# Correlation Clustering

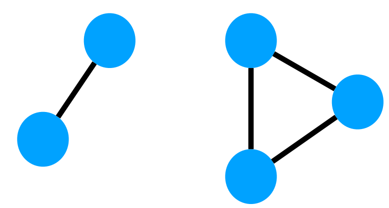
Sanjay Subramanian
PACT, Summer 2020

# Clustering

- Goal of clustering is to group similar objects together and dissimilar objects separately
- We assume that we are given pairwise similarity scores
- Some formulations (e.g. K-means) assume that we are given the number of clusters beforehand

# Correlation Clustering

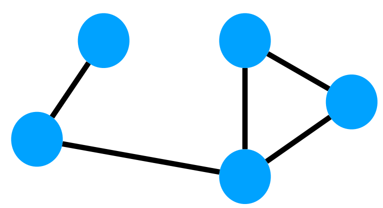
- Given a complete graph G = (V, E), each edge is labeled with a + or a -.



- Goal: cluster vertices so that we either:
    - Maximize: (# of + edges within clusters) + (# of - edges crossing clusters)
    - Minimize: (# of + edges crossing clusters) + (# of - edges within clusters)
- Note: number of clusters is not given
- Introduced by Bansal, Blum, Chawla (2002)

# Correlation Clustering

- Given a complete graph G = (V, E), each edge is labeled with a + or a -.



- Goal: cluster vertices so that we either:
    - Maximize: (# of + edges within clusters) + (# of - edges crossing clusters)
    - Minimize: (# of + edges crossing clusters) + (# of - edges within clusters)
- Note: number of clusters is not given
- Introduced by Bansal, Blum, Chawla (2002)

# Outline of Talk

1. **Introduction to the Problem**

2. Simple 3-approximation algorithm

3. Pairwise query oracle

4. NP-completeness proof

# Introductory Notes

- Why do we have a maximization and a minimization version of the problem?

# Introductory Notes

- Why do we have a maximization and a minimization version of the problem?
    - Relevant for approximation-algorithms

# Introductory Notes

- Why do we have a maximization and a minimization version of the problem?
    - Relevant for approximation-algorithms
- We will focus on the minimization version. Each + edge crossing clusters and each - minus edge within a cluster is called a “mistake.”

# Introductory Notes

- Why do we have a maximization and a minimization version of the problem?
  - Relevant for approximation-algorithms
- We will focus on the minimization version. Each + edge crossing clusters and each - minus edge within a cluster is called a “mistake.”
- What is a quick upper bound on # of mistakes?

# Introductory Notes

- Why do we have a maximization and a minimization version of the problem?
  - Relevant for approximation-algorithms
- We will focus on the minimization version. Each + edge crossing clusters and each - minus edge within a cluster is called a “mistake.”
- What is a quick upper bound on # of mistakes?
  - $|E|/2 = n(n-1)/4$

# More Introductory Notes

- Efficient algorithm for finding optimal clustering OPT when OPT makes 0 mistakes?

# More Introductory Notes

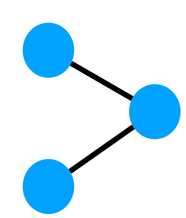
- Efficient algorithm for finding optimal clustering OPT when OPT makes 0 mistakes?
  - Connected components of subgraph with only + edges

# More Introductory Notes

- Efficient algorithm for finding optimal clustering OPT when OPT makes 0 mistakes?
  - Connected components of subgraph with only + edges
- Let $C_{OPT}$ be # of mistakes of OPT
- What feature of graph determines whether $C_{OPT} > 0$?

