A Faster Algorithm for Constrained Correlation Clustering

Jonas Klausen

With: Nick Fischer, Evangelos Kipouridis, and Mikkel Thorup

UNIVERSITY OF COPENHAGEN

BASIC ALGORITHMS RESEARCH COPENHAGEN

INSAIT

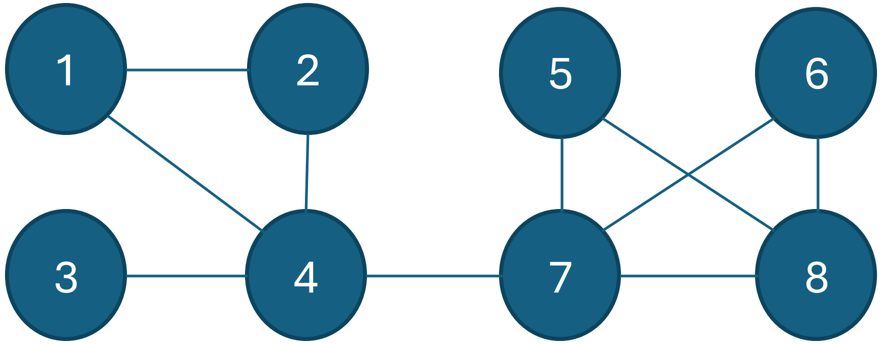
MAX PLANCK INSTITUTE FOR INFORMATICS

(Unconstrained) Correlation Clustering

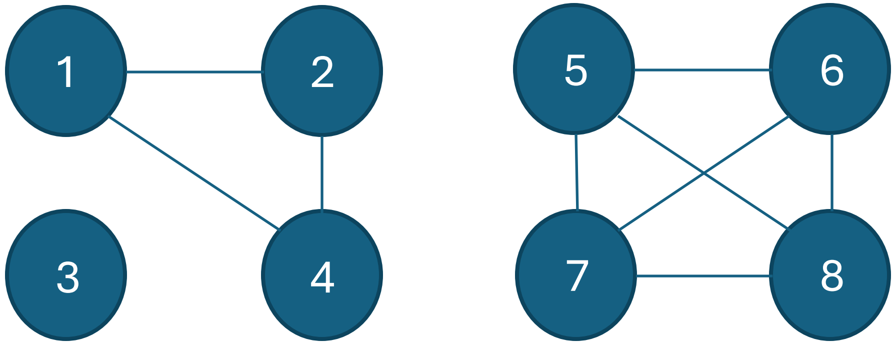
- **Input:** A complete set of *preferences*
- **Output:** Partition of vertices violating as few preferences as possible

(Unconstrained) Correlation Clustering

Input



Output (cost 3)



(Unconstrained) Correlation Clustering

- **Input:** A complete set of *preferences*

- **Output:** Partition of vertices violating as few preferences as possible

- Introduced by Bansal, Blum, and Chawla [Machine Learning '04]
- Randomized 3-apx by Ailon, Charikar, and Newman [STOC '05]
- 1.437-apx by Cao *et al.* [STOC '24]

Constrained Correlation Clustering

- **Input:** A complete set of *preferences,*
  a subset of which are *hard constraints*
- **Output:** Partition of vertices violating as few preferences as possible, and satisfying all hard constraints

- Introduced by van Zuylen, Hegde, Jain, and Williamson [SODA '07]
- 3-approximation in $O(n^{3\omega})$ time

Constrained Correlation Clustering

- **Input:** A complete set of *preferences,* a subset of which are *hard constraints*
- **Output:** Partition of vertices violating as few preferences as possible, and satisfying all hard constraints

- Introduced by van Zuylen, Hegde, Jain, and Williamson [SODA '07]
- 3-approximation in $O(n^{3\omega})$ time
- **New:** 16-approximation in $\tilde{O}(n^3)$ time

