Maximizing the Correlation: Extending Grothendieck's Inequality to Large Domains

MSc Thesis Seminar, December 2020

Dor Katzelnick

Advisor: Dr. Roy Schwartz

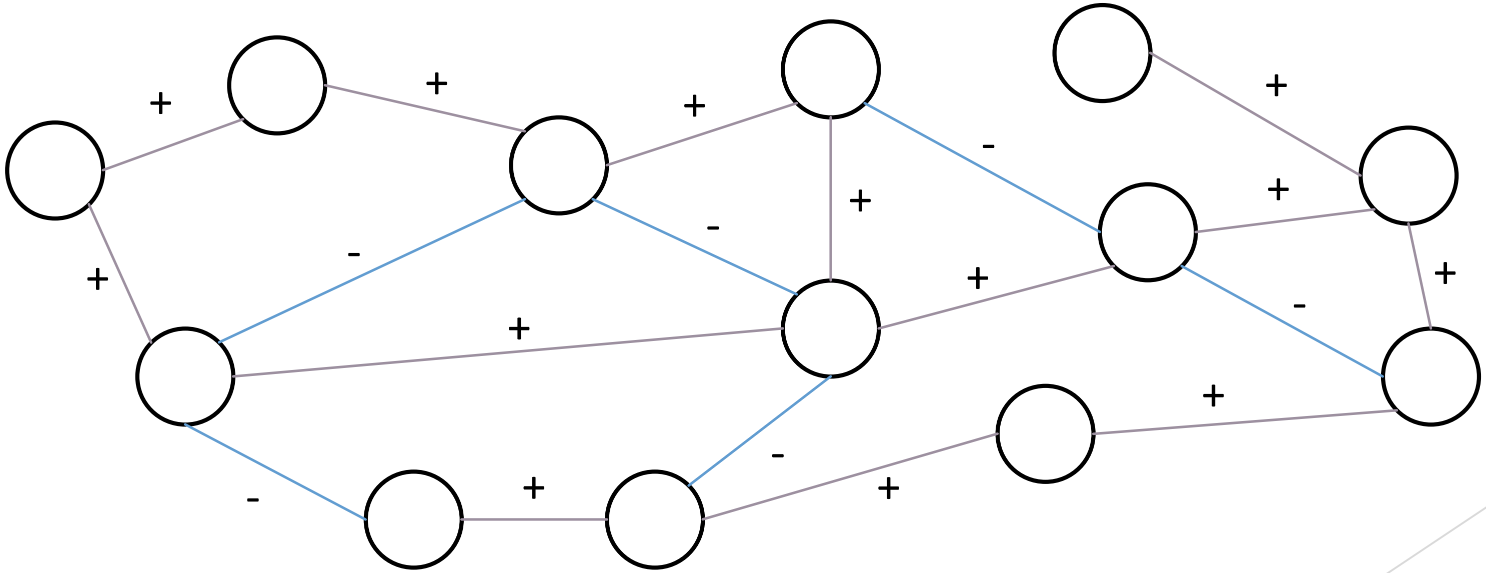
The Henry and Marilyn Taub Faculty of Computer Science, Technion, Israel

Correlation Clustering

- In the model of Correlation Clustering, we are given
  - A graph $G = (V, E)$
  - The edges are labeled by a "+" or "-" sign: $E = E^+ \cup E^-$.
  - A weight function $w: E \to \mathbb{R}^+$.
- "+" = the nodes are similar.
- "-" = the nodes are dissimilar.
- The goal: produce a clustering that agrees the most with the labels.
  - Plus edges should reside within clusters.
  - Minus edges should cross between clusters.

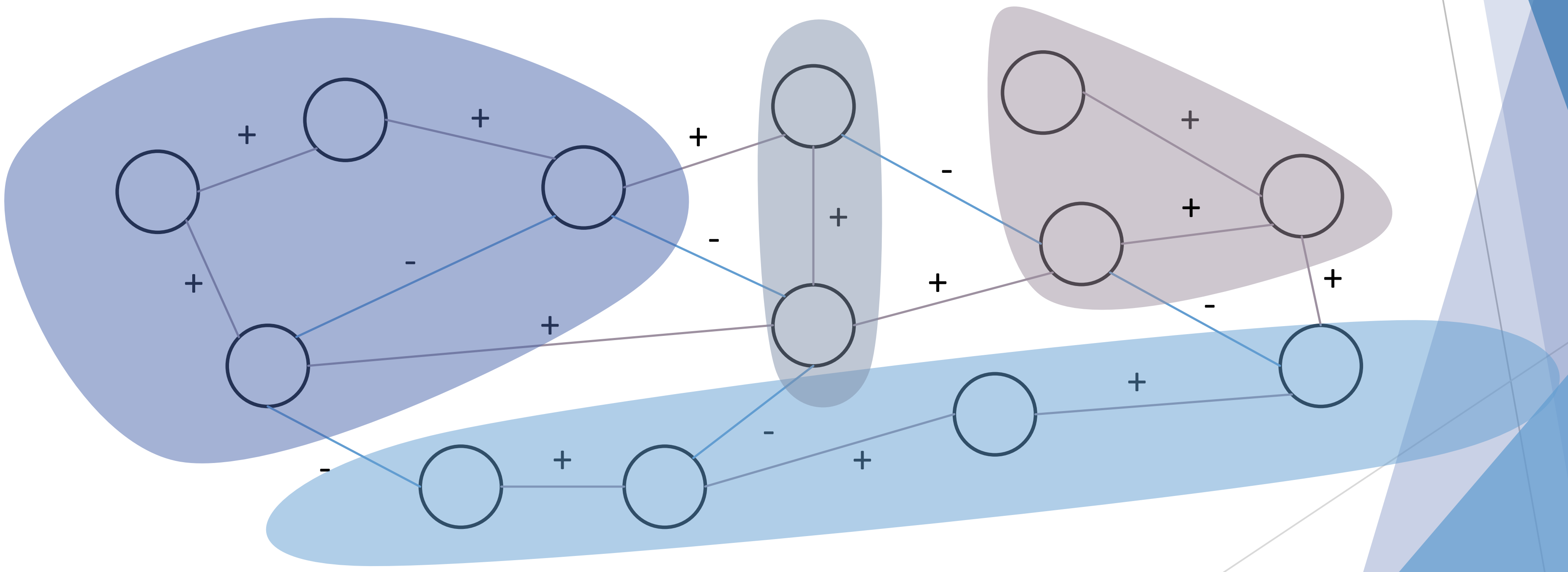
For example

- The input is a graph: $G = (V, E^+ \dot\cup E^-)$, with some edge weights $w: E \to \mathbb{R}^+$.



