the value of the corresponding decision variable.
- The lemma then implies the exact recovery theorem by a common pattern used in perturbation-stable MINIMUM CUT and MINIMUM MULTIWAY CUT.

Improved Exact Recovery, Roadmap

- Fix an instance of MAXIMUM CUT. Let $\hat{\mathrm{v}}_i$'s be an optimal solution to the vector program. Let $\hat{x}_{ij} = \frac{1}{4}\|\mathrm{v}_i - \mathrm{v}_j\|^2$ for any distinct $i, j \in V$.

Improved Exact Recovery, Roadmap

- Fix an instance of MAXIMUM CUT. Let $\hat{\mathrm{v}}_i$'s be an optimal solution to the vector program. Let $\hat{x}_{ij} = \frac{1}{4}\|\mathrm{v}_i - \mathrm{v}_j\|^2$ for any distinct $i, j \in V$.
- $\hat{\mathrm{x}}$ is a metric by the triangle inequality constraints of (VP-MAXCUT).

Improved Exact Recovery, Roadmap

- Fix an instance of $\textsc{Maximum Cut}$. Let $\hat{\mathsf{v}}_i$'s be an optimal solution to the vector program. Let $\hat{x}_{ij} = \frac{1}{4}\|\mathsf{v}_i - \mathsf{v}_j\|^2$ for any distinct $i, j \in V$.
- $\hat{\mathsf{x}}$ is a metric by the triangle inequality constraints of $(\text{VP-}\textsc{MaxCut})$.
- Furthermore, $\hat{x}_{ij}$ represents the squared Euclidean distances between points in $\mathbb{R}^k$ for some k.

Improved Exact Recovery, Roadmap

- Fix an instance of MAXIMUM CUT. Let $\hat{\mathrm{v}}_i$'s be an optimal solution to the vector program. Let $\hat{x}_{ij} = \frac{1}{4}\|\mathrm{v}_i - \mathrm{v}_j\|^2$ for any distinct $i, j \in V$.
- $\hat{\mathrm{x}}$ is a metric by the triangle inequality constraints of (VP-MAXCUT).
- Furthermore, $\hat{x}_{ij}$ represents the squared Euclidean distances between points in $\mathbb{R}^k$ for some k.

Definition (ℓ_2^2 Metrics)

A metric (X, d) is an ℓ_2^2 **metric** if it represents squared Euclidean distances between points in $\mathbb{R}^k$ for some k.
That is, there exists an embedding from X to $\mathbb{R}^k$ for some k such that $d(\cdot,\cdot)$'s are squared Euclidean distances between points in $\mathbb{R}^k$.

Improved Exact Recovery, Roadmap

- Not every metric is an ℓ_2^2 metric, e.g. the discrete metric.

Improved Exact Recovery, Roadmap

- Not every metric is an ℓ_2^2 metric, e.g. the discrete metric.
- The $\Theta(\log n)$ approximation in Bourgain's Theorem is the best possible for **arbitrary** pseudometric spaces.

Improved Exact Recovery, Roadmap

- Not every metric is an ℓ_2^2 metric, e.g. the discrete metric.
- The $\Theta(\log n)$ approximation in Bourgain's Theorem is the best possible for **arbitrary** pseudometric spaces.
- **Question:** Can the approximation factor α be improved for the more restricted ℓ_2^2 metrics?

Improved Exact Recovery, Roadmap

- Not every metric is an ℓ_2^2 metric, e.g. the discrete metric.
- The $\Theta(\log n)$ approximation in Bourgain's Theorem is the best possible for **arbitrary** pseudometric spaces.
- **Question:** Can the approximation factor α be improved for the more restricted ℓ_2^2 metrics?

Theorem (Arora, Lee, and Naor, 2005)

For every n-point ℓ_2^2 metric space (X, d), there exists a randomized algorithm that generates a random partition (A, B) of X and a scaling parameter $\sigma > 0$ such that, for all distinct $i, j \in X$,

$$\mathbb{P}[i, j \text{ on different sides of } (A, B)] \in \sigma \cdot \left[\frac{d(i,j)}{\alpha}, d(i,j)\right],$$

where $\alpha = O(\sqrt{\log n} \log \log n)$.