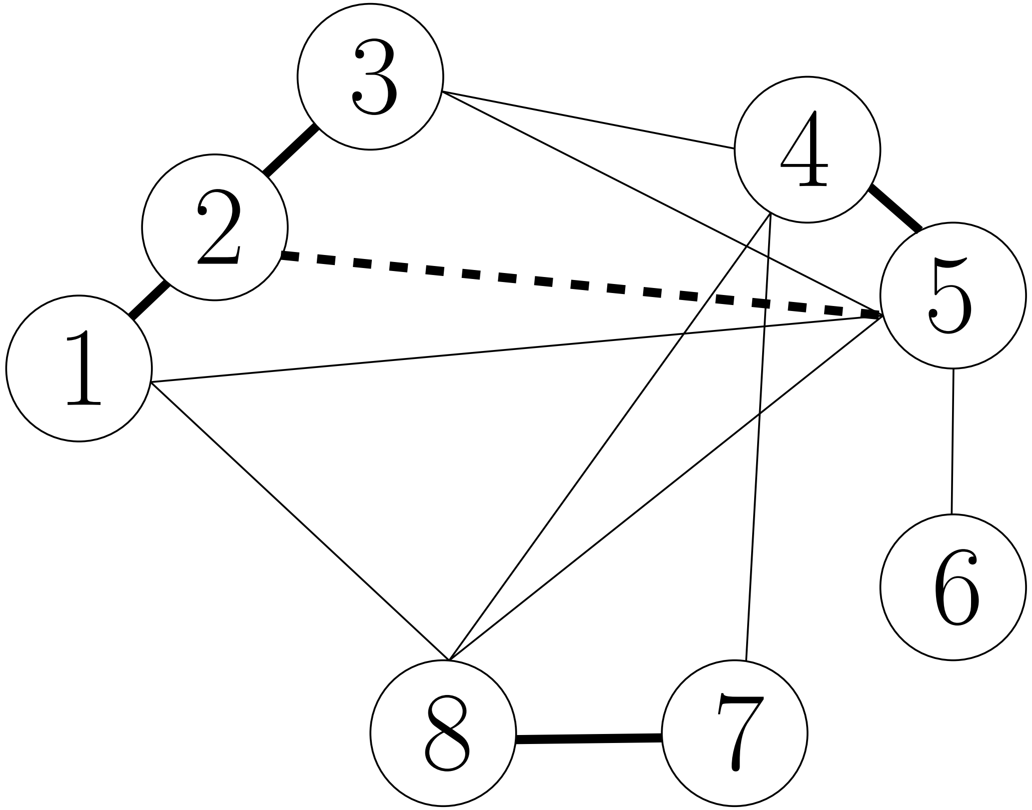
A Fast Approximation Algorithm (**new**)

1. Modify instance to one having *nice* neighborhoods
2. Forget hard constraints
3. Run *Pivot* algorithm on unconstrained instance

