Markov Chain Mixing Times And Applications III:

**Conductance**

and

**Canonical Paths**

Ivona Bezáková

(Rochester Institute of Technology)

Simons Institute for the Theory of Computing
Counting Complexity and Phase Transitions Bootcamp

January 27th, 2016
Outline

Other techniques for bounding the mixing time:

- conductance
- canonical paths
- canonical flows

Thanks to:

Bhatnagar, Diaconis, Dyer, Jerrum, Lawler, Müller, Randall, Sinclair, Sokal, Štefankovič, Stroock, Vazirani, Vigoda, ...
Recall:

- Ergodic MC \((\Omega, P)\) \(\Rightarrow\) unique stationary distribution \(\pi\)
- Mixing time: \(t_{\text{mix}}(\varepsilon) = \text{minimum } t \text{ such that for every start state } x, \text{ after } t \text{ steps within } \varepsilon \text{ of } \pi\)

An ergodic reversible Markov chain \((\Omega, P)\):
Recall:

• Ergodic MC \((\Omega, P)\) \(\Rightarrow\) unique stationary distribution \(\pi\)

• Mixing time: \(t_{\text{mix}}(\epsilon)\) = minimum \(t\) such that for every start state \(x\), after \(t\) steps within \(\epsilon\) of \(\pi\)

An ergodic reversible Markov chain \((\Omega, P)\):
Conductance

**Def:** For an ergodic reversible MC \((\Omega, P)\), its conductance is defined as:

\[
\Phi = \min_{S: S \subseteq \Omega, \pi(S) \leq 1/2} \frac{\sum_{x \in S, y \notin S} \pi(x)P(x, y)}{\pi(S)}
\]
Conductance

**Def:** For an ergodic reversible MC \((\Omega, P)\), its conductance is defined as:

\[
\Phi = \min_{S: S \subseteq \Omega, \pi(S) \leq 1/2} \frac{\sum_{x \in S, y \notin S} \pi(x) P(x, y)}{\pi(S)}
\]

**Example:** suppose \(\pi\) is uniform:

![Diagram](image-url)
Def: For an ergodic reversible MC $\Omega, P$, its conductance is defined as:

$$\Phi = \min_{S: S \subseteq \Omega, \pi(S) \leq 1/2} \frac{\sum_{x \in S, y \notin S} \pi(x)P(x, y)}{\pi(S)}$$

Example: suppose $\pi$ is uniform:

$$\pi(S) = 3/11 \leq 1/2$$
Conductance

**Def:** For an ergodic reversible MC \( (\Omega, P) \), its conductance is defined as:

\[
\Phi = \min_{S: S \subseteq \Omega, \pi(S) \leq 1/2} \frac{\sum_{x \in S, y \notin S} \pi(x)P(x, y)}{\pi(S)}
\]

**Example:** suppose \( \pi \) is uniform:

\[
\pi(S) = \frac{3}{11} \leq \frac{1}{2}
\]
**Conductance**

**Def:** For an ergodic reversible MC \((\Omega, P)\), its **conductance** is defined as:

\[
\Phi = \min_{S:\ S \subseteq \Omega, \pi(S) \leq 1/2} \sum_{x \in S, y \notin S} \frac{\pi(x)P(x, y)}{\pi(S)}
\]

**Example:** suppose \(\pi\) is uniform:

\[
\Phi_S = \frac{1}{11} \cdot 0.1 + \frac{1}{11} \cdot 0.3 + \frac{1}{11} \cdot 0.1 + \frac{1}{11} \cdot 0.4 = 0.3
\]

\[\pi(S) = 3/11 \leq 1/2\]
Conductance

**Def:** For an ergodic reversible MC \((\Omega, P)\), its conductance is defined as:

\[
\Phi = \min_{S : S \subseteq \Omega, \pi(S) \leq 1/2} \frac{\sum_{x \in S, y \notin S} \pi(x)P(x, y)}{\pi(S)}
\]

**Example:** suppose \(\pi\) is uniform:

\[
\Phi_S = \frac{1}{11} \cdot \frac{0.1}{5} = 0.02
\]

\[\pi(S) = \frac{5}{11} \leq \frac{1}{2}\]
**Conductance**

**Def:** For an ergodic reversible $MC (\Omega, P)$, its **conductance** is defined as:

$$\Phi = \min_{S:S \subseteq \Omega, \pi(S) \leq 1/2} \sum_{x \in S, y \notin S} \frac{\pi(x)P(x,y)}{\pi(S)}$$

**Thm:** $\Phi^2/2 \leq \text{spectral gap} \leq 2\Phi$

Recall:

**Thm:** For an ergodic $MC$, let $\lambda_2$ be the 2$^{nd}$ largest eigenvalue of $P$ and $\pi_{\min} := \min_x \pi(x)$. Then

$$\frac{|\lambda_2|}{\text{spectral gap}} \log\left(\frac{1}{2\varepsilon}\right) \leq t_{\text{mix}}(\varepsilon) \leq \frac{1}{\text{spectral gap}} \log\left(\frac{1}{\varepsilon\pi_{\min}}\right)$$