Improved Exact Recovery, Roadmap

- Arora, Lee, and Naor's Theorem gives the lower bound for the probability of an edge being cut. Similarly, we need an upper bound for the probability of an edge not being cut.

Improved Exact Recovery, Roadmap

- Arora, Lee, and Naor's Theorem gives the lower bound for the probability of an edge being cut. Similarly, we need an upper bound for the probability of an edge not being cut.
- Hence we prove an analog of the proposition used in the exact recovery with $\gamma = \Theta(\log n)$. That is, if $\hat{x}$ is an ℓ_2^2 metric, then so is the similarly defined $\hat{y}$.

Improved Exact Recovery, Roadmap

- Arora, Lee, and Naor's Theorem gives the lower bound for the probability of an edge being cut. Similarly, we need an upper bound for the probability of an edge not being cut.
- Hence we prove an analog of the proposition used in the exact recovery with $\gamma = \Theta(\log n)$. That is, if $\hat{\mathrm{x}}$ is an ℓ_2^2 metric, then so is the similarly defined $\hat{\mathrm{y}}$.

Proposition

Fix an instance of $\textsc{Maximum Cut}$, a cut C, and a feasible solution $\hat{\mathrm{v}}_i$'s to $(\text{VP-}\textsc{MaxCut})$. Let $\hat{\mathrm{x}}$ be the induced ℓ_2^2 metric $\hat{x}_{ij} = \frac{1}{4}\|\mathrm{v}_i - \mathrm{v}_j\|^2$. For distinct $i, j \in V$, define

$$\hat{y}_{ij} = \begin{cases} \hat{x}_{ij}, & \text{if } i, j \text{ are on the same side of } C, \\ 1 - \hat{x}_{ij}, & \text{if } i, j \text{ are on different sides of } C. \end{cases}$$

Then $\hat{\mathrm{y}}$ is also an ℓ_2^2 metric.

Improved Exact Recovery, Roadmap

Proof.

- $\hat{y}$ is a metric by the triangle inequality constraints of $(\text{VP-}\textsc{MaxCut})$.

Improved Exact Recovery, Roadmap

Proof.

- $\hat{y}$ is a metric by the triangle inequality constraints of (VP-MaxCut).
- To prove that $\hat{y}$ is an ℓ_2^2 metric, we need to exhibit an embedding of V into $\mathbb{R}^k$ for some k such that $\hat{y}_{ij}$'s are squared Euclidean distances between points in $\mathbb{R}^k$.

Improved Exact Recovery, Roadmap

Proof.

- $\hat{y}$ is a metric by the triangle inequality constraints of (VP-MaxCut).
- To prove that $\hat{y}$ is an ℓ_2^2 metric, we need to exhibit an embedding of V into $\mathbb{R}^k$ for some k such that $\hat{y}_{ij}$'s are squared Euclidean distances between points in $\mathbb{R}^k$.
- Define

$$\hat{u}_i = \begin{cases} \hat{v}_i, & \text{if } i \in A, \\ -\hat{v}_i, & \text{if } i \in B. \end{cases}$$

Improved Exact Recovery, Roadmap

Proof.

- $\hat{y}$ is a metric by the triangle inequality constraints of (VP-MaxCut).
- To prove that $\hat{y}$ is an ℓ_2^2 metric, we need to exhibit an embedding of V into $\mathbb{R}^k$ for some k such that $\hat{y}_{ij}$'s are squared Euclidean distances between points in $\mathbb{R}^k$.
- Define

$$\hat{u}_i = \begin{cases} \hat{v}_i, & \text{if } i \in A, \\ -\hat{v}_i, & \text{if } i \in B. \end{cases}$$

- Then $\hat{y}_{ij} = \frac{1}{4}\|\hat{u}_i - \hat{u}_j\|^2$ for any distinct $i, j \in V$.