# More Introductory Notes

- Efficient algorithm for finding optimal clustering OPT when OPT makes 0 mistakes?
    - Connected components of subgraph with only + edges
- Let $C_{OPT}$ be # of mistakes of OPT
- What feature of graph determines whether $C_{OPT} > 0$?
- (+, +, -) triangle



# Outline of Talk

1. Introduction to the Problem
2. **<u>Simple 3-approximation algorithm</u>**
3. Pairwise query oracle
4. NP-completeness proof

# 3-approx. Algorithm

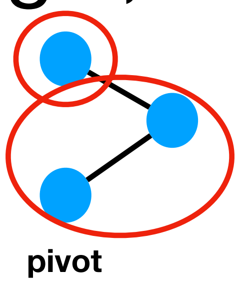
- Similar to the algorithm given before!
- Select a vertex uniformly at random as the “pivot.”
- Form a cluster with pivot and its +-neighbors
- Repeat with remaining vertices
- Algorithm+analysis published by Ailon, Charikar, Newman (2005)

# 3 Approx. Analysis

- When does the algorithm make a mistake?

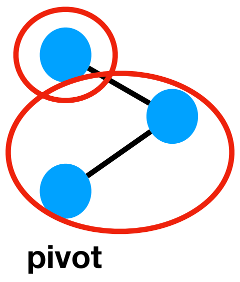
# 3 Approx. Analysis

- When does the algorithm make a mistake?
  - When the pivot is part of a (+, +, -) triangle, mistake is made on edge opposite the pivot



# 3 Approx. Analysis

- When does the algorithm make a mistake?
    - When the pivot is part of a (+, +, -) triangle, mistake is made on edge opposite the pivot




- For a (+, +, -) triangle $t$, let $A_t$ be the event that all vertices are still unclustered in a recursive call when one of the three vertices is the pivot. Let $T$ be the set of all (+, +, -) triangles. Then

$$E[\text{mistakes}] = \sum_{t \in T} \Pr[A_t]$$

# 3 Approx. Analysis

- Consider the linear program:

$$\text{minimize} \sum_{e \in E} x_e$$

$$\text{s.t. } x_{e_1} + x_{e_2} + x_{e_3} \geq 1 \ \forall \{e_1, e_2, e_3\} \in T$$

$$x_e \geq 0 \ \forall e \in E$$

# 3 Approx. Analysis

- Consider the linear program:

$$\text{minimize} \sum_{e \in E} x_e$$

$$\text{s.t. } x_{e_1} + x_{e_2} + x_{e_3} \geq 1 \ \forall \{e_1, e_2, e_3\} \in T$$

$$x_e \geq 0 \ \forall e \in E$$

- Using OPT, we can construct a feasible solution to the LP: for each edge $e$, $x_e = 1$ if OPT makes a mistake on $e$ and $x_e = 0$ otherwise.

# 3 Approx. Analysis

- Consider the linear program:

$$\text{minimize } \sum_{e \in E} x_e$$

$$\text{s.t. } x_{e_1} + x_{e_2} + x_{e_3} \geq 1 \ \forall \{e_1, e_2, e_3\} \in T$$

$$x_e \geq 0 \ \forall e \in E$$

- Using OPT, we can construct a feasible solution to the LP: for each edge $e$, $x_e = 1$ if OPT makes a mistake on $e$ and $x_e = 0$ otherwise.
- For this definition of $x_e$, $C_{OPT} = \sum_{e \in E} x_e$.
- Therefore, $C_{OPT}$ is lower-bounded by the optimal LP cost.

# 3 Approx. Analysis

- Now consider the dual LP:

$$\text{maximize } \sum_{t \in T} \beta_t$$

$$\sum_{t:e \in t} \beta_t \leq 1 \;\; \forall e \in E$$

- Here is a feasible solution to the dual LP: $\beta_t = \dfrac{\Pr[A_t]}{3}$

- Why is this solution feasible?

# 3 Approx. Analysis

- Now consider the dual LP:

$$\text{maximize} \sum_{t \in T} \beta_t$$

$$\sum_{t: e \in t} \beta_t \leq 1 \ \forall e \in E$$

- Here is a feasible solution to the dual LP: $\beta_t = \dfrac{\Pr[A_t]}{3}$

- Why is this solution feasible?

    - For a given edge $e$, let $B_e$ be the event that algorithm makes a mistake on $e$.