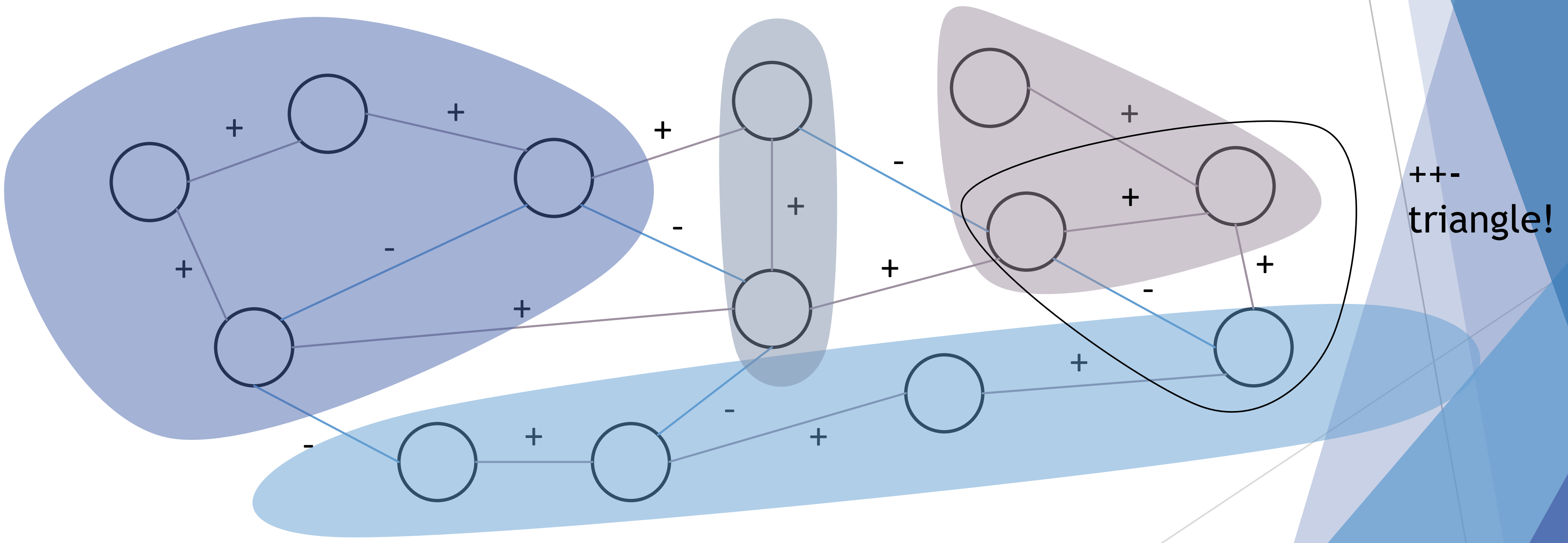
For example

- The output, will be a clustering of the graph, $C = \{C_1, C_2, .., C_k\}$



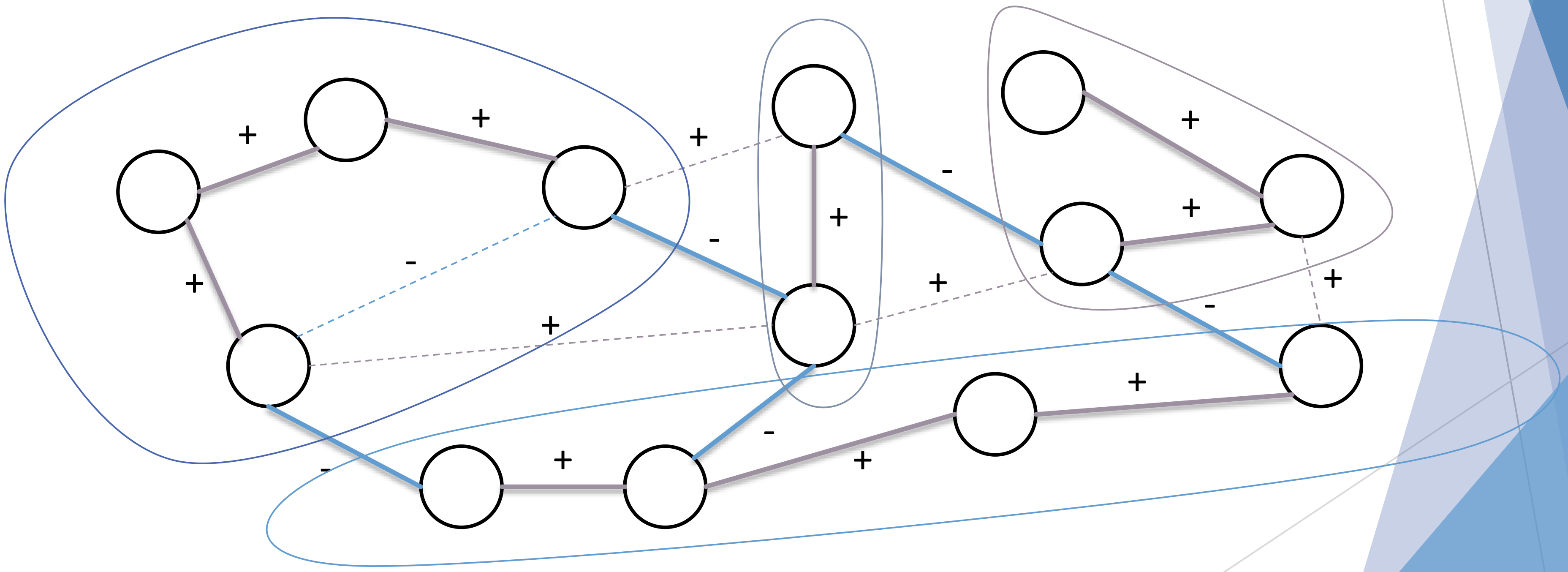
For example

- A perfect clustering does not always exist! For example, a ++- cycle can not be clustered in a way the agrees with all the edges.