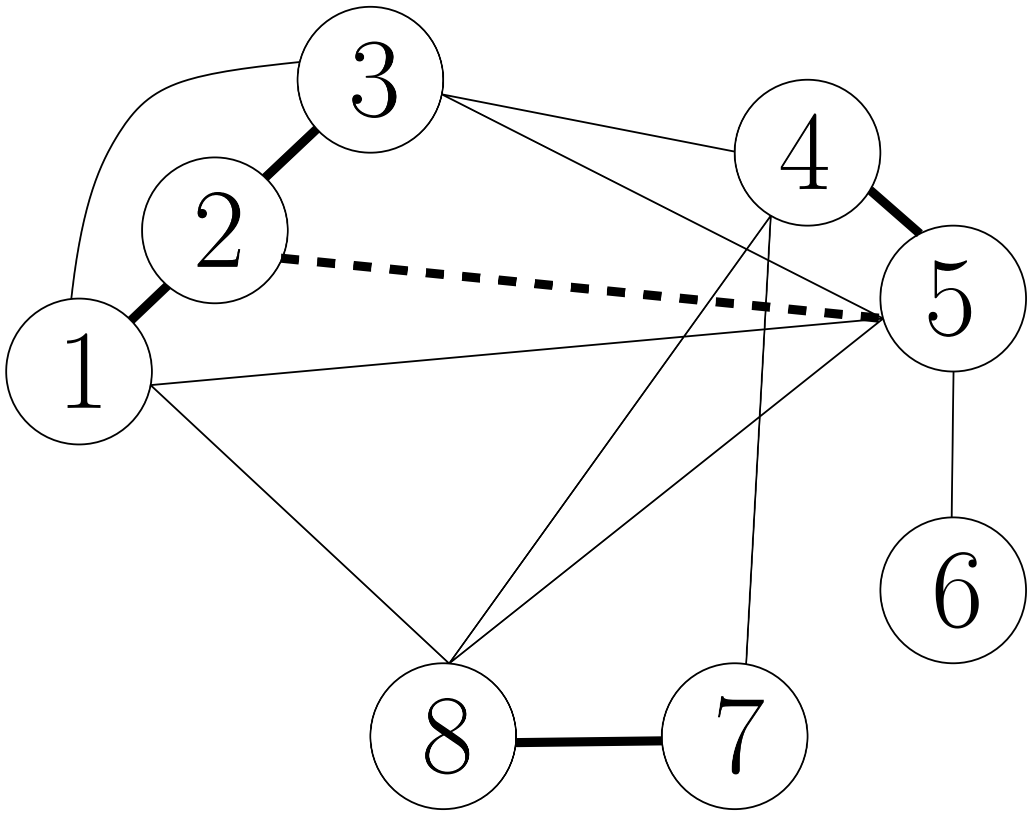
Nice Neighborhoods

- If (u, v) is a *friendly* pair they now have the same neighborhood
- If (u, v) is a *hostile* pair they now have no common neighbor

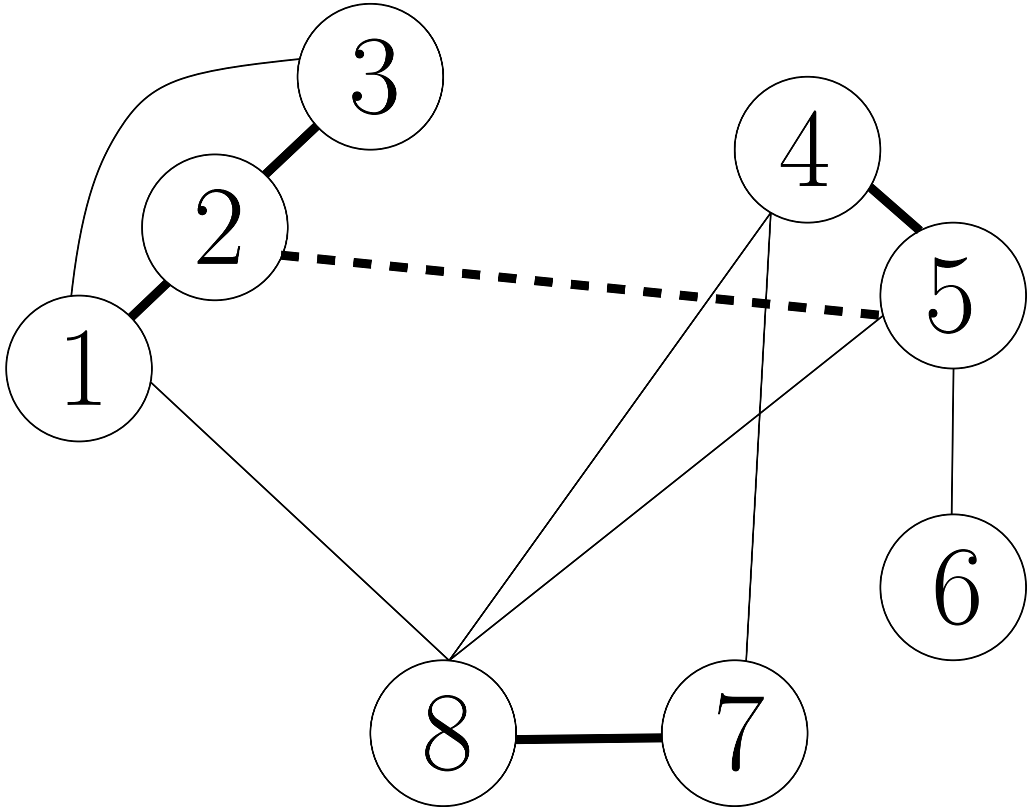
Transforming to Unconstrained Clustering