    - $\Pr[B_e \cap A_t] = \Pr[B_e \,|\, A_t] \Pr[A_t] = \dfrac{1}{3} \Pr[A_t]$

# 3 Approx. Analysis

- Now consider the dual LP:

$$\text{maximize} \sum_{t \in T} \beta_t$$

$$\sum_{t : e \in t} \beta_t \leq 1 \ \forall e \in E$$

- Here is a feasible solution to the dual LP: $\beta_t = \dfrac{\Pr[A_t]}{3}$

- Why is this solution feasible?

  - For a given edge $e$, let $B_e$ be the event that algorithm makes a mistake on $e$.

  - $\Pr[B_e \cap A_t] = \Pr[B_e \,|\, A_t] \Pr[A_t] = \dfrac{1}{3} \Pr[A_t]$

  - Finally, note that for two triangles $t$ and $t'$, $(B_e \cap A_t) \cap (B_e \cap A_{t'}) = \emptyset$

# 3 Approx. Analysis

- We have shown that $\beta_t = \dfrac{\Pr[A_t]}{3}$ is a feasible solution to the dual LP.

- Recall that $C_{OPT}$ is lower-bounded by the optimal LP cost. Therefore, $C_{OPT} \geq \sum_{t \in T} \beta_t^* \geq \sum_{t \in T} \dfrac{\Pr[A_t]}{3}$, where $\beta^*$ denotes an optimal solution to the dual LP.

- Since $E[\text{mistakes}] = \sum_{t \in T} \Pr[A_t]$, $E[\text{mistakes}] \leq 3C_{OPT}$.

# State-of-the-art

- The best algorithm (to my knowledge) gives a 2.06 approximation using LP-rounding (Chawla et al. 2015)
- There is a lot of work on other versions of the problem, e.g.
  - when edges have weights between 0 and 1,
  - when the number of clusters is treated as a constant,
  - when the number of mistakes is treated as a constant,
  - when edge weights are drawn from some distribution, …

# Outline of Talk

1. Introduction to the Problem

2. Simple 3-approximation algorithm

3. **Pairwise query oracle**

4. NP-completeness proof

# Same-cluster queries

- Suppose we have access to an oracle that knows an optimal clustering $OPT$ and that can answer for any given two vertices $u, v$, "Does $OPT$ put $u$ and $v$ in the same cluster?"

# Same-cluster queries

- Suppose we have access to an oracle that knows an optimal clustering $OPT$ and that can answer for any given two vertices $u, v$, "Does $OPT$ put $u$ and $v$ in the same cluster?"

- Obviously, we can easily find $OPT$ by making $O(n^2)$ queries to the oracle.

# Same-cluster queries

- Suppose we have access to an oracle that knows an optimal clustering $OPT$ and that can answer for any given two vertices $u, v$, "Does $OPT$ put $u$ and $v$ in the same cluster?"

- Obviously, we can easily find $OPT$ by making $O(n^2)$ queries to the oracle.

- But can we make fewer queries and either (1) find $OPT$ or (2) get a better approximation factor?

    - "Correlation Clustering with Same-cluster queries bounded by Optimal Cost", Saha and Subramanian, 2019

# Same-cluster queries

- Why is this a realistic/useful setting?
    - Crowdsourcing has become a popular method of obtaining annotations. We might obtain initial pairwise scores using some algorithm/model and then issue queries to crowd workers for a small set of vertex pairs.
    - Related to the machine learning paradigm called “active learning”

# Finding $OPT$ with $2C_{OPT}$ queries

- How can we modify the 3-approx. Algorithm to achieve this result?

