Stabilizing Weighted Graphs

Zhuan Khye Koh    Laura Sanità

Combinatorics and Optimization
University of Waterloo, Canada

July 3, 2018
Let $G = (V, E)$ be a graph with edge-weights $w \in \mathbb{R}^E \geq 0$.

**Def.** A vector $x \in \mathbb{R}^E$ is a fractional matching if it is a feasible solution to

$$\nu_f(G) := \max \left\{ w^\top x : x(\delta(v)) \leq 1 \forall v \in V, x \geq 0 \right\}.$$ 

**Def.** A vector $y \in \mathbb{R}^V$ is a fractional $w$-vertex cover if it is a feasible solution to

$$\tau_f(G) := \min \left\{ 1^\top y : y_{uv} + y_{vu} \geq w_{uv} \forall uv \in E, y \geq 0 \right\}.$$ 

• Denote $\nu(G)$ as the value of a maximum-weight matching in $G$.

• By LP duality, $\nu(G) \leq \nu_f(G) = \tau_f(G)$. 

Matchings and $w$-vertex covers
Matchings and $w$-vertex covers

- Let $G = (V, E)$ be a graph with edge-weights $w \in \mathbb{R}^E_{\geq 0}$. 
Matchings and $w$-vertex covers

• Let $G = (V, E)$ be a graph with edge-weights $w \in \mathbb{R}^E_{\geq 0}$.

**Def.** A vector $x \in \mathbb{R}^E$ is a fractional matching if it is a feasible solution to

$$
\nu_f(G) := \max \left\{ w^T x : x(\delta(v)) \leq 1 \ \forall v \in V, x \geq 0 \right\}.
$$

• Denote $\nu(G)$ as the value of a maximum-weight matching in $G$.

• By LP duality, $\nu(G) \leq \nu_f(G) = \tau_f(G)$.
Matchings and $w$-vertex covers

• Let $G = (V,E)$ be a graph with edge-weights $w \in \mathbb{R}^E_{\geq 0}$.

**Def.** A vector $x \in \mathbb{R}^E$ is a fractional matching if it is a feasible solution to

$$\nu_f(G) := \max \left\{ w^\top x : x(\delta(v)) \leq 1 \ \forall v \in V, \ x \geq 0 \right\}.$$

**Def.** A vector $y \in \mathbb{R}^V$ is a fractional $w$-vertex cover if it is a feasible solution to

$$\tau_f(G) := \min \left\{ 1^\top y : y_u + y_v \geq w_{uv} \ \forall uv \in E, \ y \geq 0 \right\}.$$
**Matchings and \( w \)-vertex covers**

- Let \( G = (V, E) \) be a graph with edge-weights \( w \in \mathbb{R}^E_{\geq 0} \).

**Def.** A vector \( x \in \mathbb{R}^E \) is a **fractional matching** if it is a feasible solution to

\[
\nu_f(G) := \max \left\{ w^T x : x(\delta(v)) \leq 1 \ \forall v \in V, x \geq 0 \right\}.
\]

**Def.** A vector \( y \in \mathbb{R}^V \) is a **fractional \( w \)-vertex cover** if it is a feasible solution to

\[
\tau_f(G) := \min \left\{ 1^T y : y_u + y_v \geq w_{uv} \ \forall uv \in E, y \geq 0 \right\}.
\]

- Denote \( \nu(G) \) as the value of a maximum-weight matching in \( G \).
Matchings and $w$-vertex covers

• Let $G = (V, E)$ be a graph with edge-weights $w \in \mathbb{R}^E_{\geq 0}$.

**Def.** A vector $x \in \mathbb{R}^E$ is a **fractional matching** if it is a feasible solution to
\[
\nu_f(G) := \max \left\{ w^\top x : x(\delta(v)) \leq 1 \ \forall v \in V, x \geq 0 \right\}.
\]

**Def.** A vector $y \in \mathbb{R}^V$ is a **fractional $w$-vertex cover** if it is a feasible solution to
\[
\tau_f(G) := \min \left\{ \mathbb{1}^\top y : y_u + y_v \geq w_{uv} \ \forall uv \in E, y \geq 0 \right\}.
\]

• Denote $\nu(G)$ as the value of a maximum-weight matching in $G$.

• By LP duality,
\[
\nu(G) \leq \nu_f(G) = \tau_f(G).
\]
Stable graphs

There are graphs where $\nu(G) < \nu(f(G))$. 1

Def. A graph $G$ is stable if $\nu(G) = \nu(f(G))$. 1

$x = 1$
Stable graphs

- There are graphs where $\nu(G) < \nu_f(G)$. 
Stable graphs

- There are graphs where $\nu(G) < \nu_f(G)$.

\[x_e = 1\]

\[x_e = \frac{1}{2}\]
Stable graphs

- There are graphs where $\nu(G) < \nu_f(G)$.

\[ \nu(G) = 1 \quad \text{and} \quad \nu_f(G) = 1.5 \]

**Def.** A graph $G$ is **stable** if $\nu(G) = \nu_f(G)$.
Stable graphs

• There are graphs where \( \nu(G) < \nu_f(G) \).

\[
\begin{align*}
\nu(G) &= 1 \\
\nu_f(G) &= 1.5
\end{align*}
\]

**Def.** A graph \( G \) is stable if \( \nu(G) = \nu_f(G) \).

\[
\begin{align*}
\nu(G) &= 1 \\
\nu_f(G) &= 1.5
\end{align*}
\]
Stable graphs

• There are graphs where $\nu(G) < \nu_f(G)$.

\[\nu(G) = 1\]
\[\nu_f(G) = 1.5\]

**Def.** A graph $G$ is **stable** if $\nu(G) = \nu_f(G)$.
Stabilizers

Def. An edge-stabilizer is a subset $F \subset E$ such that $G \setminus F$ is stable.

Def. A vertex-stabilizer is a subset $S \subseteq V$ such that $G \setminus S$ is stable.
Stabilizers

**Def.** An edge-stabilizer is a subset $F \subset E$ such that $G \setminus F$ is stable.
**Stabilizers**

**Def.** An edge-stabilizer is a subset $F \subset E$ such that $G \setminus F$ is stable.

![Diagram of edge-stabilizer]

**Def.** A vertex-stabilizer is a subset $S \subseteq V$ such that $G \setminus S$ is stable.

![Diagram of vertex-stabilizer]
**Stabilizers**

**Def.** An edge-stabilizer is a subset $F \subseteq E$ such that $G \setminus F$ is stable.

**Def.** A vertex-stabilizer is a subset $S \subseteq V$ such that $G \setminus S$ is stable.
**Stabilizers**

**Def.** An edge-stabilizer is a subset $F \subseteq E$ such that $G \setminus F$ is stable.

![Diagram of edge-stabilizer](image1)

**Def.** A vertex-stabilizer is a subset $S \subseteq V$ such that $G \setminus S$ is stable.

![Diagram of vertex-stabilizer](image2)
Finding small stabilizers

This gives rise to the following two optimization problems:

Minimum Vertex-Stabilizer
Find a vertex-stabilizer of minimum cardinality.

Minimum Edge-Stabilizer
Find an edge-stabilizer of minimum cardinality.