[Jerrum-Sinclair, Diaconis-Stroock, Lawler-Sokal]
**Conductance**

**Def:** For an ergodic reversible MC $(\Omega, P)$, its conductance is defined as:

$$\Phi = \min_{S: S \subseteq \Omega, \pi(S) \leq 1/2} \sum_{x \in S, y \notin S} \frac{\pi(x)P(x, y)}{\pi(S)}$$

**Thm:** $\frac{\Phi^2}{2} \leq \text{spectral gap} \leq 2\Phi$

**Thm:** For a lazy ergodic MC, where $\pi_{\text{min}} := \min_x \pi(x)$:

$$\frac{1}{2} \left( \frac{1}{2\Phi} - 1 \right) \log \left( \frac{1}{2\epsilon} \right) \leq t_{\text{mix}}(\epsilon) \leq \frac{2}{\Phi^2} \log \left( \frac{1}{\epsilon \pi_{\text{min}}} \right)$$

[Jerrum-Sinclair, Diaconis-Stroock, Lawler-Sokal]
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state $I$ to every other state $F$:
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state I to every other state F ($|\Omega|x|\Omega|$ paths)
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega| \times |\Omega|$ paths)

[Jerrum-Sinclair]
Canonical Paths

Bounding the conductance:
- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega| \times |\Omega|$ paths)
- Then, take any $S$:

$$\pi(S) \pi(\overline{S}) = \sum_{I \in S, F \notin S} \pi(I) \pi(F)$$

[Jerrum-Sinclair]
Canonical Paths

Bounding the conductance:
- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega| \times |\Omega|$ paths)
- Then, take any $S$:

$$\frac{\pi(S)\pi(\overline{S})}{\sum_{x \in S, y \notin S} \pi(x)P(x, y)} = \frac{\sum_{I \in S, F \notin S} \pi(I)\pi(F)}{\sum_{x \in S, y \notin S} \pi(x)P(x, y)}$$

[Jerrum-Sinclair]
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state $I$ to every other state $F (|\Omega|x|\Omega| \text{ paths})$

- Then, take any $S$: 

\[
\frac{\pi(S)/2}{\sum_{x \in S, y \notin S} \pi(x)P(x,y)} \leq \frac{\pi(S)\pi(S)'}{\sum_{x \in S, y \notin S} \pi(x)P(x,y)} = \frac{\sum_{I \in S, F \notin S} \pi(I)\pi(F)}{\sum_{x \in S, y \notin S} \pi(x)P(x,y)}
\]

\[\text{[Jerrum-Sinclair]}\]
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega| \times |\Omega|$ paths)

- Then, let $S$ be the “smallest cut”:

\[
\frac{1}{2\Phi} = \frac{\pi(S) / 2}{\sum_{x \in S, y \notin S} \pi(x)P(x, y)} \leq \frac{\pi(S)\pi(\overline{S})}{\sum_{x \in S, y \notin S} \pi(x)P(x, y)} = \sum_{I \in S, F \notin S} \pi(I)\pi(F)
\]

[Jerrum-Sinclair]
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega|x|\Omega|$ paths)

- Then, let $S$ be the “smallest cut”:

$$\frac{1}{2\Phi} = \frac{\pi(S)/2}{\sum_{x \in S, y \not\in S} \pi(x)P(x, y)} \leq \frac{\pi(S)\pi(\bar{S})}{\sum_{x \in S, y \not\in S} \pi(x)P(x, y)} = \frac{\sum_{I \in S, F \not\in S} \pi(I)\pi(F)}{\sum_{x \in S, y \not\in S} \pi(x)P(x, y)} \leq \frac{\sum_{x \in S, y \not\in S} \sum_{(I,F):I \in S, F \not\in S, (x,y) \text{ on } I \to F} \pi(I)\pi(F)}{\sum_{x \in S, y \not\in S} \pi(x)P(x, y)}$$

[Jerrum-Sinclair]
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega| \times |\Omega|$ paths)

- Then, let $S$ be the “smallest cut”:

\[
\frac{1}{2\Phi} = \frac{\pi(S)/2}{\sum_{x \in S, y \not\in S} \pi(x)P(x, y)} \leq \frac{\pi(S)\pi(\bar{S})}{\sum_{x \in S, y \not\in S} \pi(x)P(x, y)} = \sum_{I \in S, F \not\in S} \pi(I)\pi(F) \leq \sum_{x \in S, y \not\in S} \pi(x)P(x, y) \leq \sum_{(I, F): I \in S, F \not\in S, (u, v) \text{ on } I \to F \text{ path}} \pi(u)P(u, v)
\]

for some $u$ in $S$, $v$ not in $S.$

[Jerrum-Sinclair]
Canonical Paths

Bounding the conductance:
- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega|x|\Omega|$ paths)
- Then, for some $u$ in $S$, $v$ not in $S$:

\[
\frac{1}{2\Phi} \leq \sum_{(I,F):I \in S, F \notin S, (u,v) \text{ on } I \to F \text{ path}} \frac{\pi(I)\pi(F)}{\pi(u)P(u,v)}
\]
Canonical Paths