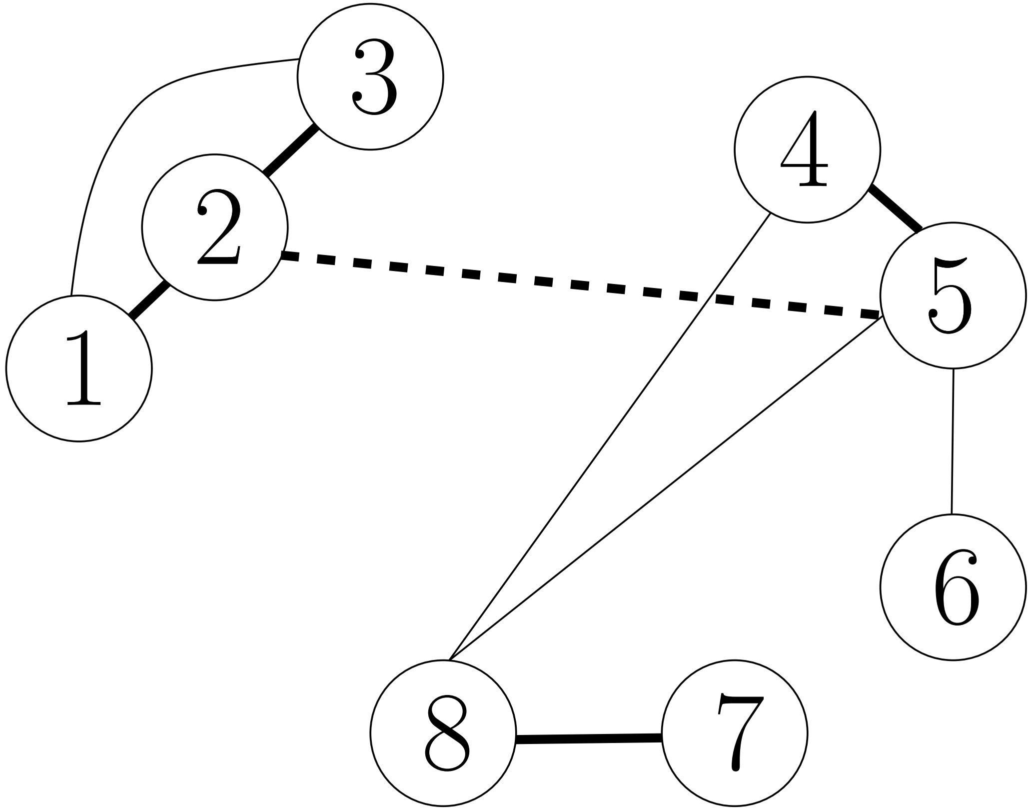
Transforming to Unconstrained Clustering