Why are stable graphs interesting?

Motivated by network bargaining games and cooperative matching games.
Finding small stabilizers

- This gives rise to the following two optimization problems:
Finding small stabilizers

- This gives rise to the following two optimization problems:

  **Minimum Vertex-Stabilizer**
  Find a vertex-stabilizer of minimum cardinality.

  **Minimum Edge-Stabilizer**
  Find an edge-stabilizer of minimum cardinality.
Finding small stabilizers

- This gives rise to the following two optimization problems:

  **Minimum Vertex-Stabilizer**
  Find a *vertex-stabilizer* of minimum cardinality.

  **Minimum Edge-Stabilizer**
  Find an *edge-stabilizer* of minimum cardinality.

- Why are stable graphs interesting?
Finding small stabilizers

- This gives rise to the following two optimization problems:
  
  **Minimum Vertex-Stabilizer**
  Find a *vertex-stabilizer* of minimum cardinality.

  **Minimum Edge-Stabilizer**
  Find an *edge-stabilizer* of minimum cardinality.

- Why are stable graphs interesting?
  - Motivated by network bargaining games and cooperative matching games.
Network bargaining games

• Given an edge-weighted graph $G = (V, E)$

  ▶ Every vertex represents a player.
  ▶ Every edge $e$ represents a deal of value $w_e$.

• Every player can make a deal with at most 1 neighbour. → matching $M$

  ▶ When a deal is made, players split the value. → allocation $y \in \mathbb{R}^V_\geq 0$:

    $y_u + y_v = w_{uv} \quad \forall uv \in M$

    $y_u = 0$ if $u$ is $M$-exposed.

• An outcome is given by $(M, y)$.

• An outcome is stable if $y_u + y_v \geq w_{uv}$ for all $uv \in E$.

• A stable outcome is balanced if the deal values are "fairly" split.

• A stable outcome exists $\iff$ A balanced outcome exists $\iff G$ is stable
Network bargaining games

• [Kleinberg and Tardos '08] Given an edge-weighted graph $G = (V, E)$
Network bargaining games

- [Kleinberg and Tardos '08] Given an edge-weighted graph $G = (V, E)$
  - Every vertex represents a player.
  - Every edge $e$ represents a deal of value $w_e$. 

 verdict

Network bargaining games

- [Kleinberg and Tardos '08] Given an edge-weighted graph $G = (V, E)$
  - Every vertex represents a player.
  - Every edge $e$ represents a deal of value $w_e$.
- Every player can make a deal with at most 1 neighbour.
  $\rightarrow$ matching $M$
Network bargaining games

- [Kleinberg and Tardos '08] Given an edge-weighted graph $G = (V, E)$
  - Every vertex represents a player.
  - Every edge $e$ represents a deal of value $w_e$.
- Every player can make a deal with at most 1 neighbour.
  \[ \rightarrow \text{matching } M \]
- When a deal is made, players split the value.
  \[ \rightarrow \text{allocation } y \in \mathbb{R}_{\geq 0}^V : \]
  \[ y_u + y_v = w_{uv} \quad \forall uv \in M \]
  \[ y_u = 0 \text{ if } u \text{ is } M\text{-exposed}. \]
Network bargaining games

- [Kleinberg and Tardos '08] Given an edge-weighted graph \( G = (V, E) \)
  - Every vertex represents a player.
  - Every edge \( e \) represents a deal of value \( w_e \).
- Every player can make a deal with at most 1 neighbour.
  \[ \rightarrow \text{matching } M \]
- When a deal is made, players split the value.
  \[ \rightarrow \text{allocation } y \in \mathbb{R}_\geq 0^V: \]
  \[ y_u + y_v = w_{uv} \quad \forall uv \in M \]
  \[ y_u = 0 \text{ if } u \text{ is } M\text{-exposed.} \]
- An outcome is given by \((M, y)\).
Network bargaining games

- [Kleinberg and Tardos '08] Given an edge-weighted graph $G = (V, E)$
  - Every vertex represents a player.
  - Every edge $e$ represents a deal of value $w_e$.
- Every player can make a deal with at most 1 neighbour.
  → matching $M$
- When a deal is made, players split the value.
  → allocation $y \in \mathbb{R}^V_{\geq 0}$:
    $y_u + y_v = w_{uv}$ \(\forall uv \in M\)
    $y_u = 0$ if $u$ is $M$-exposed.
- An outcome is given by $(M, y)$.
- An outcome is stable if $y_u + y_v \geq w_{uv}$ for all $uv \in E$. 
Network bargaining games

- [Kleinberg and Tardos '08] Given an edge-weighted graph $G = (V, E)$
  - Every vertex represents a player.
  - Every edge $e$ represents a deal of value $w_e$.
- Every player can make a deal with at most 1 neighbour.
  - $\rightarrow$ matching $M$

- When a deal is made, players split the value.
  - $\rightarrow$ allocation $y \in \mathbb{R}^V_{\geq 0}$:
    - $y_u + y_v = w_{uv} \quad \forall uv \in M$
    - $y_u = 0$ if $u$ is $M$-exposed.

- An outcome is given by $(M, y)$.
- An outcome is stable if $y_u + y_v \geq w_{uv}$ for all $uv \in E$.
- A stable outcome is balanced if the deal values are “fairly” split.
Network bargaining games

• [Kleinberg and Tardos '08] Given an edge-weighted graph $G = (V, E)$
  ▶ Every vertex represents a player.
  ▶ Every edge $e$ represents a deal of value $w_e$.
• Every player can make a deal with at most 1 neighbour.
  → matching $M$

• When a deal is made, players split the value.
  → allocation $y \in \mathbb{R}^V_{\geq 0}$:
    $y_u + y_v = w_{uv}$ $\forall uv \in M$
    $y_u = 0$ if $u$ is $M$-exposed.

• An outcome is given by $(M, y)$.
• An outcome is stable if $y_u + y_v \geq w_{uv}$ for all $uv \in E$.
• A stable outcome is balanced if the deal values are “fairly” split.

A stable outcome exists $\iff$ A balanced outcome exists $\iff$ $G$ is stable
Cooperative matching games

Let $G = (V, E)$ be an edge-weighted graph.

Goal: Allocate the value $\nu(G)$ among the vertices such that

- No subset $S \subseteq V$ is incentivized to form a coalition to deviate
  $\sum_{v \in S} y_v \geq \nu(G[S])$ \quad $\forall S \subseteq V$

- Such an allocation $y$ is called stable.

[Deng et al. '99] proved that a stable allocation exists $\iff G$ is stable.

Can we stabilize unstable games through minimal changes in the underlying network?

e.g. by blocking some players

Vertex-stabilizer

by blocking some deals

Edge-stabilizer
Cooperative matching games

- [Shapley and Shubik ’71] Let $G = (V, E)$ be an edge-weighted graph.

- [Deng et al. ’99] proved that a stable allocation exists $\iff G$ is stable. Can we stabilize unstable games through minimal changes in the underlying network? e.g. by blocking some players (Vertex-stabilizer) or blocking some deals (Edge-stabilizer).
Cooperative matching games

• [Shapley and Shubik '71] Let $G = (V, E)$ be an edge-weighted graph.