# Finding $OPT$ with $2C_{OPT}$ queries

- How can we modify the 3-approx. Algorithm to achieve this result?
  - The 3 “came from” having a 1/3 chance of choosing the edge $OPT$ makes a mistake on in a given triangle
  - Can we instead find which edge $OPT$ makes a mistake on in each triangle by paying at most 2 queries for each of $OPT$'s mistakes?
- Extension: Query each triangle with probability $p = 0.25$ –> 2-approximation with $C_{OPT}$ queries (in expectation)

# RandomQueryPivot in Detail

- Pick pivot uniformly at random
- For each (+, +, -) triangle containing the pivot:
  - Let pivot be $u$, and let other two vertices be $v, w$.
  - With probability $p$:
    - WLOG assume $\{u, v\}$ is a + edge. Query $\{u, v\}$.
    - If $\{u, w\}$ is a + edge OR OPT doesn't make a mistake on $\{u, v\}$, query $\{u, w\}$
- For all edges adjacent to pivot, make decision according to oracle if queried and otherwise according to edge weight (+/-)

# Interesting properties of RandomQueryPivot

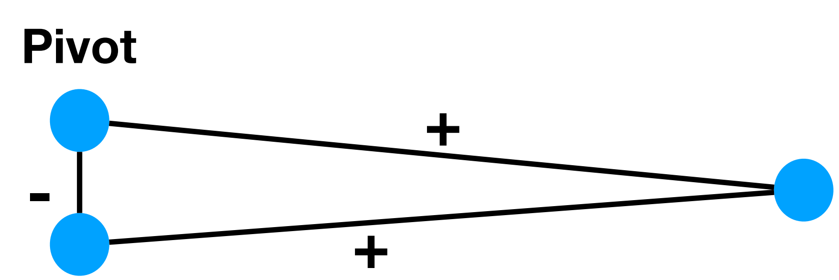
- What is the probability of querying an edge $\{u, v\}$?
  - If + edge: $1 - (1-p)^{\text{\# of (+,+,-) triangles including } \{u,v\}}$
  - If - edge: Similar to above but only triangles in which $OPT$ doesn't make a mistake on other pivot edge

# Interesting properties of RandomQueryPivot

- What is the probability of querying an edge $\{u, v\}$?
  - If + edge: $1-(1-p)^{\text{\# of (+,+,-) triangles including } \{u,v\}}$
  - If - edge: Similar to above but only triangles in which $OPT$ doesn't make a mistake on other pivot edge
  - Now consider only edges on which $OPT$ makes a mistake. Does it matter whether $u$ or $v$ is the pivot?

# Interesting properties of RandomQueryPivot

- What is the probability of querying an edge $\{u, v\}$?
  - If + edge: $1 - (1 - p)^{\#\text{ of (+,+,-) triangles including } \{u,v\}}$
  - If - edge: Similar to above but only triangles in which $OPT$ doesn't make a mistake on other pivot edge
  - Now consider only edges on which $OPT$ makes a mistake. Does it matter whether $u$ or $v$ is the pivot?
    - No! Let $T_{uv}$ be set of (+, +, -) triangles including $\{u, v\}$, and let $T^1_{uv}$ be subset in which $\{u, v\}$ is only edge on which $OPT$ makes a mistake. Then
      - If + edge: $1 - (1 - p)^{|T_{uv}|}$
      - If - edge: $1 - (1 - p)^{|T^1_{uv}|}$



# Query Complexity for RandomQueryPivot

- We will charge queries to edges on which $OPT$ makes a mistake:
  - If we query an edge on which $OPT$ makes a mistake, charge that edge.
  - Otherwise, charge to an edge in the triangle on which $OPT$ makes a mistake.

# Query Complexity for RandomQueryPivot

- We will charge queries to edges on which $OPT$ makes a mistake:
  - If we query an edge on which $OPT$ makes a mistake, charge that edge.
  - Otherwise, charge to an edge in the triangle on which $OPT$ makes a mistake.
  - Claim: we only make the second kind of charge in triangles in which $OPT$ makes 1 mistake ($T^1_{uv}$).