Bounding the conductance:
- Find a path in the transition graph from every state I to every other state F ($|\Omega| \times |\Omega|$ paths)
- Then, for some $u$ in $S$, $v$ not in $S$:
  $$\frac{1}{2\Phi} \leq \sum_{(I,F): I \in S, F \notin S, (u,v)\text{ on } I \rightarrow F \text{ path}} \pi(I)\pi(F) \frac{\pi(u)P(u,v)}{\pi(u)P(u,v)}$$

Congestion through transition $(u,v)$
Canonical Paths

Bounding the conductance:
- Find a path in the transition graph from every state $I$ to every other state $F$ ($|\Omega| \times |\Omega|$ paths)

Def: Congestion

$$\rho := \max_{u,v} \frac{1}{\pi(u)P(u,v)} \sum_{I \to F \text{ path through } (u,v)} \pi(I) \pi(F) \text{(length of } I \to F \text{ path)}$$
Canonical Paths

Bounding the conductance:

- Find a path in the transition graph from every state I to every other state F (|Ω|×|Ω| paths)

**Def:** Congestion

\[
\rho := \max_{u,v} \frac{1}{\pi(u)P(u,v)} \sum_{I \rightarrow F \text{ path through } (u,v)} \pi(I)\pi(F) \text{(length of } I \rightarrow F \text{ path)}
\]

**Thm** [Sinclair]: For a lazy ergodic reversible MC:

\[
t_{\text{mix}}(\varepsilon) \leq 4 \rho \ln\left(\frac{1}{\varepsilon \pi_{\text{min}}}ight)
\]
Given an undirected graph $G=(V,E)$, a matching $M \subseteq E$ is a set of vertex disjoint edges. A matching is perfect if $|M|=n/2$, where $n = \# \text{ vertices}$ (and $m = \# \text{ edges}$).

Example:
Matchings Revisited

Given an undirected graph $G=(V,E)$, a **matching** $M \subseteq E$ is a set of vertex disjoint edges. A matching is **perfect** if $|M|=n/2$, where $n = \# \text{ vertices}$ (and $m = \# \text{ edges}$).

Example:

A matching
Matchings Revisited

Given an undirected graph $G=(V,E)$, a matching $M \subseteq E$ is a set of vertex disjoint edges. A matching is **perfect** if $|M|=n/2$, where $n = \# \text{ vertices}$ (and $m = \# \text{ edges}$).

**Example:**

A perfect matching
Given an undirected graph $G=(V,E)$, a **matching** $M \subseteq E$ is a set of vertex disjoint edges. A matching is **perfect** if $|M| = n/2$, where $n = \#$ vertices (and $m = \#$ edges).

**Example:**

A perfect matching

**Goal:**

An FPRAS for

- # matchings
- # perfect matchings
Given an undirected graph $G=(V,E)$, a matching $M \subseteq E$ is a set of vertex disjoint edges. A matching is perfect if $|M|=n/2$, where $n = \#$ vertices (and $m = \#$ edges).

**Example:**
A perfect matching

**Goal:**
An FPRAS for
- # matchings
- # perfect matchings
**A Markov Chain for Matchings**

**Input:** a graph $G$

**State space $\Omega$:** all matchings of $G$

**Markov chain (slide chain):** Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

**Input:** a graph \( G \)

**State space** \( \Omega \): all matchings of \( G \)

**Markov chain** (slide chain):
Let \( M \) be the current matching, we get the next state by choosing a random edge \( e=(u,v) \in E \) and:

- if \( e \in M \), remove \( e \) from \( M \)
- if \( u,v \) are not covered by edges in \( M \), add \( e \) to \( M \)
- if \( u \) is covered by edge \( e' \in M \) and \( v \) is not covered by \( M \), replace \( e' \) with \( e \) in \( M \)
- otherwise, stay in \( M \)
A Markov Chain for Matchings

**Input:** a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain): Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**Input:** a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**A Markov Chain for Matchings**

**Input:** a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain):

Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**Input:** a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain): Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
Input: a graph $G$

State space $\Omega$: all matchings of $G$

Markov chain (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
Input: a graph $G$

State space $\Omega$: all matchings of $G$

Markov chain (slide chain): Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

**Input**: a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain): Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u, v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

**Input:** a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**A Markov Chain for Matchings**

**Input:** a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain): Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

**Input**: a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**Input**: a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
Input: a graph $G$

State space $\Omega$: all matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

**Input:** a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

**Input**: a graph $G$

**State space** $\Omega$: all matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**A Markov Chain for Matchings**

**Input:** a graph $G$

**State space $\Omega$:** all matchings of $G$

**Markov chain (slide chain):** Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**A Markov Chain for Matchings**

**Input:** a graph $G$

**State space $\Omega$:** all matchings of $G$

**Markov chain** (slide chain): Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

Input: a graph $G$

State space $\Omega$: all matchings of $G$

Markov chain (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
**A Markov Chain for Matchings**

**Input:** a graph $G$

**State space $\Omega$:** all matchings of $G$

**Markov chain (slide chain):**

Let $M$ be the current matching, we get the next state by choosing a random edge $e=(u,v) \in E$ and:

- if $e \in M$, remove $e$ from $M$
- if $u,v$ are not covered by edges in $M$, add $e$ to $M$
- if $u$ is covered by edge $e' \in M$ and $v$ is not covered by $M$, replace $e'$ with $e$ in $M$
- otherwise, stay in $M$
A Markov Chain for Matchings

**Input**: a graph \( G \)

**State space** \( \Omega \): all matchings of \( G \)

**Markov chain** (slide chain):
Let \( M \) be the current matching, we get the next state by choosing a random edge \( e=(u,v) \in E \) and:

- if \( e \in M \), remove \( e \) from \( M \)
- if \( u,v \) are not covered by edges in \( M \), add \( e \) to \( M \)
- if \( u \) is covered by edge \( e' \in M \) and \( v \) is not covered by \( M \), replace \( e' \) with \( e \) in \( M \)
- otherwise, stay in \( M \)

**Technicality**:
A lazy chain: with probability 1/2 stay in \( M \), otherwise
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

[Diagram showing a transition graph with paths marked in red and blue, labeled $I$ and $F$.]
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
Canonical Paths

For any pair of matchings I,F, define a path from I to F in the transition graph:

Going from red to blue:
• take $I \oplus F$ (sym. difference)
• components are alternating cycles or paths
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

**Canonical Paths**

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

**Canonical Paths**

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if cycle:
    - remove lowest edge
    - slide the rest
    - add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

**Canonical Paths**

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if cycle:
    - remove lowest edge
    - slide the rest
    - add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I,F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if cycle:
    - remove lowest edge
    - slide the rest
    - add the last edge

(dashed edges: not in the current matching)
Canonical Paths

For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if cycle:
    - remove lowest edge
    - slide the rest
    - add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I$,$F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if cycle:
    - remove lowest edge
    - slide the rest
    - add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

**Canonical Paths**

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if cycle:
    - remove lowest edge
    - slide the rest
    - add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I,F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if cycle:
    - remove lowest edge
    - slide the rest
    - add the last edge

(dashed edges: not in the current matching)
For any pair of matchings I,F, define a path from I to F in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge
For any pair of matchings $I,F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge

(dashed edges: not in the current matching)
For any pair of matchings I,F, define a path from I to F in the transition graph:

**Canonical Paths**

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge
Canonical Paths

For any pair of matchings \( I, F \), define a path from \( I \) to \( F \) in the transition graph:

Going from red to blue:
- take \( I \oplus F \) (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

**Canonical Paths**

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge

(dashed edges: not in the current matching)
For any pair of matchings $I, F$, define a path from $I$ to $F$ in the transition graph:

Going from red to blue:
- take $I \oplus F$ (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge

(dashed edges: not in the current matching)
For any pair of matchings I, F, define a path from I to F in the transition graph:

Going from red to blue:
- take I ⊕ F (sym. difference)
- components are alternating cycles or paths
- order the components by the lowest vertex
- process components in order
  - if path:
    - if needed, remove lower end edge
    - slide the rest
    - if needed, add the last edge

(dashed edges: not in the current matching)
Bounding the Congestion

Congestion through transition \( M \rightarrow M' \):

\[
\frac{1}{\pi(M)P(M,M')} \sum_{I \rightarrow F \text{ path through } M \rightarrow M'} \pi(I)\pi(F) \text{ length}(I \rightarrow F)
\]

Since \( \pi(M) = \pi(I) = \pi(F) = 1/|\Omega| \) and \( P(M,M') = 1/(2m) \), and length(I→F) ≤ n:

\[
\leq \frac{2m}{|\Omega|} \sum_{I \rightarrow F \text{ path through } M \rightarrow M'} n
\]

\[
= \frac{2mn}{|\Omega|} \text{ (# canonical paths through } M \rightarrow M')
\]
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega| \text{poly}(n)$]

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching

An $I \rightarrow F$ path through $M \rightarrow M'$

$M \rightarrow M'$
Let $M \rightarrow M'$ be a transition.
How many canonical paths go through it? \[\text{Want } \leq |\Omega| \text{poly}(n)\]

An $\mathbf{I} \rightarrow \mathbf{F}$ path through $M \rightarrow M'$

**Legend:**
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
Bounding the Congestion: Encoding

Let $M \rightarrow M'$ be a transition.
How many canonical paths go through it? [Want $\leq |\Omega| \text{poly}(n)$]

An $I \rightarrow F$ path through $M \rightarrow M'$

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
Let $M\rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

An $I \rightarrow F$ path through $M \rightarrow M'$

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
Let $M\rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching

An $I\rightarrow F$ path through $M\rightarrow M'$

**Observation:**
$I \oplus F - (M \cup M')$ is a matching.

$\Rightarrow$ encoding $E$
(for $I,F$ given $M$)
Let $M \rightarrow M'$ be a transition.
How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

An $I \rightarrow F$ path through $M \rightarrow M'$

Observation:
$I \oplus F - (M \cup M')$ is a matching.