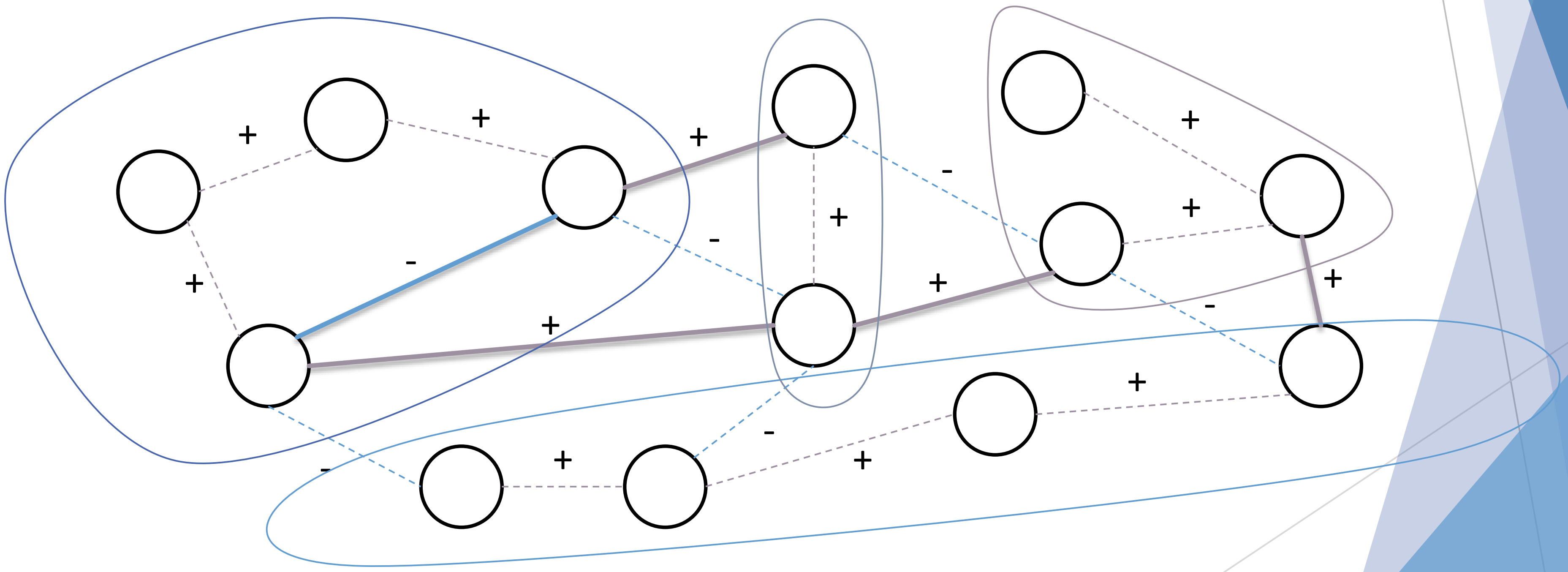
The *agreements*

- are edges that classified correctly: “+” edges within clusters, and “-” edges across clusters.



The *disagreements*

- are edges that classified incorrectly: “+” edges across clusters, and “-” within clusters.



What are the objectives?

- MaxAgree: The goal is to find a clustering that maximizes the number of *agreements*.
- MinDisagree: The goal is to find a clustering that minimizes the number of *disagreements*.
- MaxCorr: We aim to maximize the *Correlation*, which is the difference between the number of *agreements* and the number of *disagreements*.

Motivation for Correlation Clustering

Theory	Practical applications
Captures classic graph cuts problems: • Min $s - t$ Cut • Multiway Cut • Multicut • Max-Cut • etc.	• Image segmentation • Cross-lingual link detection • Clustering gene expression patterns • Coreference resolution • etc.

Previous work on Correlation Clustering

	MaxAgree	MinDisagree	MaxCorr
General graphs	0.5, 0.75, 0.766	$O(\log n)$	$\Omega(1/\log n)$
Complete unweighted graphs	PTAS	4, 2.5, 2.06	
Bipartite graphs	PTAS	11, 4, 3	
Restricted Number of clusters	For special cases	For special cases	

In our work

- We study the problem of MaxCorr on bipartite graphs.
- We extend MaxCorr to restricted number of clusters - Max-k-Corr
- We present approximation algorithms for these problems.
- and show the relation to Grothendieck’s Inequality.

MaxCorr: previous work

- [Charikar-Wirth-04] studied the problem of maximizing a quadratic form (MaxQuad):
- Given a matrix $B \in \mathbb{R}^{n \times n}$, find $x \in \{\pm 1\}^n$ such that the form $x^T B x$ is maximized. They presented an approximation algorithm with guarantee of $\Omega\left(\frac{1}{\log n}\right)$. (uses semi-definite program and random projections)
- Then, they presented an elegant reduction from MaxCorr.

The reduction from MaxCorr

- Given a graph G with signed and weighted edges, we define:

$$B_{i,j} = \begin{cases} w_{i,j}, & (i,j) \in E^+ \\ -w_{i,j}, & (i,j) \in E^- \\ 0, & otherwise \end{cases}$$

- Solve (approximately) $\max_{x \in \{\pm 1\}^n} x^T B x$ and let $S = \{C_1, C_2\}$ be the clustering induced by the solution.
- Let $T = \{\{v_1\}, \dots, \{v_n\}\}$ be the all-singletons clustering.
- Return the best out of S and T.