**Goal:** Allocate the value $\nu(G)$ among the vertices such that

- No subset $S \subseteq V$ is incentivized to form a coalition to deviate

\[
\sum_{v \in S} y_v \geq \nu(G[S]) \quad \forall S \subseteq V
\]
Cooperative matching games

• [Shapley and Shubik ’71] Let $G = (V, E)$ be an edge-weighted graph.

**Goal:** Allocate the value $\nu(G)$ among the vertices such that
  1. No subset $S \subseteq V$ is incentivized to form a coalition to deviate
     $$\sum_{v \in S} y_v \geq \nu(G[S]) \quad \forall S \subseteq V$$
  2. Such an allocation $y$ is called stable.

• [Deng et al. ’99] proved that $G$ is stable if and only if a stable allocation exists.
Cooperative matching games

• [Shapley and Shubik '71] Let \( G = (V, E) \) be an edge-weighted graph. 

**Goal:** Allocate the value \( \nu(G) \) among the vertices such that

- No subset \( S \subseteq V \) is incentivized to form a coalition to deviate

\[
\sum_{v \in S} y_v \geq \nu(G[S]) \quad \forall S \subseteq V
\]

- Such an allocation \( y \) is called **stable**.

• [Deng et al. '99] proved that a stable allocation exists \( \iff \) \( G \) is stable
Cooperative matching games

- [Shapley and Shubik ’71] Let $G = (V, E)$ be an edge-weighted graph.

**Goal:** Allocate the value $\nu(G)$ among the vertices such that
- No subset $S \subseteq V$ is incentivized to form a coalition to deviate

$$\sum_{v \in S} y_v \geq \nu(G[S]) \quad \forall S \subseteq V$$

- Such an allocation $y$ is called **stable**.

- [Deng et al. ’99] proved that a stable allocation exists $\iff G$ is stable.

*Can we stabilize unstable games through minimal changes in the underlying network?*
Cooperative matching games

• [Shapley and Shubik ’71] Let $G = (V, E)$ be an edge-weighted graph.

**Goal:** Allocate the value $\nu(G)$ among the vertices such that

- No subset $S \subseteq V$ is incentivized to form a coalition to deviate

$$\sum_{v \in S} y_v \geq \nu(G[S]) \quad \forall S \subseteq V$$

- Such an allocation $y$ is called stable.

• [Deng et al. ’99] proved that a stable allocation exists $\iff G$ is stable

*Can we stabilize unstable games through minimal changes in the underlying network?*

*E.g. by blocking some players*  
**Vertex-stabilizer**

*by blocking some deals*  
**Edge-stabilizer**
State of the art

Unweighted Graphs

• [Bock et al. '15] Finding a minimum edge-stabilizer is hard to approximate within a factor of $(2 - \varepsilon)$ for any $\varepsilon > 0$ assuming UGC.

• They gave an $O(\omega)$-approximation algorithm, where $\omega$ is the sparsity of the graph.

• [Ahmadian et al. '16, Ito et al. '16] Finding a minimum vertex-stabilizer is polynomial time solvable.

• Stabilizing a graph via different operations:
  ▶ [Ito et al. '16] Adding vertices/edges.
  ▶ [Chandrasekaran et al. '16] Fractionally increasing edge weights.

• [Ahmadian et al. '16] Vertex-stabilizer with costs.

• Other variants [Mishra et al. '11, Biró et al. '12, Kőnemann et al. '15].
Unweighted Graphs

- Finding a minimum edge-stabilizer is hard to approximate within a factor of $2 - \varepsilon$ for any $\varepsilon > 0$ assuming UGC.
- An $O(\omega)$-approximation algorithm was given, where $\omega$ is the sparsity of the graph.

- Finding a minimum vertex-stabilizer is polynomial time solvable.

- Stabilizing a graph via different operations:
  - Adding vertices/edges.
  - Fractionally increasing edge weights.
  - Vertex-stabilizer with costs.

- Other variants: [Mishra et al. '11, Biró et al. '12, Könenmann et al. '15].
State of the art

Unweighted Graphs

- [Bock et al. '15] Finding a minimum edge-stabilizer is hard to approximate within a factor of $(2 - \varepsilon)$ for any $\varepsilon > 0$ assuming UGC.
State of the art

Unweighted Graphs

- [Bock et al. '15] Finding a minimum edge-stabilizer is hard to approximate within a factor of \((2 - \varepsilon)\) for any \(\varepsilon > 0\) assuming UGC.
- They gave an \(O(\omega)\)-approximation algorithm, where \(\omega\) is the sparsity of the graph.

- [Ahmadian et al. '16, Ito et al. '16] Finding a minimum vertex-stabilizer is polynomial time solvable.

- Stabilizing a graph via different operations:
  - [Ito et al. '16] Adding vertices/edges.
  - [Chandrasekaran et al. '16] Fractionally increasing edge weights.
  - [Ahmadian et al '16] Vertex-stabilizer with costs.

- Other variants [Mishra et al. '11, Biró et al. '12, Kőnemann et al. '15].
State of the art

Unweighted Graphs

• [Bock et al. ’15] Finding a minimum edge-stabilizer is hard to approximate within a factor of \((2 - \varepsilon)\) for any \(\varepsilon > 0\) assuming UGC.

• They gave an \(O(\omega)\)-approximation algorithm, where \(\omega\) is the sparsity of the graph.

State of the art

Unweighted Graphs

- [Bock et al. '15] Finding a minimum edge-stabilizer is hard to approximate within a factor of $(2 - \varepsilon)$ for any $\varepsilon > 0$ assuming UGC.

- They gave an $O(\omega)$-approximation algorithm, where $\omega$ is the sparsity of the graph.

- [Ahmadian et al. '16, Ito et al. '16] Finding a minimum vertex-stabilizer is polynomial time solvable.

- Stabilizing a graph via different operations:
  - [Ito et al. '16] Adding vertices/edges.
  - [Chandrasekaran et al. '16] Fractionally increasing edge weights.
State of the art

Unweighted Graphs

- [Bock et al. ’15] Finding a minimum edge-stabilizer is hard to approximate within a factor of \((2 - \varepsilon)\) for any \(\varepsilon > 0\) assuming UGC.

- They gave an \(O(\omega)\)-approximation algorithm, where \(\omega\) is the sparsity of the graph.


- Stabilizing a graph via different operations:
  - [Ito et al. ’16] Adding vertices/edges.
  - [Chandrasekaran et al. ’16] Fractionally increasing edge weights.

State of the art

Unweighted Graphs

• [Bock et al. ’15] Finding a minimum edge-stabilizer is hard to approximate within a factor of \((2 - \varepsilon)\) for any \(\varepsilon > 0\) assuming UGC.

• They gave an \(O(\omega)\)-approximation algorithm, where \(\omega\) is the sparsity of the graph.


• Stabilizing a graph via different operations:
  ▶ [Chandrasekaran et al. ’16] Fractionally increasing edge weights.

• [Ahmadian et al ’16] Vertex-stabilizer with costs.