$\Rightarrow$ encoding $E$ (for $I,F$ given $M$)

Legend:
- purple: transition
- red: initial matching
- blue: final matching
- orange: encoding
Bounding the Congestion: Encoding

Let $M \rightarrow M'$ be a transition.
How many canonical paths go through it? [Want $\leq |\Omega| \text{poly}(n)$]

$E = I \oplus F - (M \cup M')$

Is $E$ an “encoding”? (Given $M \rightarrow M'$ and $E$, can reconstruct $I,F$?)
If yes, then $\# \text{ can.paths through } M \rightarrow M' \leq |\Omega|$
Bounding the Congestion: Encoding

Reconstructing $I,F$ from $E, M \rightarrow M'$:

$E = I \oplus F - (M \cup M')$

Is $E$ an “encoding”? (Given $M \rightarrow M'$ and $E$, can reconstruct $I,F$?)
If yes, then $\# \text{ can.paths through } M \rightarrow M'$ is $\leq |\Omega|$
Bounding the Congestion: Encoding

Reconstructing $\mathbf{I}, \mathbf{F}$ from $\mathbf{E}, \mathbf{M} \rightarrow \mathbf{M}'$

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Know:
- order of components
- currently working on 2\textsuperscript{nd}
- 1\textsuperscript{st} done, 3\textsuperscript{rd} not yet
- current: done up to the transition

$E = I \oplus F - (MUM')$
Bounding the Congestion: Encoding

Reconstructing $I, F$ from $E, M \rightarrow M'$:

$E = I \oplus F - (M \cup M')$

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Know:
- order of components
- currently working on $2^{nd}$
- $1^{st}$ done, $3^{rd}$ not yet
- current: done up to the transition
Bounding the Congestion: Encoding

Reconstructing $I,F$ from $E,M\rightarrow M'$:

$E = I \oplus F - (MUM')$

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Know:
- order of components
- currently working on $2^{nd}$
- $1^{st}$ done, $3^{rd}$ not yet
- current: done up to the transition
Bounding the Congestion: Encoding

Reconstructing $I, F$ from $E, M \rightarrow M'$:

Know:
- order of components
- currently working on $2^{nd}$
- $1^{st}$ done, $3^{rd}$ not yet
- current: done up to the transition

Legend:
- purple: transition
- red: initial matching
- blue: final matching
- orange: encoding
Bounding the Congestion: Encoding

Reconstructing $I,F$ from $E,M \rightarrow M'$:

\[ E = I \oplus F - (M \cup M') \]

Legend:
- purple: transition
- red: initial matching
- blue: final matching
- orange: encoding

Know:
- order of components
- currently working on 2\textsuperscript{nd}
- 1\textsuperscript{st} done, 3\textsuperscript{rd} not yet
- current: done up to the transition
Bounding the Congestion: Encoding

Reconstructing $I, F$ from $E, M \rightarrow M'$:

$E = I \oplus F - (M \cup M')$

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Know:
- order of components
- currently working on $2^{nd}$
- $1^{st}$ done, $3^{rd}$ not yet
- current: done up to the transition
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

**Legend:**
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

$$E = I \oplus F - (M \cup M')$$

Not a matching…
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

**Legend:**
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

$E = I \oplus F - (M \cup M') - \{e\}$

Now a matching: $E \in \Omega$
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

**Legend:**
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Can “decode” $E$ into $I,F$?

$$E = I \oplus F - (M \cup M') - \{e\}$$

Now a matching: $E \in \Omega$
Let $M \rightarrow M'$ be a transition.
How many canonical paths go through it? [Want $\leq |\Omega| \text{poly}(n)$]

Bounding the Congestion: Encoding

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Can “decode” $E$ into $I,F$?

$$E = I \oplus F - (M \cup M') - \{e\}$$

Now a matching: $E \in \Omega$
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

**Legend:**
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

$$E = I \oplus F - (M \cup M') - \{e\}$$

Now a matching: $E \in \Omega$

Can “decode” $E$ into $I,F$?
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega| \text{poly}(n)$]

Bounding the Congestion: Encoding

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Can “decode” $E$ into $I,F$?

\[ E = I \oplus F - (M \cup M') - \{e\} \]

Now a matching: $E \in \Omega$
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

Bounding the Congestion: Encoding

$E = I \oplus F - (M \cup M') - \{e\}$

Now a matching: $E \in \Omega$

Can “decode” $E$ into $I, F$?
- Is 2$^{nd}$ component a path?

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

**Legend:**
- purple: transition
- red: initial matching
- blue: final matching
- orange: encoding

**Diagram:**
- $1^{st}$ component
- $2^{nd}$ component

Now a matching: $E = I \oplus F - (M \cup \nu M') - \{e\}$

Can “decode” $E$ into $I,F$?
- Is $2^{nd}$ component a path?
Let $M \rightarrow M'$ be a transition.
How many canonical paths go through it? \[\text{Want } \leq |\Omega|\text{poly}(n)\]

**Legend:**
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

**Diagram:**
- $E = I \oplus F - (M \cup M') - \{e\}$
- Now a matching: $E \in \Omega$

**Can “decode” $E$ into $I,F$?**
- Is 2$\text{nd}$ component a path?
- Or a cycle?
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|\text{poly}(n)$]

Legend:
- **purple**: transition
- **red**: initial matching
- **blue**: final matching
- **orange**: encoding

Can “decode” $E$ into $I,F$?
- Is 2\textsuperscript{nd} component a path?
- Or a cycle?