$\square$

Improved Exact Recovery, Roadmap

Lemma

Fix an instance of $\textsc{Maximum Cut}$, *with* F^* *the edges in the optimal cut, and* v_i*'s the optimal solution to* $(\textsc{VP-MaxCut})$*. Then there exists a randomized algorithm that generates a random cut* (A, B) *and a scaling parameter* $\sigma > 0$ *such that:*

1. *For every edge* $e = ij \notin F^*$,

$$\mathbb{P}[e \text{ cut by } (A, B)] \geq \sigma \cdot \frac{\frac{1}{4}\|\mathsf{v}_i - \mathsf{v}_j\|^2}{\alpha},$$

   where $\alpha = \Theta(\sqrt{\log n}\log\log n)$*;*

2. *For every edge* $e = ij \in F^*$,

$$\mathbb{P}[e \text{ not cut by } (A, B)] \leq \sigma \cdot \left(1 - \frac{1}{4}\|\mathsf{v}_i - \mathsf{v}_j\|\right);$$

3. *The rounding algorithm is deterministic iff.* $\hat{\mathsf{v}}_i$*'s are integral.*

Improved Exact Recovery, Roadmap

- By $\hat{\mathsf{v}}_\mathsf{i}$'s being **integral**, we mean that there exist antipodal unit vectors $\mathsf{w}, -\mathsf{w}$ such that $\hat{\mathsf{v}}_\mathsf{i} \in \{\mathsf{w}, -\mathsf{w}\}$ for each $i \in V$.

Improved Exact Recovery, Roadmap

- By $\hat{\mathrm{v}}_{\mathrm{i}}$'s being **integral**, we mean that there exist antipodal unit vectors $\mathrm{w}, -\mathrm{w}$ such that $\hat{\mathrm{v}}_{\mathrm{i}} \in \{\mathrm{w}, -\mathrm{w}\}$ for each $i \in V$.
- As the lemma implies the main theorem in the exact recovery with $\gamma = \Theta(\log n)$, the above lemma implies the following theorem by an analagous argument:

Improved Exact Recovery, Roadmap

- By $\hat{\mathrm{v}}_\mathrm{i}$'s being **integral**, we mean that there exist antipodal unit vectors $\mathrm{w}, -\mathrm{w}$ such that $\hat{\mathrm{v}}_\mathrm{i} \in \{\mathrm{w}, -\mathrm{w}\}$ for each $i \in V$.
- As the lemma implies the main theorem in the exact recovery with $\gamma = \Theta(\log n)$, the above lemma implies the following theorem by an analagous argument:

Theorem

There is a constant $c > 0$ such that in every $(c\sqrt{\log n}\log\log n)$-perturbation-stable instance of MAXIMUM CUT *with n vertices, every optimal solution to* (VP-MAXCUT) *is integral.*

Can We Do Better?

Theorem

There exist $O(\sqrt{\log n})$-perturbation-stable instances of $\textsc{Maximum Cut}$ *for which the optimal solution to* $(\text{VP-}\textsc{MaxCut})$ *is not integral.*

Can We Do Better?

Theorem

There exist $O(\sqrt{\log n})$-perturbation-stable instances of Maximum Cut *for which the optimal solution to* (VP-MaxCut) *is not integral.*

Theorem

Assuming the Unique Game Conjecture, for every constant $\gamma \geq 1$, there is no polynomial-time algorithm for certifiable exact recovery in γ-perturbation-stable instances of Maximum Cut.

Can We Do Better?

Theorem

There exist $O(\sqrt{\log n})$-perturbation-stable instances of MAXIMUM CUT *for which the optimal solution to* (VP-MAXCUT) *is not integral.*

Theorem

Assuming the Unique Game Conjecture, for every constant $\gamma \geq 1$, there is no polynomial-time algorithm for certifiable exact recovery in γ-perturbation-stable instances of MAXIMUM CUT.

- Both results are based upon a reduction from SPARSEST CUT to MAXIMUM CUT. See *Bilu–Linial Stable Instances of Max Cut and Minimum Multiway* by Makarychev, Makarychev, and Vijayaraghavan (2013) and Homework #4.