# Query Complexity for RandomQueryPivot

$$E[queries_t] = \sum_{\{u,v\} \in E_t} c^*_{uv} \sum_{w \in V_t} \frac{1}{|V_t|} E[Q_{uv} | A_w] \leq \sum_{\{u,v\} \in E_t} \frac{c^*_{uv}}{|V_t|} [2(1 + p|T^1_{uv}|) + 2p|T^1_{uv}|]$$

$$E[C^t_{OPT}] = \sum_{\{u,v\} \in E_t} c^*_{uv} \sum_{w \in V_t} \frac{1}{|V_t|} \Pr[D_{uv} | A_w] \geq \sum_{\{u,v\} \in E_t} \frac{c^*_{uv}}{|V_t|} (2 + |T^1_{uv}|)$$

$$\frac{E[queries_t]}{C^t_{OPT}} \leq \frac{2 + 4p|T^1_{uv}|}{2 + |T^1_{uv}|} \leq \max(1, 4p)$$

# Outline of Talk

1. Introduction to the Problem
2. Simple 3-approximation algorithm
3. Pairwise query oracle
4. **<u>NP-completeness proof</u>**

# NP-hardness proof

- Reduction from 3-SAT (Komusiewicz 2011): suppose we have $m$ clauses and $n$ variables.

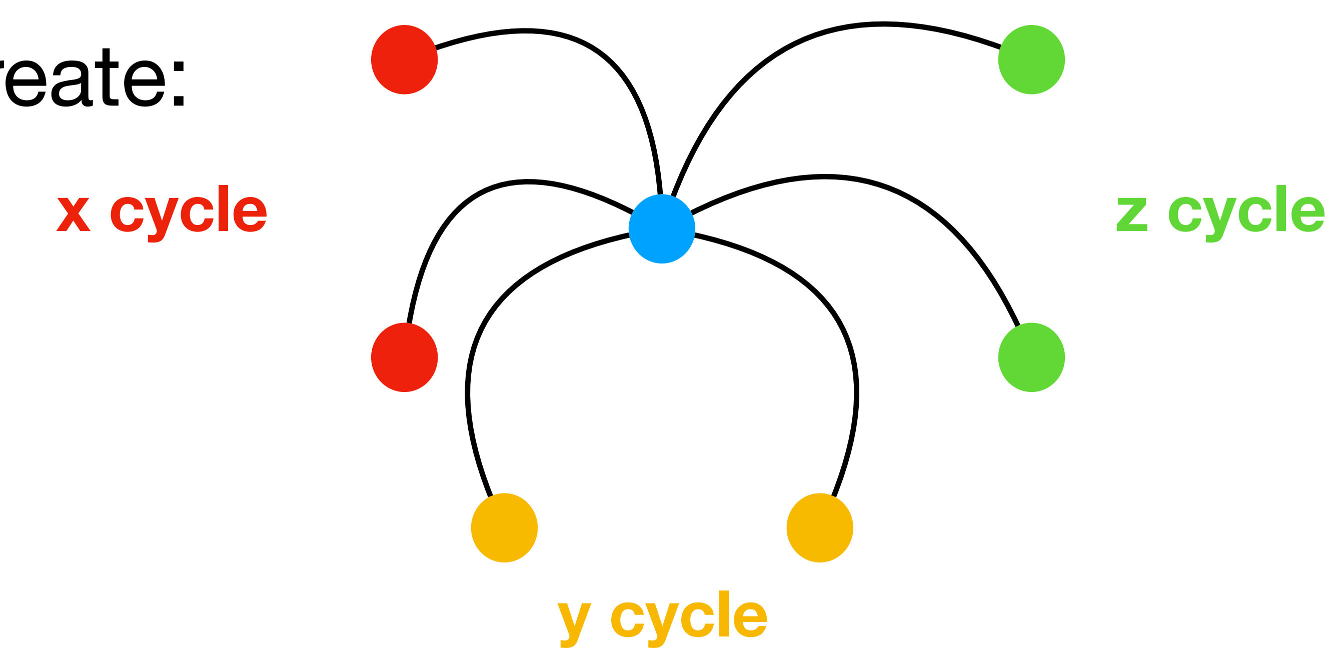
- For each variable $x$, create a cycle of + edges of size $4m_x$, where $m_x$ is the number of clauses including $x$. (all other edges among the vertices in the cycle are - edges).