Redefine encoding:
$E' := (E,0)$
$E' := (E,1)$
$\in \Omega \times \{0,1\}$
Let $M \rightarrow M'$ be a transition. How many canonical paths go through it? [Want $\leq |\Omega|poly(n)$]

- For the sliding transition: $\leq 2|\Omega|$

- Need to analyze the add and remove transitions

**Bounding the Congestion: Encoding**

$$\rho \leq \frac{2mn}{|\Omega|}$$ (# can. paths through $M \rightarrow M') = \frac{2mn}{|\Omega|} 2|\Omega| = 4mn$$

**Mixing time:**

$$t_{\text{mix}}(\epsilon) = O(mn \log(1/(\epsilon \pi_{\text{min}})))$$

$$= O^*(mn^2)$$

[O* - ignore polylog]

**FPRAS:**

$$O( T(n,m,\epsilon/(6m)) m^2/\epsilon^2 ) = O^*(m^3n^2/\epsilon^2)$$
Input: a graph $G$

State space $\Omega$: all perfect and near-perfect matchings of $G$

Markov chain (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w=u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$
**Input**: a graph $G$

**State space** $\Omega$: all perfect and near-perfect matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w=u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$
Input: a graph $G$

State space $\Omega$: all perfect and near-perfect matchings of $G$

Markov chain (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w = u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$
A Markov Chain for Perfect Matchings

**Input:** a graph $G$

**State space** $\Omega$: all perfect and near-perfect matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$’s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w=u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$’s current edge by $(u,w)$ or $(v,w)$
A Markov Chain for Perfect Matchings

**Input**: a graph \( G \)

**State space** \( \Omega \): all perfect and near-perfect matchings of \( G \)

**Markov chain** (slide chain):
Let \( M \) be the current matching, we get the next state by choosing a random vertex \( w \) and:

- if \( M \) is perfect: remove \( w \)'s edge
- if \( M \) is near-perfect with holes \( u,v \):
  - if \( w=u \) or \( v \), add \((u,v)\) if can
  - else, randomly choose \( u \) or \( v \), replace \( w \)'s current edge by \((u,w)\) or \((v,w)\)
**A Markov Chain for Perfect Matchings**

**Input:** a graph $G$

**State space** $\Omega$: all perfect and near-perfect matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w = u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$
A Markov Chain for Perfect Matchings

**Input**: a graph $G$

**State space** $\Omega$: all perfect and near-perfect matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w=u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$

![Diagram of a graph with vertices and edges, showing the transition from one state to another through a random choice of vertex and edge modification.](attachment:image.png)
A Markov Chain for Perfect Matchings

**Input:** a graph $G$

**State space $\Omega$:** all perfect and near-perfect matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w=u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$
A Markov Chain for Perfect Matchings

**Input:** a graph $G$

**State space $\Omega$:** all perfect and near-perfect matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w=u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$
A Markov Chain for Perfect Matchings

**Input:** a graph $G$

**State space $\Omega$:** all perfect and near-perfect matchings of $G$

**Markov chain** (slide chain):
Let $M$ be the current matching, we get the next state by choosing a random vertex $w$ and:

- if $M$ is perfect: remove $w$'s edge
- if $M$ is near-perfect with holes $u,v$:
  - if $w=u$ or $v$, add $(u,v)$ if can
  - else, randomly choose $u$ or $v$, replace $w$'s current edge by $(u,w)$ or $(v,w)$
Analysis of this MC:

- canonical paths as before for perfect to perfect
- for near to perfect:
  - exactly one alternating path - process it last
A Markov Chain for Perfect Matchings

Analysis of this MC:

- canonical paths as before for perfect to perfect
- for near to perfect:
  - exactly one alternating path - process it last
Analysis of this MC:

- canonical paths as before for perfect to perfect
- for near to perfect:
  exactly one alternating path – process it last

A Markov Chain for Perfect Matchings
Analysis of this MC:

- canonical paths as before for perfect to perfect
- for near to perfect:
  exactly one alternating path - process it last
A Markov Chain for Perfect Matchings

Analysis of this MC:

- canonical paths as before for perfect to perfect
- for near to perfect:
  exactly one alternating path – process it last

Want to process last, otherwise 4 holes -> not in $\Omega$
A Markov Chain for Perfect Matchings

Analysis of this MC:

• canonical paths as before for perfect to perfect

• for near to perfect:
  exactly one alternating path - process it last

• for near to near:
  go through a random perfect matching
  (Instead of canonical paths, split into a flow.)
A Markov Chain for Perfect Matchings

Analysis of this MC:

• canonical paths as before for perfect to perfect

• for near to perfect:
  exactly one alternating path - process it last

• for near to near:
  go through a random perfect matching
  (Instead of canonical paths, split into a flow.)