• Other variants [Mishra et al. ’11, Biró et al. ’12, Könemann et al. ’15].
Unweighted vs. weighted graphs

• On unweighted graphs,
  • For any minimum edge-stabilizer $F$, $\nu(G \setminus F) = \nu(G)$.
  • For any minimum vertex-stabilizer $S$, $\nu(G \setminus S) = \nu(G)$.

• This property does not hold on weighted graphs.
Unweighted vs. weighted graphs

- On unweighted graphs,
  - For any minimum edge-stabilizer $F$, $\nu(G \setminus F) = \nu(G)$.
  - For any minimum vertex-stabilizer $S$, $\nu(G \setminus S) = \nu(G)$.

This property does not hold on weighted graphs.
Unweighted vs. weighted graphs

• On unweighted graphs,
  ▶ For any minimum edge-stabilizer \( F \), \( \nu(G \setminus F) = \nu(G) \).
  ▶ For any minimum vertex-stabilizer \( S \), \( \nu(G \setminus S) = \nu(G) \).

• This property does not hold on weighted graphs.
Unweighted vs. weighted graphs

- On unweighted graphs,
  - For any minimum edge-stabilizer $F$, $\nu(G \setminus F) = \nu(G)$.
  - For any minimum vertex-stabilizer $S$, $\nu(G \setminus S) = \nu(G)$.

- This property does not hold on weighted graphs.

\begin{align*}
\nu(G) &= 5 \\
\nu_f(G) &= 6
\end{align*}
Main results

Thm 1: There exists a polynomial time algorithm that computes a minimum vertex-stabilizer $S$ for a weighted graph $G$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$.

Thm 2: Deciding whether a graph $G$ has a vertex-stabilizer $S$ where $\nu(G \setminus S) = \nu(G)$ is NP-complete.

Thm 3: There is no constant factor approximation for the minimum edge-stabilizer problem unless $P = NP$.

Thm 4: There exists an efficient $O(\Delta)$-approximation algorithm for the minimum edge-stabilizer problem.
Main results

**Thm 1:** There exists a polynomial time algorithm that computes a minimum vertex-stabilizer $S$ for a weighted graph $G$. Moreover,

$$\nu(G \setminus S) \geq \frac{2}{3} \nu(G).$$

**Thm 2:** Deciding whether a graph $G$ has a vertex-stabilizer $S$ where $\nu(G \setminus S) = \nu(G)$ is $\textbf{NP}$-complete.
Main results

**Thm 1:** There exists a polynomial time algorithm that computes a minimum vertex-stabilizer $S$ for a weighted graph $G$. Moreover,

$$\nu(G \setminus S) \geq \frac{2}{3} \nu(G).$$

**Thm 2:** Deciding whether a graph $G$ has a vertex-stabilizer $S$ where $\nu(G \setminus S) = \nu(G)$ is NP-complete.

**Thm 3:** There is no constant factor approximation for the minimum edge-stabilizer problem unless $P = NP$.

**Thm 4:** There exists an efficient $O(\Delta)$-approximation algorithm for the minimum edge-stabilizer problem.
Thm [Balinski '70]:
A fractional matching $\hat{x}$ in $G$ is basic if and only if
$1 \leq \hat{x}_e \leq 1$ for every edge $e$; and
The edges $e$ with $\hat{x}_e = \frac{1}{2}$ induce vertex-disjoint odd cycles in $G$.

$\mathcal{C}(\hat{x}) := \{C_1,...,C_q\}$ as the set of odd cycles induced by $\hat{x}_e = \frac{1}{2}$
$\mathcal{M}(\hat{x}) := \{e \in E : \hat{x}_e = 1\}$.

Def. $\gamma(G) := \min \hat{x} \in X | \mathcal{C}(\hat{x})|$ where $X$ is the set of basic maximum-weight fractional matchings in $G$.

$G$ is stable if and only if $\gamma(G) = 0$.

Let $y$ be a minimum fractional $w$-vertex cover in $G$.
An edge $uv$ is tight if $y_u + y_v = w_{uv}$.
A path is tight if all its edges are tight.
Thm [Balinski ’70]: A fractional matching \( \hat{x} \) in \( G \) is basic if and only if

1. \( \hat{x}_e \in \{0, \frac{1}{2}, 1\} \) for every edge \( e \); and
2. The edges \( e \) with \( \hat{x}_e = \frac{1}{2} \) induce vertex-disjoint odd cycles in \( G \).
**Preliminaries**

**Thm [Balinski '70]:** A fractional matching $\hat{x}$ in $G$ is basic if and only if

1. $\hat{x}_e \in \{0, \frac{1}{2}, 1\}$ for every edge $e$; and
2. The edges $e$ with $\hat{x}_e = \frac{1}{2}$ induce vertex-disjoint odd cycles in $G$.

- Given a basic fractional matching $\hat{x}$ in $G$, denote
Thm [Balinski ’70]: A fractional matching $\hat{x}$ in $G$ is basic if and only if
1. $\hat{x}_e \in \{0, \frac{1}{2}, 1\}$ for every edge $e$; and
2. The edges $e$ with $\hat{x}_e = \frac{1}{2}$ induce vertex-disjoint odd cycles in $G$.

- Given a basic fractional matching $\hat{x}$ in $G$, denote
  - $\mathcal{C}(\hat{x}) := \{C_1, \ldots, C_q\}$ as the set of odd cycles induced by $\hat{x}_e = \frac{1}{2}$
**Thm [Balinski '70]:** A fractional matching \( \hat{x} \) in \( G \) is basic if and only if

1. \( \hat{x}_e \in \{0, \frac{1}{2}, 1\} \) for every edge \( e \); and
2. The edges \( e \) with \( \hat{x}_e = \frac{1}{2} \) induce vertex-disjoint odd cycles in \( G \).

- Given a basic fractional matching \( \hat{x} \) in \( G \), denote
  - \( C(\hat{x}) := \{C_1, \ldots, C_q\} \) as the set of odd cycles induced by \( \hat{x}_e = \frac{1}{2} \)
  - \( M(\hat{x}) := \{e \in E : \hat{x}_e = 1\} \).
Preliminaries

Thm [Balinski ’70]: A fractional matching $\hat{x}$ in $G$ is basic if and only if
1. $\hat{x}_e \in \{0, \frac{1}{2}, 1\}$ for every edge $e$; and
2. The edges $e$ with $\hat{x}_e = \frac{1}{2}$ induce vertex-disjoint odd cycles in $G$.

• Given a basic fractional matching $\hat{x}$ in $G$, denote
  ▶ $\mathcal{C}(\hat{x}) := \{C_1, \ldots, C_q\}$ as the set of odd cycles induced by $\hat{x}_e = \frac{1}{2}$
  ▶ $M(\hat{x}) := \{e \in E : \hat{x}_e = 1\}$.

Def.

$$\gamma(G) := \min_{\hat{x} \in \mathcal{X}} |\mathcal{C}(\hat{x})|$$

where $\mathcal{X}$ is the set of basic maximum-weight fractional matchings in $G$. 
Thm [Balinski ’70]: A fractional matching \( \hat{x} \) in \( G \) is \textit{basic} if and only if

1. \( \hat{x}_e \in \{0, \frac{1}{2}, 1\} \) for every edge \( e \); and
2. The edges \( e \) with \( \hat{x}_e = \frac{1}{2} \) induce vertex-disjoint odd cycles in \( G \).