# NP-hardness proof

- Reduction from 3-SAT: suppose we have $m$ clauses and $n$ variables.

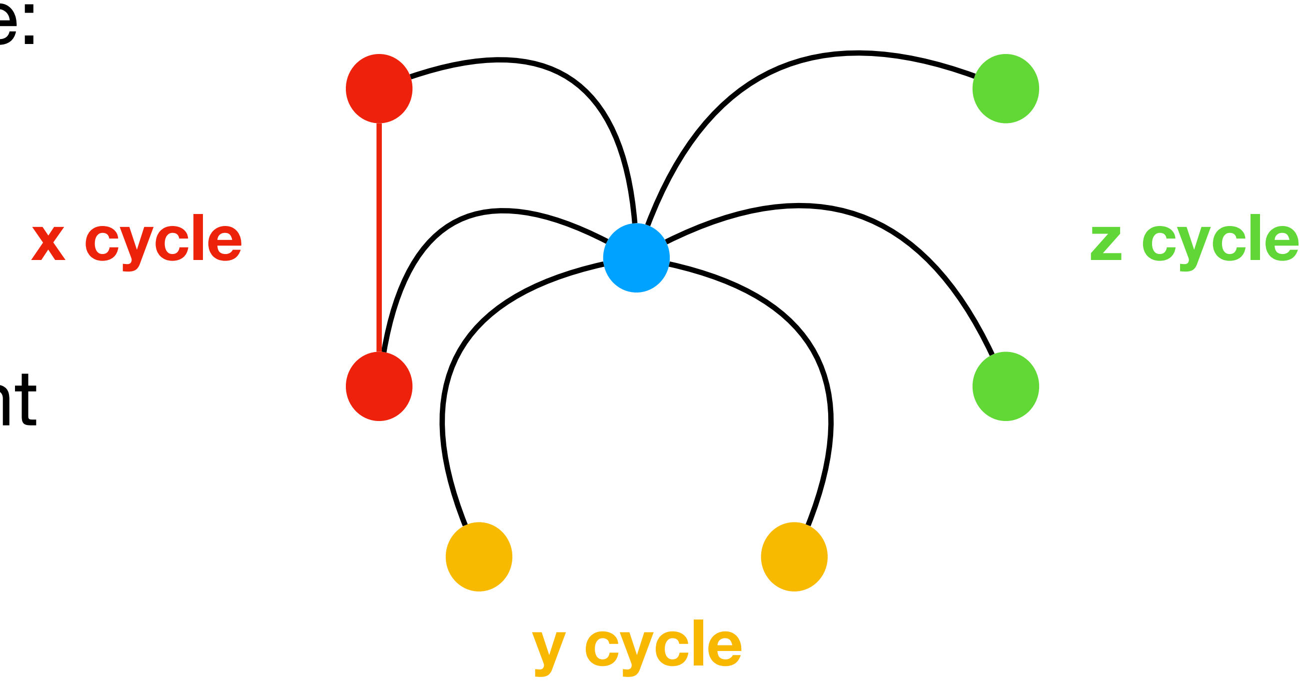
- For each variable $x$, create a cycle of + edges of size $4m_x$, where $m_x$ is the number of clauses including $x$. (all other edges among the vertices in the cycle are - edges).