The reduction from MaxCorr

- Charikar and Wirth showed that using this reduction, an α-approximation for maximizing a quadratic form, turns into an $\frac{\alpha}{2+\alpha}$ approximation for MaxCorr on general graphs.
- That is, they obtained a guarantee of $\Omega\left(\frac{1}{\log n}\right)$ for the problem.

From General to Bipartite?

- The problem of maximizing a bipartite quadratic form, denote by Max-BiQuad:

  Given a matrix $A \in R^{n \times m}$, find vectors $x \in \{\pm 1\}^n$, $y \in \{\pm 1\}^m$ that maximize the form $x^T Ay$.

- Equivalent to Max-2-Corr on bipartite graphs.
- Solving Max-BiQuad, is typically achieved by rounding the natural semi-definite program.

Grothendieck's Inequality

- Grothendieck's inequality [1953] states that there is a universal constant K_G such that for every matrix $A \in \mathbb{R}^{n \times m}$,

$$\max_{\{\mathbf{u}_i\}_{i=1}^{n} \cup \{\mathbf{v}_j\}_{j=1}^{m} \subseteq S^{n+m-1}} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} A_{i,j} \langle \mathbf{u}_i, \mathbf{v}_j \rangle \right\} \leq K_G \cdot \max_{\mathbf{x} \in \{\pm 1\}^n, \mathbf{y} \in \{\pm 1\}^m} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} A_{i,j} x_i y_j \right\}$$

- Bounding K_G:
  - Krivine's algorithm shows that $K_G \leq \frac{\pi}{2 \ln(1+\sqrt{2})} \approx 1.782$.
  - [Reeds-91] showed that $K_G \geq 1.6769$.

Applying CW’s reduction to bipartite graphs

- Krivine’s 0.5611-approximation for Max-BiQuad + CW’s reduction yields a 0.219-approximation for MaxCorr on bipartite graphs.
- However, the lower bound on K_G implies a barrier of 0.2296 using this approach, since the only known algorithm for Max-BiQuad is by rounding the natural semi-definite program.
- CW’s algorithm may output a huge number of clusters.
- Therefore, we depart from this approach.

Our results

- Theorem 1: *There exists a polynomial-time* 0.254*-approximation algorithm for the problem of* MaxCorr *on bipartite graphs.*
- Theorem 2: (Non formal) Max-k-Corr admits the following:

k	2	3	4	5	6	10	….	∞
Approximation	0.5611	0.397	0.348	0.32	0.309	0.285	….	0.254

These extend Grothendieck's inequality to *large domains*.

The techniques

- We suggest a natural SDP relaxation for MaxCorr problem.
- We extend Krivine's rounding method to more than two clusters.
- We adapt the above for the problem of Max-k-corr.

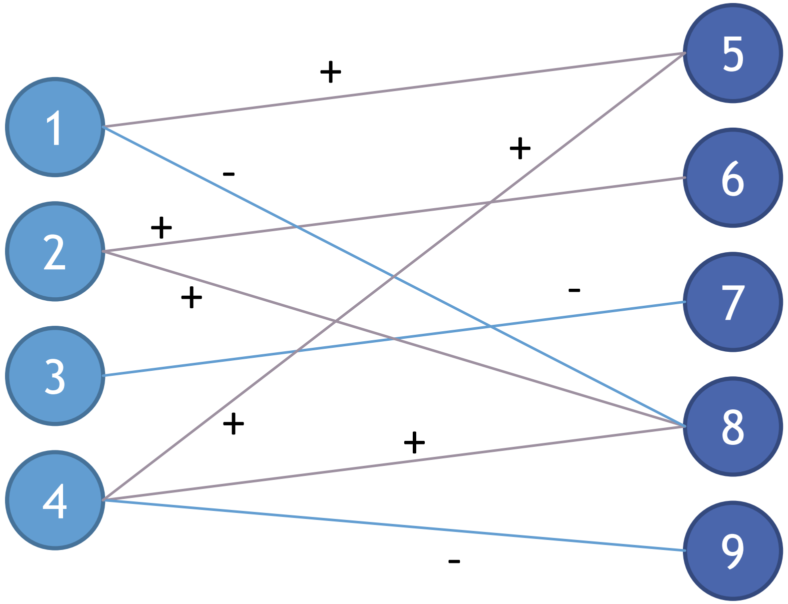
The SDP relaxation for MaxCorr

$$\begin{aligned}
\max \quad & \sum_{(u,v)\in E^+} w_{u,v}(2y_u \cdot y_v - 1) + \sum_{(u,v)\in E^-} w_{u,v}(1 - 2y_u \cdot y_v) & \\
\mathrm{s.t.} \quad & y_v \cdot y_v = 1 & \forall v \in V \\
& y_u \cdot y_v \geq 0 & \forall v, u \in V
\end{aligned}$$