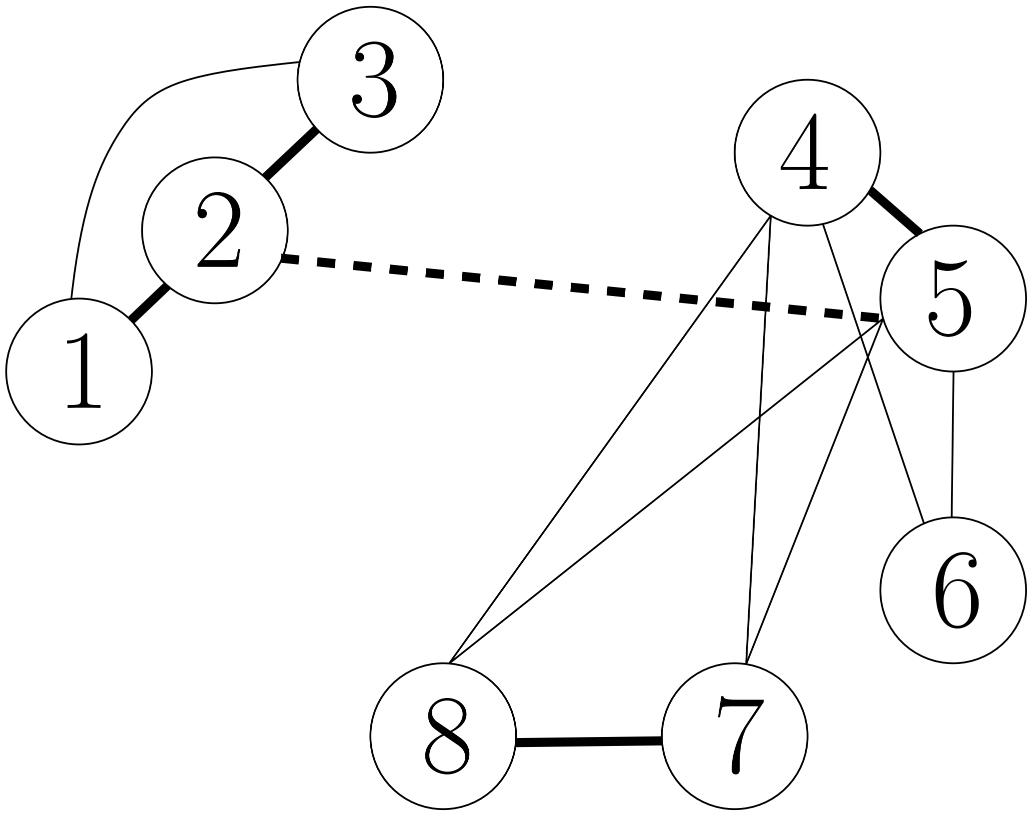
Transforming to Unconstrained Clustering



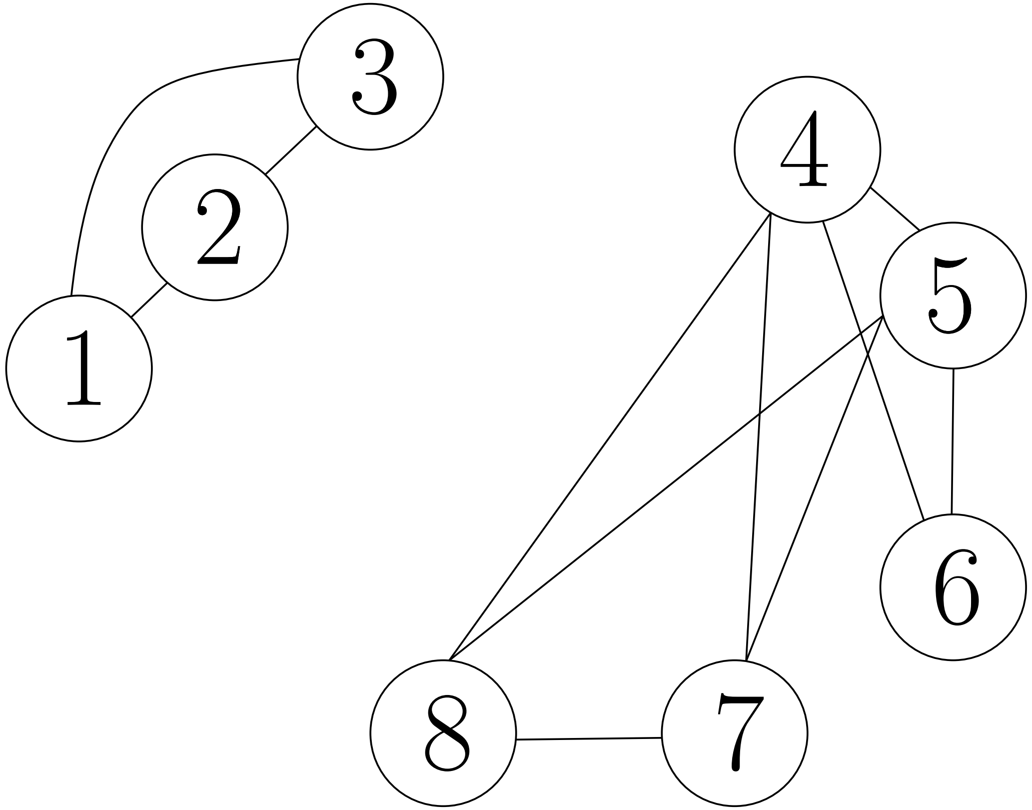
Transforming to Unconstrained Clustering