- Given a basic fractional matching \( \hat{x} \) in \( G \), denote
  - \( \mathcal{C}(\hat{x}) := \{C_1, \ldots, C_q\} \) as the set of odd cycles induced by \( \hat{x}_e = \frac{1}{2} \)
  - \( M(\hat{x}) := \{e \in E : \hat{x}_e = 1\} \).

\textbf{Def.}

\[
\gamma(G) := \min_{\hat{x} \in \mathcal{X}} |\mathcal{C}(\hat{x})|,
\]

where \( \mathcal{X} \) is the set of basic maximum-weight fractional matchings in \( G \).

- \( G \) is stable if and only if \( \gamma(G) = 0 \).
**Preliminaries**

**Thm [Balinski ’70]:** A fractional matching \( \hat{x} \) in \( G \) is basic if and only if
1. \( \hat{x}_e \in \{0, \frac{1}{2}, 1\} \) for every edge \( e \); and
2. The edges \( e \) with \( \hat{x}_e = \frac{1}{2} \) induce vertex-disjoint odd cycles in \( G \).

- Given a basic fractional matching \( \hat{x} \) in \( G \), denote
  - \( C(\hat{x}) := \{C_1, \ldots, C_q\} \) as the set of odd cycles induced by \( \hat{x}_e = \frac{1}{2} \)
  - \( M(\hat{x}) := \{e \in E : \hat{x}_e = 1\} \).

**Def.**

\[
\gamma(G) := \min_{\hat{x} \in \mathcal{X}} |C(\hat{x})|
\]

where \( \mathcal{X} \) is the set of basic maximum-weight fractional matchings in \( G \).

- \( G \) is stable if and only if \( \gamma(G) = 0 \).

- Let \( y \) be a minimum fractional \( w \)-vertex cover in \( G \).
Preliminaries

**Thm [Balinski '70]:** A fractional matching \( \hat{x} \) in \( G \) is basic if and only if

1. \( \hat{x}_e \in \{0, \frac{1}{2}, 1\} \) for every edge \( e \); and
2. The edges \( e \) with \( \hat{x}_e = \frac{1}{2} \) induce vertex-disjoint odd cycles in \( G \).

• Given a basic fractional matching \( \hat{x} \) in \( G \), denote
  - \( \mathcal{C}(\hat{x}) := \{C_1, \ldots, C_q\} \) as the set of odd cycles induced by \( \hat{x}_e = \frac{1}{2} \)
  - \( M(\hat{x}) := \{e \in E : \hat{x}_e = 1\} \).

**Def.**

\[
\gamma(G) := \min_{\hat{x} \in \mathcal{X}} |\mathcal{C}(\hat{x})|
\]

where \( \mathcal{X} \) is the set of basic maximum-weight fractional matchings in \( G \).

- \( G \) is stable if and only if \( \gamma(G) = 0 \).

• Let \( y \) be a minimum fractional \( w \)-vertex cover in \( G \).
  - An edge \( uv \) is tight if \( y_u + y_v = w_{uv} \).
Preliminaries

Thm [Balinski '70]: A fractional matching $\hat{x}$ in $G$ is basic if and only if

1. $\hat{x}_e \in \{0, \frac{1}{2}, 1\}$ for every edge $e$; and
2. The edges $e$ with $\hat{x}_e = \frac{1}{2}$ induce vertex-disjoint odd cycles in $G$.

- Given a basic fractional matching $\hat{x}$ in $G$, denote
  - $C(\hat{x}) := \{C_1, \ldots, C_q\}$ as the set of odd cycles induced by $\hat{x}_e = \frac{1}{2}$
  - $M(\hat{x}) := \{e \in E : \hat{x}_e = 1\}$.

Def.

$$\gamma(G) := \min_{\hat{x} \in \mathcal{X}} |C(\hat{x})|$$

where $\mathcal{X}$ is the set of basic maximum-weight fractional matchings in $G$.

- $G$ is stable if and only if $\gamma(G) = 0$.

- Let $y$ be a minimum fractional $w$-vertex cover in $G$.
  - An edge $uv$ is tight if $y_u + y_v = w_{uv}$.
  - A path is tight if all its edges are tight.
Preliminaries

1. By complementing on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$.

2. By alternate rounding on $C \in \mathbb{C}(\hat{x})$ at vertex $v$, we mean...
Preliminaries

- We will use the following 2 operations:

1. By complementing on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$.

2. By alternate rounding on $C \in C(\hat{x})$ at vertex $v$, we mean...
Preliminaries

- We will use the following 2 operations:
  1. By complementing on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$. 

Preliminaries

• We will use the following 2 operations:

1. By complementing on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$. 

![Diagram of graph with alternating path]
Preliminaries

• We will use the following 2 operations:

  1. By **complementing** on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$.

     ![Diagram](image-url)
Preliminaries

• We will use the following 2 operations:

1. By complementing on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$.

2. By alternate rounding on $C \in \mathcal{C}(\hat{x})$ at vertex $v$, we mean
Preliminaries

• We will use the following 2 operations:

1. By complementing on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$.

2. By alternate rounding on $C \in \mathcal{C}(\hat{x})$ at vertex $v$, we mean

![Diagram of an alternating path](image)
• We will use the following 2 operations:

1. **By complementing** on \( F \subseteq E \), we mean replacing \( \hat{x}_e \) by \( \bar{x}_e = 1 - \hat{x}_e \) for all \( e \in F \).

2. **By alternate rounding** on \( C \in \mathcal{C}(\hat{x}) \) at vertex \( v \), we mean

---

**Preliminaries**

---

\( \hat{x}_e \) by \( \bar{x}_e = 1 - \hat{x}_e \) for all \( e \in F \).
Preliminaries

- We will use the following 2 operations:
  
  1. By complementing on $F \subseteq E$, we mean replacing $\hat{x}_e$ by $\bar{x}_e = 1 - \hat{x}_e$ for all $e \in F$.

  
  2. By alternate rounding on $C \in \mathcal{C}(\hat{x})$ at vertex $v$, we mean

```
    C
    ▶
    ✔
    ✔_
```

  
  Def. An alternating path is valid if it
  - starts with an exposed vertex or a matched edge
  - ends with an exposed vertex or a matched edge
Computing vertex-stabilizers

1. Compute a basic maximum-weight fractional matching $\hat{x}$ in $G$ with $\gamma(G)$ odd cycles.

2. Compute a minimum fractional $w$-vertex cover $y$ in $G$.

3. For every odd cycle, delete the vertex with the smallest $y$ value.
Computing vertex-stabilizers

The algorithm:

1. Compute a basic maximum-weight fractional matching \( \hat{x} \) in \( G \) with \( \gamma(G) \) odd cycles.
2. Compute a minimum fractional \( w \)-vertex cover \( y \) in \( G \).
3. For every odd cycle, delete the vertex with the smallest \( y \) value.
Computing vertex-stabilizers

The algorithm:

1. Compute a basic maximum-weight fractional matching \( \hat{x} \) in \( G \) with \( \gamma(G) \) odd cycles.
Computing vertex-stabilizers

The algorithm:

1. Compute a basic maximum-weight fractional matching $\hat{x}$ in $G$ with $\gamma(G)$ odd cycles.
Computing vertex-stabilizers

The algorithm:

1. Compute a basic maximum-weight fractional matching \( \hat{x} \) in \( G \) with \( \gamma(G) \) odd cycles.
2. Compute a minimum fractional \( w \)-vertex cover \( y \) in \( G \).
Computing vertex-stabilizers

The algorithm:

1. Compute a basic maximum-weight fractional matching $\hat{x}$ in $G$ with $\gamma(G)$ odd cycles.
2. Compute a minimum fractional $w$-vertex cover $y$ in $G$.
3. For every odd cycle, delete the vertex with the smallest $y$ value.
Computing vertex-stabilizers

The algorithm:

1. Compute a basic maximum-weight fractional matching \( \hat{x} \) in \( G \) with \( \gamma(G) \) odd cycles.
2. Compute a minimum fractional \( w \)-vertex cover \( y \) in \( G \).
3. For every odd cycle, delete the vertex with the smallest \( y \) value.
Minimize number of odd cycles

Goal: Given a weighted graph $G$, compute a basic maximum-weight fractional matching $\hat{x}$ such that $|C(\hat{x})| = \gamma(G)$.

Thm [Balas '81]: Let $\hat{x}$ be a basic maximum fractional matching in an unweighted graph $G$. If $|C(\hat{x})| > \gamma(G)$, then there exists an $M(\hat{x})$-alternating path $P$ which connects two odd cycles $C_i, C_j \in C(\hat{x})$.

Furthermore, alternate rounding on $C_i, C_j$ and complementing on $P$ produces a basic maximum fractional matching $\bar{x}$ in $G$ such that $C(\bar{x}) \subset C(\hat{x})$. 
Goal: Given a weighted graph $G$, compute a basic maximum-weight fractional matching $\hat{x}$ such that $|C(\hat{x})| = \gamma(G)$. 

Thm [Balas '81]: Let $\hat{x}$ be a basic maximum fractional matching in an unweighted graph $G$. If $|C(\hat{x})| > \gamma(G)$, then there exists an $M(\hat{x})$-alternating path $P$ which connects two odd cycles $C_i, C_j \in C(\hat{x})$. Furthermore, alternate rounding on $C_i, C_j$ and complementing on $P$ produces a basic maximum fractional matching $\bar{x}$ in $G$ such that $C(\bar{x}) \subset C(\hat{x})$. 

Minimize number of odd cycles
Minimize number of odd cycles

Goal: Given a weighted graph $G$, compute a basic maximum-weight fractional matching $\hat{x}$ such that $|\mathcal{C}(\hat{x})| = \gamma(G)$.

Thm [Balas ’81]: Let $\hat{x}$ be a basic maximum fractional matching in an unweighted graph $G$. If $|\mathcal{C}(\hat{x})| > \gamma(G)$, then there exists an $M(\hat{x})$-alternating path $P$ which connects two odd cycles $C_i, C_j \in \mathcal{C}(\hat{x})$. 
Minimize number of odd cycles

**Goal:** Given a weighted graph $G$, compute a basic maximum-weight fractional matching $\hat{x}$ such that $|C(\hat{x})| = \gamma(G)$.

**Thm [Balas ’81]:** Let $\hat{x}$ be a basic maximum fractional matching in an unweighted graph $G$. If $|C(\hat{x})| > \gamma(G)$, then there exists an $M(\hat{x})$-alternating path $P$ which connects two odd cycles $C_i, C_j \in C(\hat{x})$. 
Minimize number of odd cycles

**Goal:** Given a weighted graph $G$, compute a basic maximum-weight fractional matching $\hat{x}$ such that $|C(\hat{x})| = \gamma(G)$.

**Thm [Balas ’81]:** Let $\hat{x}$ be a basic maximum fractional matching in an unweighted graph $G$. If $|C(\hat{x})| > \gamma(G)$, then there exists an $M(\hat{x})$-alternating path $P$ which connects two odd cycles $C_i, C_j \in C(\hat{x})$.

Furthermore, alternate rounding on $C_i, C_j$ and complementing on $P$ produces a basic maximum fractional matching $\bar{x}$ in $G$ such that $C(\bar{x}) \subset C(\hat{x})$. 
Minimize number of odd cycles

**Goal:** Given a weighted graph $G$, compute a basic maximum-weight fractional matching $\hat{x}$ such that $|C(\hat{x})| = \gamma(G)$.

**Thm [Balas '81]:** Let $\hat{x}$ be a basic maximum fractional matching in an unweighted graph $G$. If $|C(\hat{x})| > \gamma(G)$, then there exists an $M(\hat{x})$-alternating path $P$ which connects two odd cycles $C_i, C_j \in C(\hat{x})$.

Furthermore, alternate rounding on $C_i, C_j$ and complementing on $P$ produces a basic maximum fractional matching $\bar{x}$ in $G$ such that $C(\bar{x}) \subset C(\hat{x})$. 
Minimize number of odd cycles

**Goal:** Given a weighted graph $G$, compute a basic maximum-weight fractional matching $\hat{x}$ such that $|C(\hat{x})| = \gamma(G)$.

**Thm [Balas ’81]:** Let $\hat{x}$ be a basic maximum fractional matching in an unweighted graph $G$. If $|C(\hat{x})| > \gamma(G)$, then there exists an $M(\hat{x})$-alternating path $P$ which connects two odd cycles $C_i, C_j \in C(\hat{x})$.

Furthermore, alternate rounding on $C_i, C_j$ and complementing on $P$ produces a basic maximum fractional matching $\bar{x}$ in $G$ such that $C(\bar{x}) \subset C(\hat{x})$. 

![Diagram of graph with paths and cycles](image)
Minimize number of odd cycles

Thm 5:
Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $w$-vertex cover in $G$. If $|C(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:

- $y_v = 0$
- $C_i$ tight and valid
- $P_w$ tight

Furthermore, alternate rounding on the odd cycles and complementing on the path produces a basic maximum-weight fractional matching $\overline{x}$ such that $C(\overline{x}) \subset C(\hat{x})$. 
Minimize number of odd cycles

**Thm 5**: Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $\omega$-vertex cover in $G$. If $|C(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:
Minimize number of odd cycles

**Thm 5:** Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $w$-vertex cover in $G$. If $|C(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:

- $C_i$ tight and valid
- $P_{C_i}^j$ tight
- Furthermore, alternate rounding on the odd cycles and complementing on the path produces a basic maximum-weight fractional matching $\bar{x}$ such that $C(\bar{x}) \subset C(\hat{x})$. 
**Thm 5:** Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $w$-vertex cover in $G$. If $|C(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:

- $C_i$ with $y_v = 0$
- $C_i$ with tight and valid $P$ and $y_v = 0$
Thm 5: Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $w$-vertex cover in $G$. If $|\mathcal{C}(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:

- $y_v = 0$
- $C_i$ tight and valid $P$
- $C_i$ tight $P$
- $C_i$ and $C_j$ connected by a path $P$
Thm 5: Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $w$-vertex cover in $G$. If $|C(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:

Furthermore, alternate rounding on the odd cycles and complementing on the path produces a basic maximum-weight fractional matching $\bar{x}$ such that $C(\bar{x}) \subset C(\hat{x})$. 
Minimize number of odd cycles

**Thm 5:** Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $w$-vertex cover in $G$. If $|\mathcal{C}(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:

Furthermore, alternate rounding on the odd cycles and complementing on the path produces a basic maximum-weight fractional matching $\bar{x}$ such that $\mathcal{C}(\bar{x}) \subset \mathcal{C}(\hat{x})$. 