Mixing time: \[ t_{\text{mix}}(\epsilon) = O^*(n^3 \frac{\#\text{nears}}{\#\text{perfects}}) \]

Polynomial if \# near-perfect / \# perfect matchings is polynomial...
E.g. for dense graphs: every vertex of degree > n/2.
A Markov Chain for Perfect Matchings

What if improve mixing time analysis?

\[ t_{\text{mix}}(\varepsilon) = O^*(n^3 \frac{\#\text{nears}}{\#\text{perfects}}) \]

Mixing time: \[ t_{\text{mix}}(\varepsilon) = O^*(n^3 \frac{\#\text{nears}}{\#\text{perfects}}) \]

Polynomial if \# near-perfect / \# perfect matchings is polynomial... E.g. for dense graphs: every vertex of degree > n/2.
What if improve mixing time analysis?

Just one perfect matching...

**Mixing time:** $t_{\text{mix}}(\epsilon) = O^*(n^3 \ (#\text{nears}/#\text{perfects}))$

Polynomial if # near-perfect / # perfect matchings is polynomial...
E.g. for dense graphs: every vertex of degree $> n/2$. 
What if improve mixing time analysis?

Just one perfect matching... But exponentially many nears!

**Mixing time:** $t_{\text{mix}}(\epsilon) = O^*(n^3 (#\text{nears}/#\text{perfects}))$

Polynomial if # near-perfect / # perfect matchings is polynomial...
E.g. for dense graphs: every vertex of degree $> n/2$. 

A Markov Chain for Perfect Matchings
What if improve mixing time analysis?

Just one perfect matching... But exponentially many nears!

This MC good only if \#nears/\#perfects polynomial...

**Mixing time:** \( t_{\text{mix}}(\epsilon) = O^*(n^3 (\#\text{nears}/\#\text{perfects})) \)

Polynomial if \# near-perfect / \# perfect matchings is polynomial... E.g. for dense graphs: every vertex of degree > \( n/2 \).
Another MC for Perfect Matchings

**Input**: a graph $G$

**State space** $\Omega$: all perfect matchings of $G$

**Markov chain** (swap chain):
Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.
Input: a graph $G$

State space $\Omega$: all perfect matchings of $G$

Markov chain (swap chain):
Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.
Another MC for Perfect Matchings

**Input:** a graph $G$

**State space** $\Omega$: all perfect matchings of $G$

**Markov chain** (swap chain):
Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.
**Another MC for Perfect Matchings**

**Input**: a graph $G$

**State space** $\Omega$: all perfect matchings of $G$

**Markov chain** (swap chain):
Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.
Another MC for Perfect Matchings

**Input:** a graph $G$

**State space $\Omega$:** all perfect matchings of $G$

**Markov chain** (swap chain): Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.
Another MC for Perfect Matchings

**Input**: a graph $G$

**State space** $\Omega$: all perfect matchings of $G$

**Markov chain** (swap chain):
Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.
**Another MC for Perfect Matchings**

**Input**: a graph $G$

**State space** $\Omega$: all perfect matchings of $G$

**Markov chain** (swap chain):
Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.

Symmetric but state space disconnected...
**Another MC for Perfect Matchings**

**Input**: a graph $G$

**State space** $\Omega$**: all perfect matchings of $G$**

**Markov chain** (swap chain):
Let $M$ be the current matching, choose two random edges $(u,v)$ and $(x,y)$ in $M$, replace them with $(u,y)$ and $(v,x)$ if can.

What if have an instance with connected state space:

Does it then mix rapidly?
Another MC for Perfect Matchings

Consider this instance (family of instances):

![Graph Image]
Consider this instance (family of instances):

From any matching can get to
Another MC for Perfect Matchings

Consider this instance (family of instances):

From any matching can get to
Another MC for Perfect Matchings

Consider this instance (family of instances):

From any matching can get to
Another MC for Perfect Matchings

Consider this instance (family of instances):

From any matching can get to
Another MC for Perfect Matchings

Consider this instance (family of instances):

From any matching can get to
Consider this instance (family of instances):

From any matching can get to

→ State space is connected
Another MC for Perfect Matchings

Consider this instance (family of instances):

- # matchings that do not use the bottom edge: $\geq 2^{n/4-1}$
- # matchings that use the bottom edge: 1
Another MC for Perfect Matchings

Consider this instance (family of instances):
Another MC for Perfect Matchings

Consider this instance (family of instances):

\[
\Phi := \min_{S: S \subseteq \Omega, \pi(S) \leq 1/2} \sum_{x \in S, y \notin S} \frac{\pi(x)P(x, y)}{\pi(S)} \leq \frac{1}{|\Omega|} \frac{1}{2n(n-1)} \leq \frac{1}{2^{n/2-1}n(n-1)}
\]
Another MC for Perfect Matchings

Consider this instance (family of instances):

\[ \Phi := \min_{S : S \subseteq \Omega, \pi(S) \leq 1/2} \sum_{x \in S, y \notin S} \frac{\pi(x)P(x, y)}{\pi(S)} \leq \frac{1}{|\Omega|} \frac{1}{2n(n-1)} \leq \frac{1}{2^{n/2-1}} \frac{1}{n(n-1)} \]
Consider this instance (family of instances):