The Algorithm

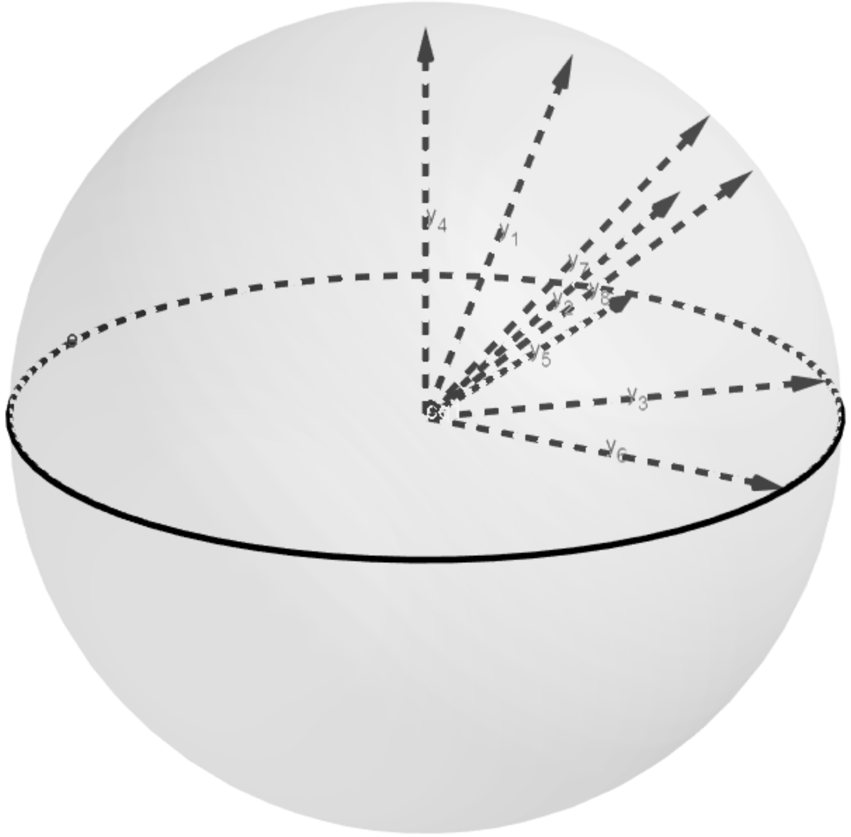
The Algorithm

1. We are given a bipartite graph, with edges labeled by a plus or minus.



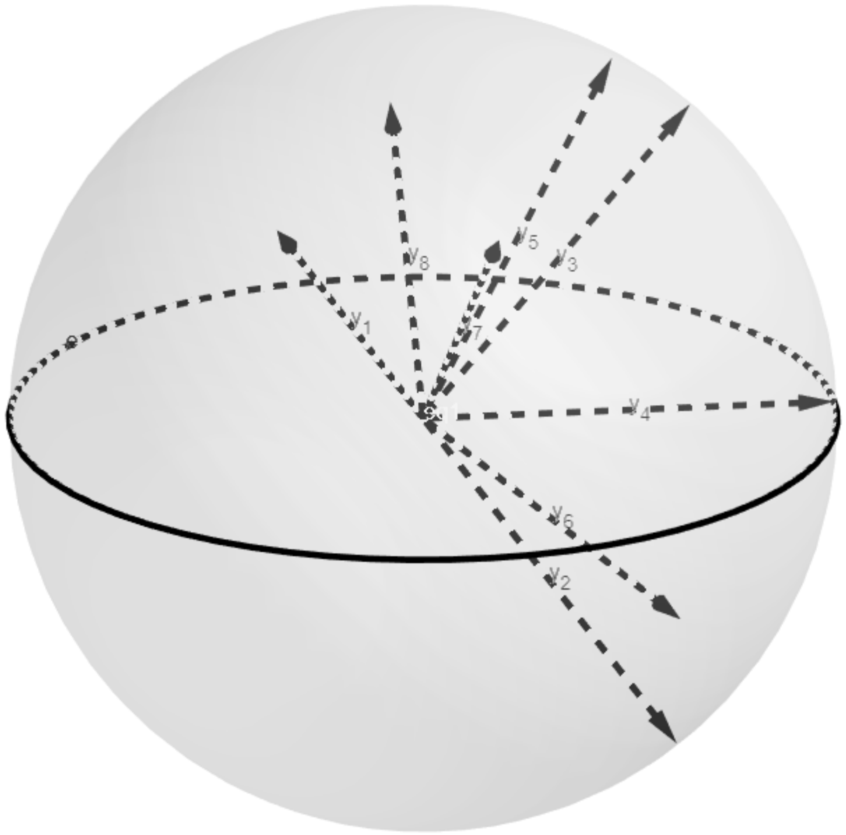
The Algorithm

1. We are given a bipartite graph, with edges labeled by a plus or minus.
2. Solve the SDP relaxation, and obtain a fractional solution $y_1, \dots, y_n$.