![Diagram of odd cycles and vertex cover](image-url)
Minimize number of odd cycles

**Thm 5:** Let $\hat{x}$ be a maximum-weight fractional matching and $y$ be a minimum fractional $w$-vertex cover in $G$. If $|\mathcal{C}(\hat{x})| > \gamma(G)$, then $G$ contains at least one of the following:

Furthermore, alternate rounding on the odd cycles and complementing on the path produces a basic maximum-weight fractional matching $\bar{x}$ such that $\mathcal{C}(\bar{x}) \subset \mathcal{C}(\hat{x})$. 

![Diagram of cycles and vertex cover](image-url)
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x} (\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x} (\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$.
5. Shrink every odd cycle $C_i \in C(\hat{x})$ into a pseudonode $i$.

Lemma: $M'$ is a maximum matching in $G'$ if and only if $|C(\hat{x})| = \gamma(G)$. 


Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

- **Step 1**: Delete all non-tight edges.
- **Step 2**: Add a vertex $z$.
- **Step 3**: For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
- **Step 4**: For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$ and $v'z$.
- **Step 5**: Shrink every odd cycle $C_i \in C(\hat{x})$ into a pseudonode $i$.

**Lemma**: $M'$ is a maximum matching in $G'$ if and only if $|C(\hat{x})| = \gamma(G)$.
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.

![Graph diagram]
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.

$\textbf{Lemma:}$ $M'$ is a maximum matching in $G'$ if and only if $|C(\hat{x})| = \gamma(G)$. 
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.

Lemma: $M'$ is a maximum matching in $G'$ if and only if $|\hat{C}(\hat{x})| = \gamma(G)$. 

\[
\begin{array}{c}
\text{Construct the unweighted graph } G' \text{ as follows:} \\
\begin{enumerate}
\item Delete all non-tight edges. \\
\item Add a vertex $z$. \\
\end{enumerate}
\end{array}
\]
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$.
5. Shrink every odd cycle $C_i \in C(\hat{x})$ into a pseudonode $i$.

Lemma: $M'$ is a maximum matching in $G'$ if and only if $\gamma(G) = |C(\hat{x})|$. 
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$ and $v'z$.

Lemma: $M'$ is a maximum matching in $G'$ if and only if $|C(\hat{x})| = \gamma(G)$.
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$. 

**Lemma:** $M'$ is a maximum matching in $G'$ if and only if $|\mathcal{C}(\hat{x})| = \gamma(G)$. 
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$. 

Lemma: $M'$ is a maximum matching in $G'$ if and only if $|C(\hat{x})| = \gamma(G)$. 

\[ G \]

\[ M' \]

\[ Z \]
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$.
5. Shrink every odd cycle $C_i \in \mathcal{C}(\hat{x})$ into a pseudonode $i$. 

Lemma: $M'$ is a maximum matching in $G'$ if and only if $|C(\hat{x})| = \gamma(G)$.
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$.
5. Shrink every odd cycle $C_i \in \mathcal{C}(\hat{x})$ into a pseudonode $i$. 
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$.
5. Shrink every odd cycle $C_i \in \mathcal{C}(\hat{x})$ into a pseudonode $i$. 

\[ \text{Lemma:} \quad M' \text{ is a maximum matching in } G' \text{ if and only if } |\mathcal{C}(\hat{x})| = \gamma(G'). \]
Minimize number of odd cycles

Construct the unweighted graph $G'$ as follows:

1. Delete all non-tight edges.
2. Add a vertex $z$.
3. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 1$ and $y_v = 0$, add edge $vz$.
4. For every vertex $v \in V$ where $\hat{x}(\delta(v)) = 0$ and $y_v = 0$, add the vertex $v'$ and edges $vv'$, $v'z$.
5. Shrink every odd cycle $C_i \in \mathcal{C}(\hat{x})$ into a pseudonode $i$.

Lemma: $M'$ is a maximum matching in $G'$ if and only if $|\mathcal{C}(\hat{x})| = \gamma(G)$.
Computing vertex-stabilizers

**Theorem 1:**
The algorithm computes a minimum vertex-stabilizer $S$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$.

**Proof:**
- **Stability** - due to complementary slackness.
- **Optimality** - $\gamma(G)$ is a lower bound on the size of a vertex-stabilizer.
Computing vertex-stabilizers

**Thm 1:** The algorithm computes a minimum vertex-stabilizer $S$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$. 

Proof: Stability - due to complementary slackness. Optimality - $\gamma(G)$ is a lower bound on the size of a vertex-stabilizer.
Computing vertex-stabilizers

**Thm 1:** The algorithm computes a minimum vertex-stabilizer $S$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$.

*Proof:* Stability - due to complementary slackness.
Computing vertex-stabilizers

**Thm 1:** The algorithm computes a minimum vertex-stabilizer $S$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$.

*Proof: Stability* - due to complementary slackness.
Thm 1: The algorithm computes a minimum vertex-stabilizer $S$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$.

Proof: Stability - due to complementary slackness.
Thm 1: The algorithm computes a minimum vertex-stabilizer $S$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$.

Proof: Stability - due to complementary slackness.
Computing vertex-stabilizers

**Thm 1:** The algorithm computes a minimum vertex-stabilizer $S$. Moreover, $\nu(G \setminus S) \geq \frac{2}{3} \nu(G)$.

*Proof:* Stability - due to complementary slackness.

Optimality - $\gamma(G)$ is a lower bound on the size of a vertex-stabilizer.
Lemma: For any vertex $v$, $\gamma(G \setminus v) \geq \gamma(G) - 1$.

Proof: Let $\hat{x}$ be a maximum-weight fractional matching in $G$ with an odd cycle $C(\hat{x})$.

Easy case: $v$ lies in a cycle of $C(\hat{x})$.

Hard case: $v$ does not lie in a cycle of $C(\hat{x})$. 
Lemma: For any vertex \( v \), \( \gamma(G \setminus v) \geq \gamma(G) - 1 \).
Lemma: For any vertex \( v \), \( \gamma(G \setminus v) \geq \gamma(G) - 1 \).

Proof: Let \( \hat{x} \) be a maximum-weight fractional matching in \( G \) with \( \gamma(G) \) odd cycles.
**Lower bound**

**Lemma:** For any vertex \( v \), \( \gamma(G \setminus v) \geq \gamma(G) - 1 \).

**Proof:** Let \( \hat{x} \) be a maximum-weight fractional matching in \( G \) with \( \gamma(G) \) odd cycles.

Diagram:

- Cycle with \( \hat{x} \) edges.
- Other edges for odd cycles.
Lemma: For any vertex $v$, $\gamma(G \setminus v) \geq \gamma(G) - 1$.