- For each clause (x,y,z), create:



# NP-hardness proof

- Reduction from 3-SAT: suppose we have $m$ clauses and $n$ variables.
- For each variable $x$, create a cycle of + edges of size $4m_x$, where $m_x$ is the number of clauses including $x$. (all other edges among the vertices in the cycle are - edges).
- For each clause (x,y,z), create:



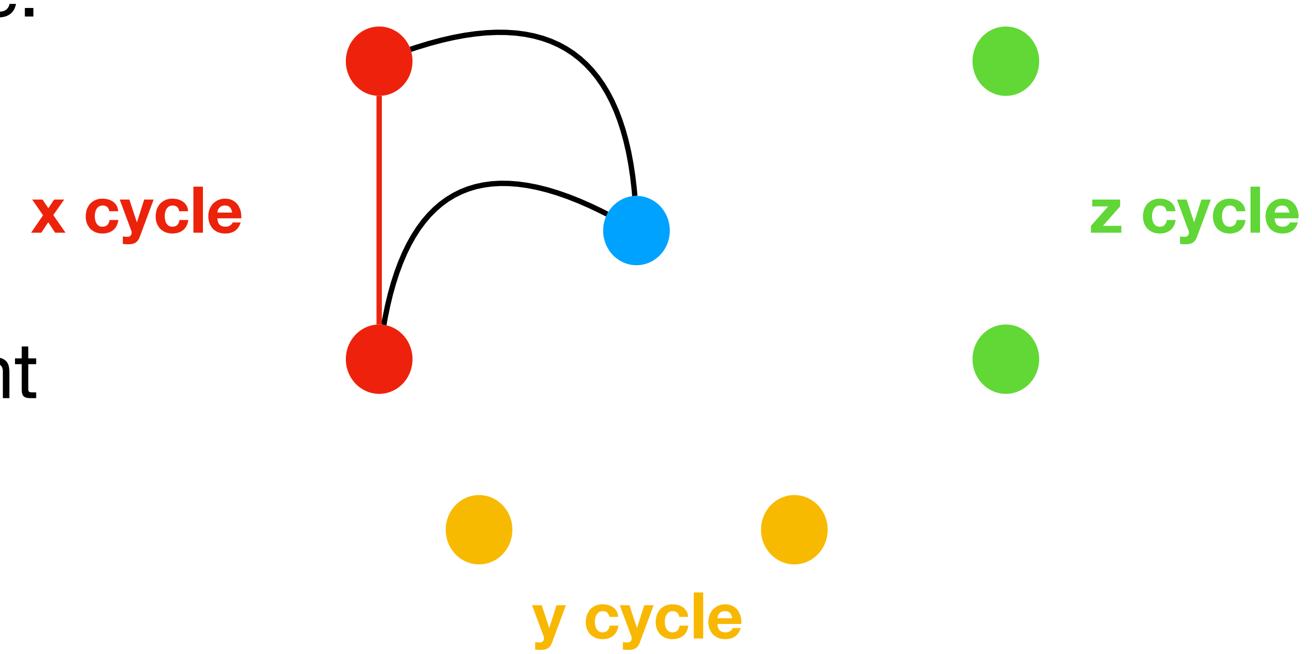

If there’s a satisfying assignment with x = True, y = z = False:

# NP-hardness proof

- Reduction from 3-SAT: suppose we have $m$ clauses and $n$ variables.

- For each variable $x$, create a cycle of + edges of size $4m_x$, where $m_x$ is the number of clauses including $x$. (all other edges among the vertices in the cycle are - edges).

- For each clause (x,y,z), create:



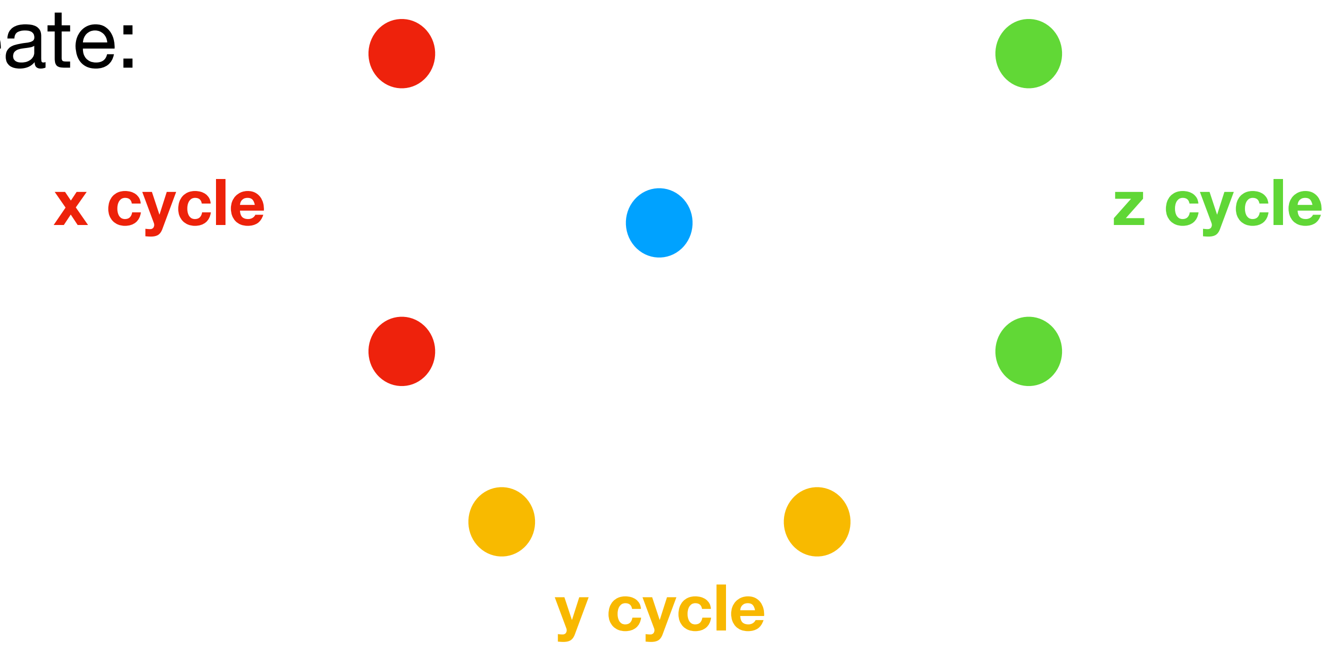

If there's a satisfying assignment with x = True, y = z = False:

# NP-hardness proof

- Reduction from 3-SAT: suppose we have $m$ clauses and $n$ variables.

- For each variable $x$, create a cycle of + edges of size $4m_x$, where $m_x$ is the number of clauses including $x$. (all other edges among the vertices in the cycle are - edges).

- For each clause (x,y,z), create:




If no satisfying assignment:

# NP-hardness proof

- There exists an optimal solution in which we make mistakes only on + edges. (see Lemma on next slide)
- In variable cycles, we'll make $\frac{1}{2}4(3m) = 6m$ mistakes
- When there's a satisfying solution to 3-SAT instance, we can make $6m + 4m = 10m$ mistakes.
- When there's no satisfying solution, we must make more mistakes since there will always be some unsatisfiable clause.
- Thus, deciding whether 3-SAT instance is satisfiable is equivalent to deciding whether optimal number of mistakes is $10m$.

# NP-hardness proof

- Lemma: If for every $u, v$ connected by a - edge, they have a common +-neighborhood of size at most 1, then there is an optimal clustering that puts each such $u, v$ in different clusters

  - Proof: Suppose not. We will take an optimal clustering and modify it to satisfy this property. Let $K$ be the cluster containing $u, v$. Let $X$ be their shared +-neighborhood. Let $K_u = |K \cap N(u)|$ and $K_v = |K \cap N(v)|$. WLOG assume $K_v \geq K_u$. If $K_u > \frac{K-1}{2}$, then $|K| \geq K_u + K_v - |X| + 2 > K - 1 - |X| + 2 \geq = K - 1 - 1 + 2 = K$ Contradiction. So $K_u \leq \frac{K-1}{2}$. Consider putting $u$ in its own cluster. The additional cost would be $K_u - (|K| - K_u - 1) = 2K_u - |K| + 1 \leq |K| - 1 - |K| + 1 = 0$