Transforming to Unconstrained Clustering



Transforming to Unconstrained Clustering



Nice Neighborhoods

- If (u, v) is a *friendly* pair they now have the same neighborhood
- If (u, v) is a *hostile* pair they now have no common neighbor

Pivot for Unconstrained Correlation Clustering

1. $C = \emptyset$
2. while $V \neq \emptyset$ do
   3. Choose a pivot node u
   4. Add a cluster containing u and all its neighbors to C
   5. Remove u and its neighborhood from G
6. return C

Pivot for Unconstrained Correlation Clustering

1. $C = \emptyset$
2. while $V \neq \emptyset$ do
    3. Choose a pivot node u **(but how?)**
    4. Add a cluster containing u and all its neighbors to C
    5. Remove u and its neighborhood from G
6. return C

Properties

- APX: $(1+\sqrt{5}) + (2+\sqrt{5}) \cdot$ Apx(Pivot)
- Time: $O(n\, m)$ + Pivot

  m: number of 'together'-preferences

Properties

- APX: $(1+\sqrt{5}) + (2+\sqrt{5}) \cdot \text{Apx(Pivot)}$
- Time: $O(n\, m)$ + Pivot
  - m: number of 'together'-preferences
- Uniform Random [ACN, STOC '05]
  - (Expected) 3-Apx, linear time