Proof: Let $\hat{x}$ be a maximum-weight fractional matching in $G$ with $\gamma(G)$ odd cycles.

Easy case: $v$ lies in a cycle of $C(\hat{x})$. 
Lemma: For any vertex $v$, $\gamma(G \setminus v) \geq \gamma(G) - 1$.

Proof: Let $\hat{x}$ be a maximum-weight fractional matching in $G$ with $\gamma(G)$ odd cycles.

Easy case: $v$ lies in a cycle of $\mathcal{C}(\hat{x})$. 
**Lower bound**

**Lemma:** For any vertex $v$, $\gamma(G \setminus v) \geq \gamma(G) - 1$.

**Proof:** Let $\hat{x}$ be a maximum-weight fractional matching in $G$ with $\gamma(G)$ odd cycles.

1. **Easy case:** $v$ lies in a cycle of $C(\hat{x})$.
2. **Hard case:** $v$ does not lie in a cycle of $C(\hat{x})$. 
**Lemma:** For any vertex $v$, $\gamma(G \setminus v) \geq \gamma(G) - 1$.

**Proof:** Let $\hat{x}$ be a maximum-weight fractional matching in $G$ with $\gamma(G)$ odd cycles.

Easy case: $v$ lies in a cycle of $C(\hat{x})$.

Hard case: $v$ does not lie in a cycle of $C(\hat{x})$. 

---

![Diagram of a graph with vertices and edges, illustrating the lemma and proof.](image-url)
Can we do better?

Can we preserve more than $\frac{2}{3} \nu(G)$?

No!

For any subset $S \subseteq V$, $\nu(G \setminus S) \leq 2 = \frac{2}{3} - \varepsilon \nu(G)$.

Can we decide if $G$ has a weight-preserving vertex-stabilizer $S$, i.e., $\nu(G \setminus S) = \nu(G)$?

NP-complete!
Can we do better?

- Can we preserve more than $\frac{2}{3} \nu(G)$?
Can we do better?

- Can we preserve more than $\frac{2}{3} \nu(G)$? No!
Can we do better?

- Can we preserve more than $\frac{2}{3} \nu(G)$? No!

![Diagram](image)
Can we do better?

- Can we preserve more than \( \frac{2}{3} \nu(G) \)? No!

For any subset \( S \subseteq V \),

\[
\nu(G \setminus S) \leq 2 = \frac{2}{3 - \varepsilon} \nu(G)
\]
Can we do better?

- Can we preserve more than $\frac{2}{3} \nu(G)$? No!

For any subset $S \subseteq V$,

$$\nu(G \setminus S) \leq 2 = \frac{2}{3 - \varepsilon} \nu(G)$$

- Can we decide if $G$ has a weight-preserving vertex-stabilizer $S$, i.e.

$$\nu(G \setminus S) = \nu(G)?$$
Can we do better?

- Can we preserve more than $\frac{2}{3} \nu(G)$? No!

For any subset $S \subseteq V$,

$$\nu(G \setminus S) \leq 2 = \frac{2}{3 - \varepsilon} \nu(G)$$

- Can we decide if $G$ has a weight-preserving vertex-stabilizer $S$, i.e.

$$\nu(G \setminus S) = \nu(G) ?$$

NP-complete!
Computing edge-stabilizers

• In contrast to vertex-stabilizers, $\gamma(G)$ is not a lower bound.

Lemma: For any edge $e$, $\gamma(G \setminus e) \geq \gamma(G) - 2$.

Lower Bound: Every edge-stabilizer has size at least $\lceil \gamma(G) / 2 \rceil$.

Thm 4: There exists an $O(\Delta)$-approximation algorithm for the minimum edge-stabilizer problem.
Computing edge-stabilizers

- In contrast to vertex-stabilizers, \( \gamma(G) \) is not a lower bound.
Computing edge-stabilizers

- In contrast to vertex-stabilizers, $\gamma(G)$ is not a lower bound.

![Graph Diagram]

**Lemma:**
For any edge $e$, $\gamma(G - e) \geq \gamma(G) - 2$.

**Lower Bound:**
Every edge-stabilizer has size at least $\lceil \gamma(G)^2 \rceil$.

**Thm 4:**
There exists an $O(\Delta)$-approximation algorithm for the minimum edge-stabilizer problem.
In contrast to vertex-stabilizers, $\gamma(G)$ is not a lower bound.
Computing edge-stabilizers

- In contrast to vertex-stabilizers, $\gamma(G)$ is not a lower bound.

![Diagram of a graph showing edge-stabilizers](image_url)
Computing edge-stabilizers

• In contrast to vertex-stabilizers, $\gamma(G)$ is not a lower bound.

**Lemma:** For any edge $e$, $\gamma(G \setminus e) \geq \gamma(G) - 2$. 
Computing edge-stabilizers

• In contrast to vertex-stabilizers, $\gamma(G)$ is not a lower bound.

Lemma: For any edge $e$, $\gamma(G \setminus e) \geq \gamma(G) - 2$.

Lower Bound: Every edge-stabilizer has size at least $\left\lceil \frac{\gamma(G)}{2} \right\rceil$. 
Computing edge-stabilizers

- In contrast to vertex-stabilizers, $\gamma(G)$ is not a lower bound.

![Diagram showing edge-stabilizers](image)

**Lemma:** For any edge $e$, $\gamma(G \setminus e) \geq \gamma(G) - 2$.

**Lower Bound:** Every edge-stabilizer has size at least $\left\lceil \frac{\gamma(G)}{2} \right\rceil$.

**Thm 4:** There exists an $O(\Delta)$-approximation algorithm for the minimum edge-stabilizer problem.
Additional results

- Given a set of deals $M$, remove as few players as possible such that $M$ is realizable as a stable outcome.

  $\rightarrow$ Find a minimum vertex-stabilizer $S$ such that $M$ is a maximum-weight matching in $G \setminus S$.

- A solution to this problem is called an $M$-vertex-stabilizer.

**Thm [Ahmadian et al. ’16]**: If $M$ is a maximum matching in an unweighted graph, then it is polytime solvable.

**Thm 6**: The problem is $\mathbf{NP}$-hard on unweighted graphs. Moreover, no $(2 - \varepsilon)$-approximation algorithm exists for any $\varepsilon > 0$ assuming UGC.

**Thm 7**: The problem admits a 2-approximation algorithm on weighted graphs. Furthermore, if $M$ is a maximum-weight matching, then it is polytime solvable.
Thank you!
Appendix 1

**Thm 2:** Deciding whether a graph has a weight-preserving vertex-stabilizer is **NP**-complete.

**Proof:** Reduction from the independent set problem.

Construct the gadget graph $G^*$ as follows:

$G$ has an independent set of size $k$ $\iff$ $G^*$ has a weight-preserving vertex-stabilizer. $\square$
Appendix 2

Thm 3: There is no constant factor approximation for the minimum edge-stabilizer problem unless \( P = NP \).

Proof: Suppose we have an \( \alpha \)-approximation algorithm. Set \( \rho = \lceil \alpha \rceil \).

- If \( G \) has an independent set of size \( k \), then \( \text{OPT} \leq k \).
- Else, \( \text{OPT} \geq (\rho + 1)k \). \( \square \)