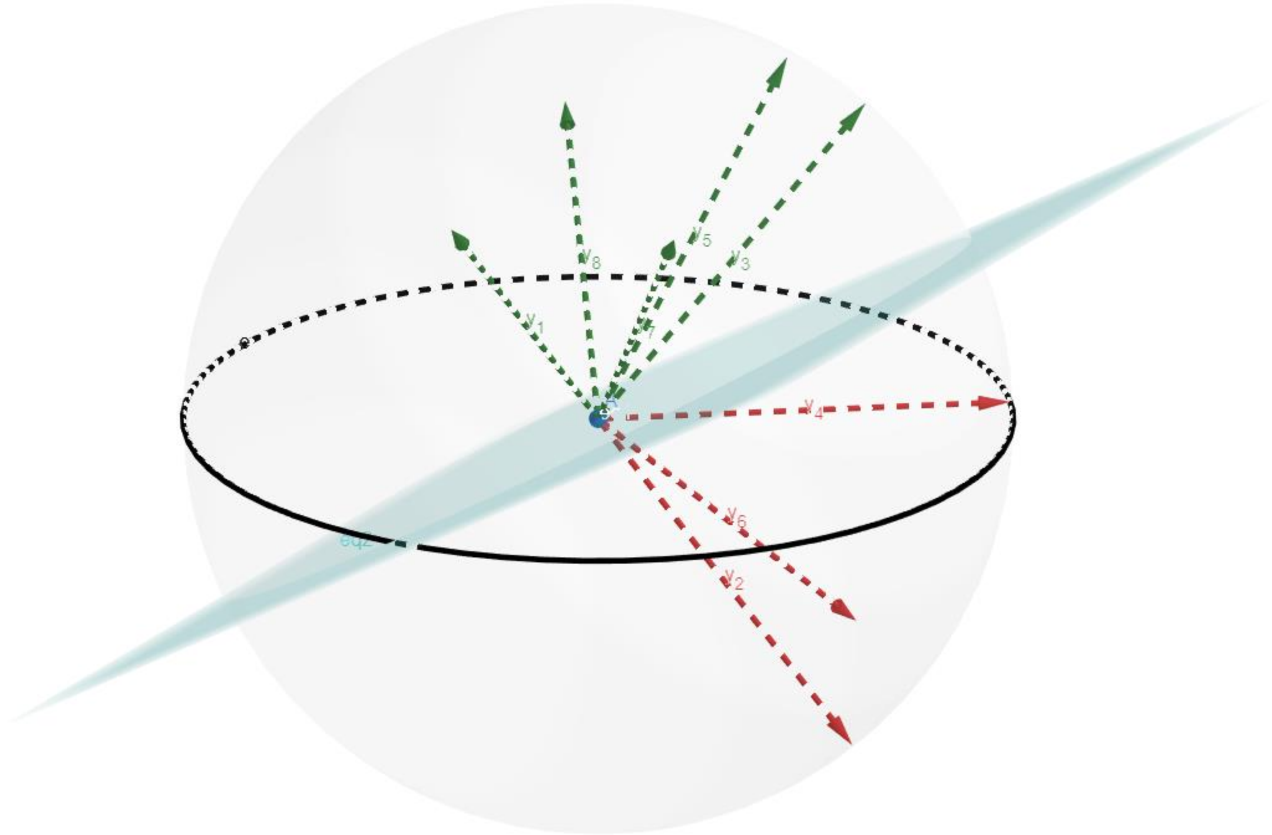
The Algorithm

1. We are given a bipartite graph, with edges labeled by a plus or minus.
2. Solve the SDP relaxation, and obtain a fractional solution $y_1, \dots, y_n$.
3. Transform the vectors to a new set of unit vectors $\tilde{y}_1, \dots, \tilde{y}_n$. The transformation is based on the bipartite structure of the graph.



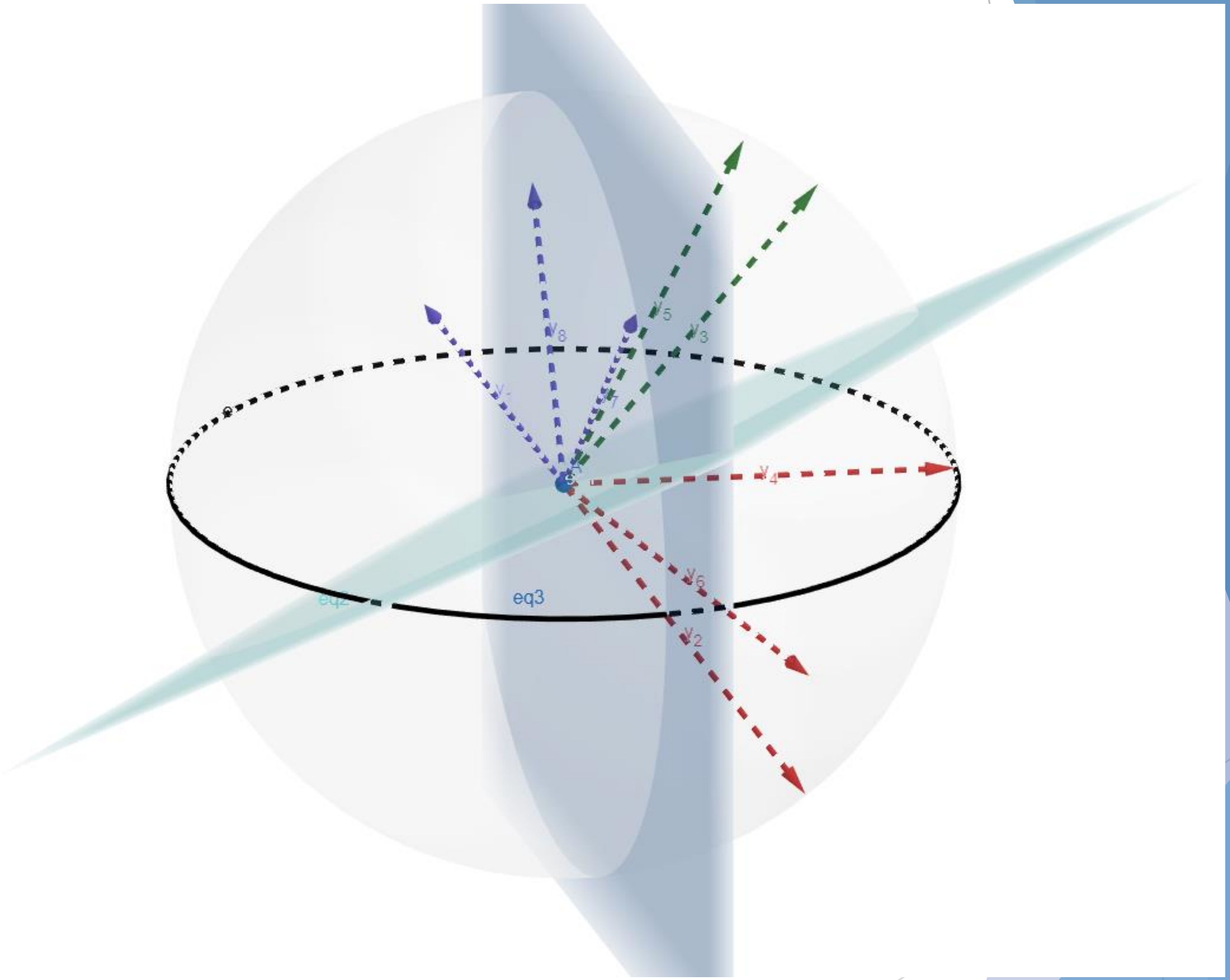
The Algorithm

1. We are given a bipartite graph, with edges labeled by a plus or minus.
2. Solve the SDP relaxation, and obtain a fractional solution $y_1, \ldots, y_n$.
3. Transform the vectors to a new set of unit vectors $\tilde{y}_1, \ldots, \tilde{y}_n$. The transformation is based on the bipartite structure of the graph.
4. Draw a unit vector, uniformly at random, that splits the hypersphere into two pieces.