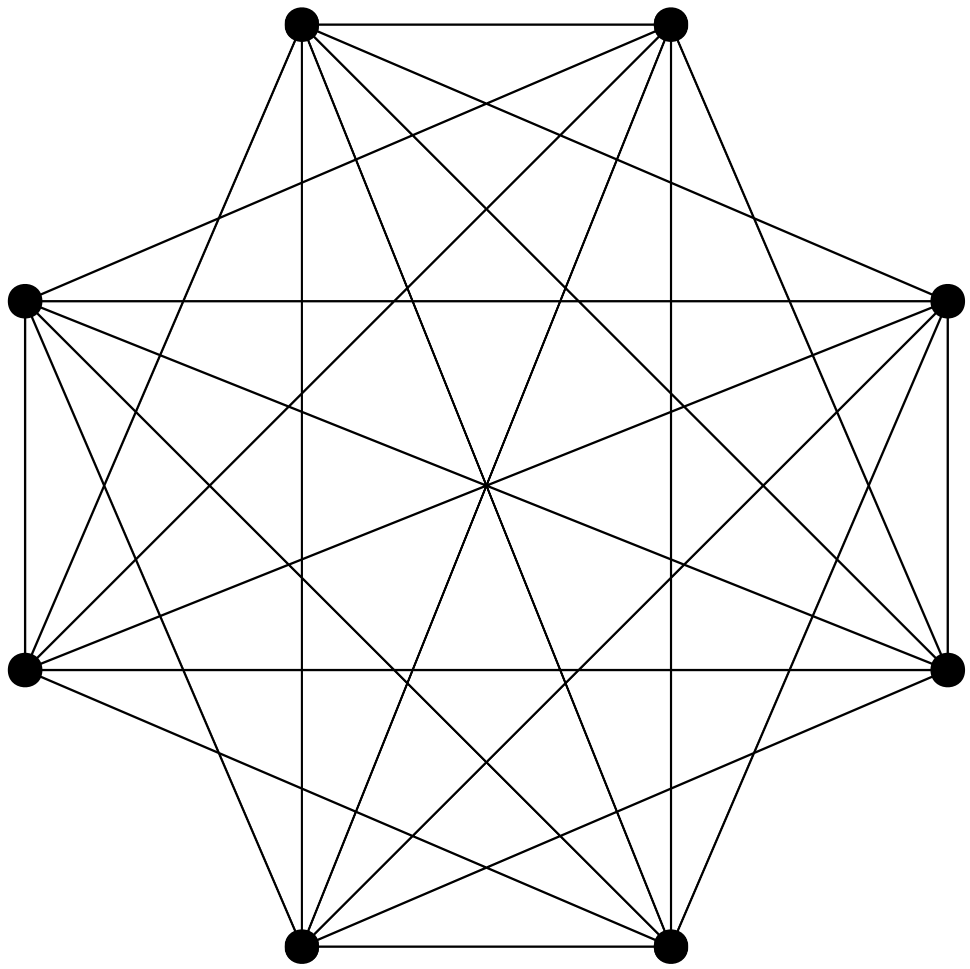
Properties

- APX: $(1+\sqrt{5}) + (2+\sqrt{5}) \cdot$ Apx(Pivot)
- Time: $O(n\,m)$ + Pivot
  m: number of 'together'-preferences
- Uniform Random [ACN, STOC '05]
  - (Expected) 3-Apx, linear time
- Deterministic (**new**)
  - (3+ε)-Apx, $\tilde{O}(n^3)$ time
  - Solves a *covering* LP

Properties

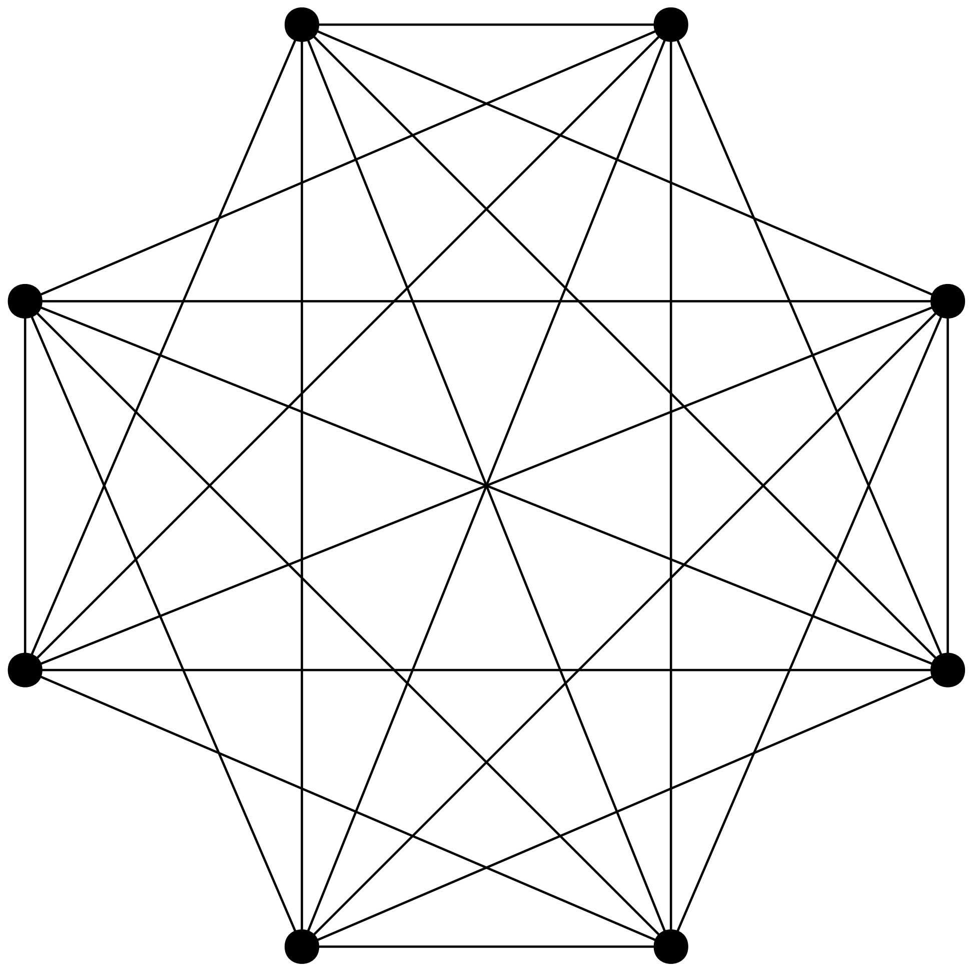
- APX: $(1+\sqrt{5}) + (2+\sqrt{5}) \cdot$ Apx(Pivot)
- Time: $O(n\ m)$ + Pivot
  - m: number of 'together'-preferences
- Uniform Random [ACN, STOC '05]
  - (Expected) 3-Apx, linear time
- Deterministic (**new**)
  - (3+ε)-Apx, $\tilde{O}(n^3)$ time
  - Solves a *covering* LP
- **Q:** Better approximations in $\tilde{O}(n^3)$ time?