$S$

**Conductance:**

$$t_{mix}(\varepsilon) \geq \frac{1}{2} \left( \frac{1}{2\Phi} - 1 \right) \log \left( \frac{1}{2\varepsilon} \right) \geq \left( 2^{n/2^{3}} n(n-1) - \frac{1}{2} \right) \log \left( \frac{1}{2\varepsilon} \right)$$
Another MC for Perfect Matchings

Consider this instance (family of instances):

\[
\begin{align*}
S = & \begin{cases}
\text{More on this chain:} & \\
\text{Dyer-Jerrum-Müller}
\end{cases}
\end{align*}
\]

**Conductance:**

\[
t_{mix}(\varepsilon) \geq \frac{1}{2} \left( \frac{1}{2\Phi} - 1 \right) \log \left( \frac{1}{2\varepsilon} \right) \geq \left( 2^{n/2-3} n(n-1) - \frac{1}{2} \right) \log \left( \frac{1}{2\varepsilon} \right)
\]
What if improve mixing time analysis?

• canonical paths as before for perfect to perfect
• for near to perfect: exactly one alternating path – process it last

Just one perfect matching...
But exponentially many nears!

Back to the Sliding Chain: Permanent

Per(A) = \sum_{\pi \in S_n} \prod_{i=1}^{n} a_{i,\pi(i)}

Counts perfect matchings in bipartite graphs

State space

Exponentially smaller!

Perfect matchings
Back to the Sliding Chain: Permanent

Idea [Jerrum-Sinclair-Vigoda]: Change the weights of the states (change stationary distribution).

State space

Exponentially smaller!

Perfect matchings
Back to the Sliding Chain: Permanent

Idea [Jerrum-Sinclair-Vigoda]:
Change the weights of the states (change stationary distribution).

$n^2+1$ regions, very different weight

Exponentially smaller!

Perfect matchings
Back to the Sliding Chain: Permanent

Idea [Jerrum-Sinclair-Vigoda]:
Change the weights of the states
(change stationary distribution).

Ideal weights
(for a matching with holes u,v):

$$\frac{\text{(# perfects})}{\text{(# nears with holes u,v)}}$$
Ideal weights
(for a matching with holes u,v):

\[
\frac{\# \text{ perfects}}{\# \text{ nears with holes } u,v}
\]

How to compute ???
Ideal weights
(for a matching with holes $u,v$):

\[
\frac{\text{(\# perfects)}}{\text{(\# nears with holes $u,v$)}}
\]

How to compute ???

Approximate:
start with an easy graph,
gradually get to the target graph
Ideal weights
(for a matching with holes u,v):

\[
\frac{\# \text{ perfects}}{\# \text{ nears with holes u,v}}
\]

How to compute ???

**Approximate:**
start with an easy graph,
gradually get to the target graph
Ideal weights
(for a matching with holes u,v):

\[
\frac{\text{(# perfects)}}{\text{( # nears with holes u,v)}}
\]

How to compute ???

Approximate:
start with an easy graph,
gradually get to the target graph
Ideal weights
(for a matching with holes $u,v$):

$$\frac{\text{(# perfects)}}{\text{( # nears with holes } u,v\text{)}}$$

How to compute ???

**Approximate:**
start with an easy graph,
gradually get to the target graph
Ideal weights
(for a matching with holes u,v):

(# perfects) / (# nears with holes u,v)

How to compute ???

Approximate:
start with an easy graph,
gradually get to the target graph
Back to the Sliding Chain: Permanent

Ideal weights
(for a matching with holes \( u,v \)):

\[
\frac{\# \text{ perfects}}{\# \text{ nears with holes } u,v}
\]

How to compute ???

Approximate:
start with an easy graph,
gradually get to the target graph
Back to the Sliding Chain: Permanent

Ideal weights
(for a matching with holes u,v):

\[(\# \text{ perfects}) / (\# \text{ nears with holes u,v})\]

How to compute ???

Approximate:
start with an easy graph,
gradually get to the target graph
Ideal weights
(for a matching with holes u,v):

\[
\frac{\text{(# perfects)}}{\text{(# nears with holes u,v)}}
\]

Edge weights:
• 1 for edge
• \(\lambda\) for non-edge

• Start with \(\lambda = 1\):

\[
\text{#perfect/#nears} = \frac{n!}{(n-1)!}
\]
Back to the Sliding Chain: Permanent

**Ideal weights**
(for a matching with holes u,v):

\[ \lambda(\text{perfects}) / \lambda(\text{nears with holes u,v}) \]

Edge weights:
- 1 for edge
- \( \lambda \) for non-edge

- Start with \( \lambda = 1 \):
  \[ \frac{\#\text{perfect}}{\#\text{nears}} = \frac{n!}{(n-1)!} \]

- Repeat until \( \lambda < 1/n! \):

\[ \lambda \text{ and 4-apx of weights} \]

\[ \lambda \text{ and 2-apx of weights} \]

2-apx = 4-apx for new \( \lambda \)
**Thm [Jerrum-Sinclair-Vigoda]:**
FPRAS for the permanent.

**OPEN PROBLEM:**
counting perfect matchings in non-bipartite graphs

\[
\lambda(\text{perfects}) / \lambda(\text{nears with holes } u,v)
\]