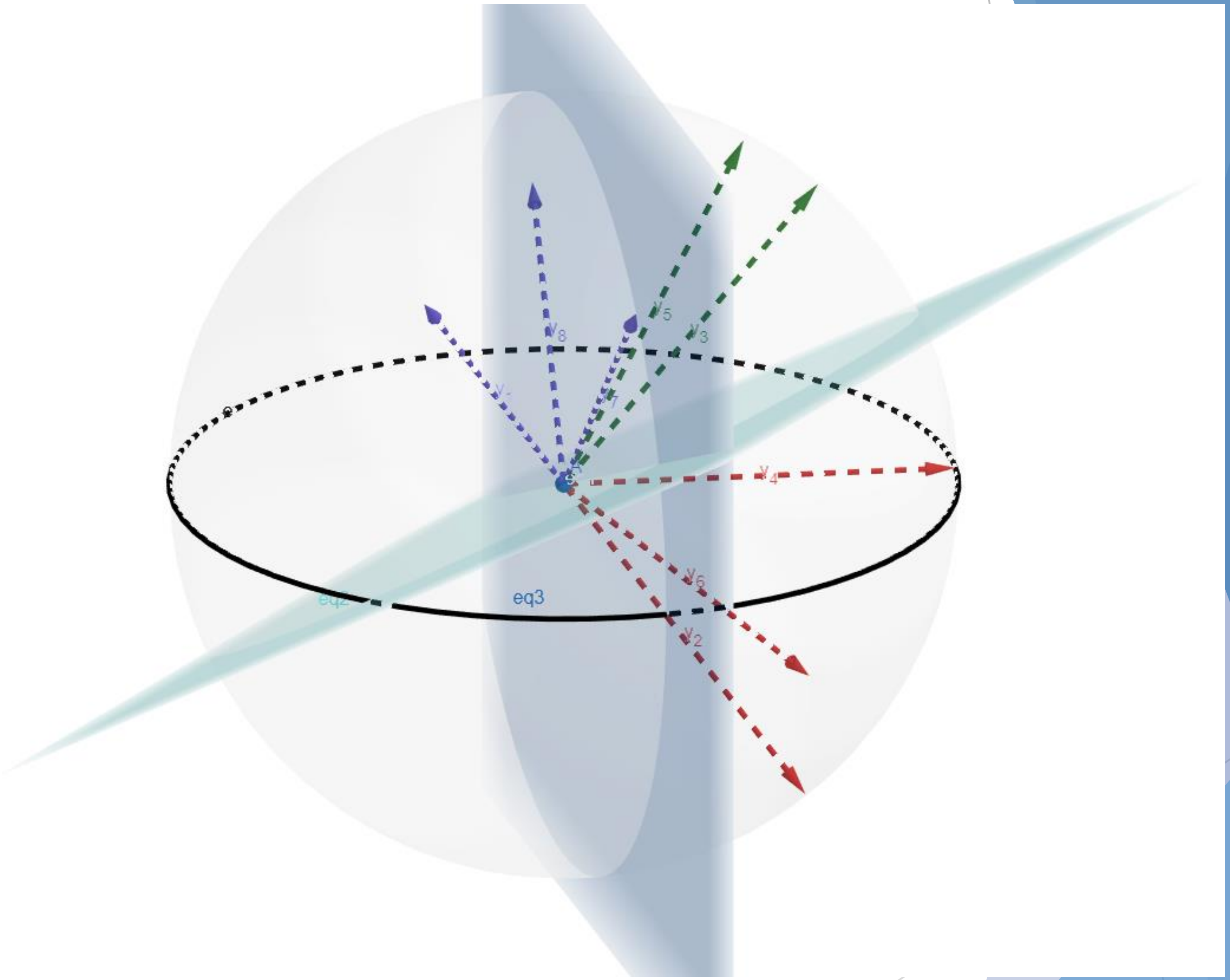
The Algorithm

1. We are given a bipartite graph, with edges labeled by a plus or minus.
2. Solve the SDP relaxation, and obtain a fractional solution $y_1, \ldots, y_n$.
3. Transform the vectors to a new set of unit vectors $\tilde{y}_1, \ldots, \tilde{y}_n$. The transformation is based on the bipartite structure of the graph.
4. Draw a unit vector, uniformly at random, that splits the hypersphere into two pieces.
5. Pick one of the pieces at random, and split it with another random hyperplane.



The Algorithm

1. We are given a bipartite graph, with edges labeled by a plus or minus.
2. Solve the SDP relaxation, and obtain a fractional solution $y_1, \ldots, y_n$.
3. Transform the vectors to a new set of unit vectors $\tilde{y}_1, \ldots, \tilde{y}_n$. The transformation is based on the bipartite structure of the graph.
4. Draw a unit vector, uniformly at random, that splits the hypersphere into two pieces.
5. Pick one of the pieces at random, and split it with another random hyperplane.
6. The output is the induced clustering by the three pieces.



The analysis of the algorithm

- For every edge $(u, v) \in E$, we denote by $X_{u,v}$ the random variable that represent the contribution of the (u, v) to the solution. $(X_{u,v} \in \{\pm w_{u,v}\})$
- We denote the contribution of $(u, v) \in E$ to the SDP fractional solution by $Z_{u,v}$.
- If we have that for absolute constant $c > 0$

$$\frac{E[X_{u,v}]}{Z_{u,v}} \geq c$$

Then we get a c-approximation for the problem.

- Problem: These values may be negative!

- Solution:
- We transform the vectors $\{y_u\}_{u \in V}$, to a new set of vectors $\{\tilde{y}_u\}_{u \in V}$.
- The new vectors will satisfy that

$$E\left[X_{u,v}\right] = c \cdot Z_{u,v}$$

for all $(u, v) \in E$, for some absolute constant $c > 0$.

- Then, we will get a c-approximation for our problem.

- First, let us calculate the expected contribution of each edge (u, v) to the solution, $E[X_{u,v}]$.
- It is widely known that if x, y are unit vectors and z is a random unit vector chosen uniformly on S^{n-1} (the n-dimensional unit sphere), then

$$\Pr[sign(z \cdot x) \neq sign(z \cdot y)] = \frac{\theta_{x,y}}{\pi}$$

Where $\theta_{x,y}$ is the angle between x and y.