The Limit of Pivoting



The Limit of Pivoting

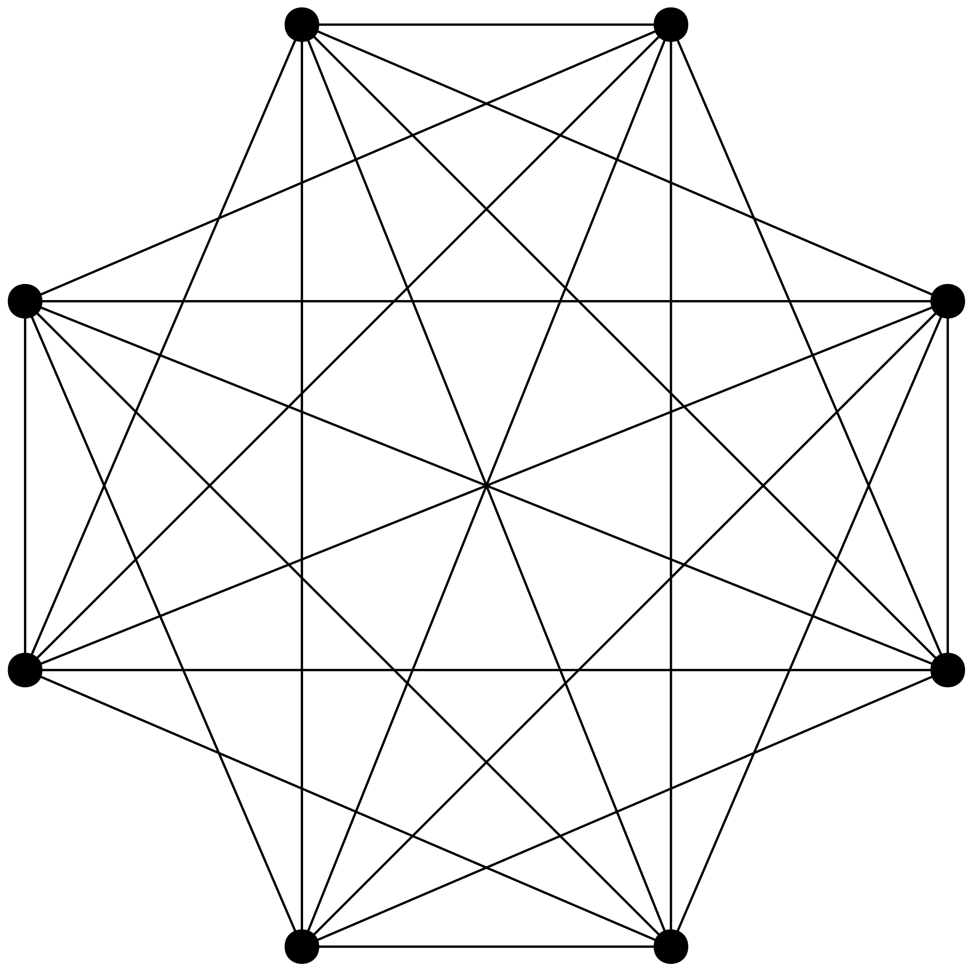
- OPT: A single cluster, cost = $n/2$



The Limit of Pivoting

- OPT: A single cluster,
  cost = $n/2$

- PIVOT: (n-1) vertices + singleton,
  cost ≈ 1/2 n + n



Thank You

Jonas Klausen

jok@di.ku.dk