- We can calculate and see that

$$E[X_{u,v}] = \begin{cases} w_{u,v}\left(1 - 3\dfrac{\tilde{\theta}_{u,v}}{\pi} + \dfrac{\tilde{\theta}_{u,v}^2}{\pi^2}\right), & (u,v) \in E^+ \\ -w_{u,v}\left(1 - 3\dfrac{\tilde{\theta}_{u,v}}{\pi} + \dfrac{\tilde{\theta}_{u,v}^2}{\pi^2}\right), & (u,v) \in E^- \end{cases}$$

where $\tilde{\theta}_{u,v}$ is the angle between the transformed vectors $\tilde{y}_u, \tilde{y}_v$.

- Recall that the contribution of each edge to the SDP solution is:

$$Z_{u,v} = \begin{cases} w_{u,v}(2y_u \cdot y_v - 1), & (u,v) \in E^+ \\ -w_{u,v}(2y_u \cdot y_v - 1), & (u,v) \in E^- \end{cases}$$

- And so our demand boils down to

$$1 - 3\frac{\theta_{u,v}}{\pi} + \frac{\theta_{u,v}^2}{\pi^2} = c \cdot (2y_u \cdot y_v - 1)$$

- Now, we can solve this equation and get the following solution:

$$\tilde{\theta}_{u,v} = \frac{1}{2}\left(3\pi - \pi\sqrt{5 - 4c + 8c(y_u \cdot y_v)}\right)$$

Where y_u, y_v are the original vectors.

- We define the following function

$$f(x) = \cos\left(\frac{1}{2}\left(3\pi - \pi\sqrt{5 - 4c + 8cx}\right)\right)$$

- We want to apply f on the matrix A, where $A_{i,j} = y_i \cdot y_j$.
- Problems:
  - We want the new matrix to be PSD.
  - We want 1's on the diagonal. (the transformed vectors will be unit vectors).

- We define the function

$$g(x) = \sum_{k=0}^{\infty} |f_k| x^k$$

where f_k is the coefficient of x^k in the Taylor expansion of f.

$$f(x) = \cos\left(\frac{1}{2}\left(3\pi - \pi\sqrt{5 - 4c + 8cx}\right)\right) \quad g(x) = \sum_{k=0}^{\infty} |f_k| x^k$$

- The transformation will be:

$$\tilde{A}_{i,j} \leftarrow \begin{cases} f(A_{i,j}) & \text{if } i \text{ and } j \text{ in different sides of } V \\ g(A_{i,j}) & \text{otherwise} \end{cases}$$

- Now, we want to show that $\tilde{A}$ is a PSD matrix.

- We denote

$$a_k = \sqrt{|f_k|}, \qquad b_k = sign(f_k) \cdot \sqrt{|f_k|}$$

- Assuming $V = (V_1, V_2)$, we define a new set of vectors $\{y_u'\}_{u \in V}$,
- If $u \in V_1$, then
  - $y_u' = (a_0, a_1 y_u, a_2(y_u \otimes y_u), a_3(y_u \otimes y_u \otimes y_u), \dots)$
- If $u \in V_2$, then
  - $y_u' = (b_0, b_1 y_u, b_2(y_u \otimes y_u), b_3(y_u \otimes y_u \otimes y_u), \dots)$
- The k-times tensor product $y_u \otimes \cdots \otimes y_u$ is in fact n^k coordinates in y_u'.

- We denote the k-times tensor product $v \otimes \cdots \otimes v$ by $v^{\otimes k}$.
- Known and useful fact:
  - $u^{\otimes k} \cdot v^{\otimes k} = (u \cdot v)^k$

  for every two vectors $u, v \in \mathbb{R}^n$ and $k \in \mathbb{N}$. (exercise)
- If u, v are in different sides of V, then

$$y'_u \cdot y'_v = \sum_{k=0}^{\infty} a_k b_k \left(y_u^{\otimes k} \cdot y_v^{\otimes k}\right) = \sum_{k=0}^{\infty} f_k (y_u \cdot y_v)^k = f(y_u \cdot y_v)$$

- And If u, v are in the same side of V (w.l.o.g V_1), then

$$y'_u \cdot y'_v = \sum_{k=0}^{\infty} a_k^2 \left(y_u^{\otimes k} \cdot y_v^{\otimes k}\right) = \sum_{k=0}^{\infty} |f_k| (y_u \cdot y_v)^k = g(y_u \cdot y_v)$$

- Note that in the analysis we only care for the edges $(u, v) \in E$, and so u, v must be on different sides.

- Now we only have to show that the new vectors remain unit vectors.
- That is, we want to show that $y'_u \cdot y'_u = 1$ for all $u \in V$. Indeed,

$$y'_u \cdot y'_u = \sum_{k=0}^{\infty} |f_k| (y_u \cdot y_u)^k = g(y_u \cdot y_u) = g(1)$$

- What is $g(1)$?
- It is quite technical, but one can show that

$$g(x) = f(-x) - 2f_0 + 2f_1 x - 2f_2 x^2$$

- $g(1)$ depends only on c. Therefore, if c is the solution for $g(1) = 1$, we are done.
- The solution is $c \approx 0.254$, and so is the approximation factor.

- This completes the proof.

- More
  - The above algorithm can be improved slightly:
  - Instead of just splitting into 3 clusters, we randomly chose between 2 and 4 clusters.
  - The 3 clusters algorithm, is like clustering to 2 clusters w.p $\frac{1}{2}$ or 4 clusters w.p $\frac{1}{2}$ (exercise ☺)
  - If we chose to cluster into 2 clusters w.p. $p = 0.49$ or 4 clusters w.p. $1 - p$, we get a 0.2551-approximation.
    - Which is very close to the simple three clusters algorithm.
  - Will more clusters help?

Extending to Max-k-Corr

- We define an SDP relaxation, which will depend on maximal number of clusters - k.
- We use a similar approach in the rounding algorithm and the analysis.

Questions?

Thank you!