The Matching Problem is in Quasi-NC

Ola Svensson and Jakub Tarnawski

Institute for Advanced Study, 22.01.2018
Perfect matching problem

Given a graph, can we pair up all vertices using edges?
Perfect matching problem

Given a graph, can we pair up all vertices using edges?

very tough instance: graph is non-bipartite!
Perfect matching problem

Given a graph, can we pair up all vertices using edges?

very tough instance: graph is non-bipartite!
Perfect matching problem

Benchmark problem in computer science
Perfect matching problem

Benchmark problem in computer science

Algorithms:

- bipartite: Jacobi [XIX century, weighted!]
- general: Edmonds [1965]
  - polynomial-time = efficient
- since then, tons of research and still active
- many models of computation: monotone circuits, extended formulations, parallel, streaming/sublinear, ...
Perfect matching problem

Benchmark problem in computer science

Algorithms:

▶ bipartite: Jacobi [XIX century, weighted!]

▶ general: Edmonds [1965]
  ▶ polynomial-time = efficient

▶ since then, tons of research and still active

▶ many models of computation: monotone circuits, extended formulations, parallel, streaming/sublinear, ...
Parallel complexity

**Class \( \mathcal{NC} \):** problems that parallelize completely

- **poly \( n \) processors**
- **poly log \( n \) time**

Main open question: is matching in \( \mathcal{NC} \)?
Parallel complexity

**Class** $\mathcal{NC}$: problems that parallelize completely

- poly $n$ processors
  - poly log $n$ time

**Main open question:** is matching in $\mathcal{NC}$?
Parallel complexity

**Class $\mathcal{NC}$:** problems that parallelize completely

- poly $n$ processors
- poly log $n$ time

*It’s in Randomized $\mathcal{NC}$*

Main open question: is matching in $\mathcal{NC}$?
Matching is in \textsc{Randomized} \textit{NC} [Lovász 1979]:
has \textit{randomized} algorithm that uses:

\begin{itemize}
  \item polynomially many processors
  \item polylog time
\end{itemize}
Matching is in \textit{Randomized }\mathcal{NC} \ [\text{Lovász 1979}]:
has \textit{randomized} algorithm that uses:

\begin{itemize}
  \item polynomialsly many processors
  \item polylog time
\end{itemize}

Search version in \textit{Randomized }\mathcal{NC}:

\begin{itemize}
  \item [\text{Karp, Upfal, Wigderson 1986}]
  \item [\text{Mulmuley, Vazirani, Vazirani 1987}]
\end{itemize}
Matching is in Randomized $\mathcal{NC}$ [Lovász 1979]: has randomized algorithm that uses:

- polynomially many processors
- polylog time

Search version in Randomized $\mathcal{NC}$:

- [Karp, Upfal, Wigderson 1986]
- [Mulmuley, Vazirani, Vazirani 1987]

Can we derandomize efficient computation?
Matching is in \textbf{Randomized NC} [Lovász 1979]: has randomized algorithm that uses:

- polynomially many processors
- polylog time

Search version in \textbf{Randomized NC}:

- [Karp, Upfal, Wigderson 1986]
- [Mulmuley, Vazirani, Vazirani 1987]

Can we derandomize efficient computation?

Can we derandomize one of these algorithms?
Is matching in $\mathcal{NC}$?

- For restricted graph classes:
  - Bipartite regular [Lev, Pippenger, Valiant 1981]
  - Bipartite convex [Dekel, Sahni 1984]
  - Incomparability graphs [Kozen, Vazirani, Vazirani 1985]
  - Bipartite graphs with small number of perfect matchings [Grigoriev, Karpinski 1987]
  - Claw-free [Chrobak, Naor, Novick 1989]
  - $K_{3,3}$-free (decision version) [Vazirani 1989]
  - Planar bipartite [Miller, Naor 1989]
  - Dense [Dahlhaus, Hajnal, Karpinski 1993]
  - Strongly chordal [Dahlhaus, Karpinski 1998]
  - $P_4$-tidy [Parfenoff 1998]
  - Bipartite small genus [Mahajan, Varadarajan 2000]
  - Graphs with small number of perfect matchings [Agrawal, Hoang, Thierauf 2006]
  - Planar (search version) [Anari, Vazirani 2017]

- But not known for:
  - Bipartite
Is matching in $\mathcal{NC}$?

Yes, for restricted graph classes:

- bipartite regular [Lev, Pippenger, Valiant 1981]
- bipartite convex [Dekel, Sahni 1984]
- incomparability graphs [Kozen, Vazirani, Vazirani 1985]
- bipartite graphs with small number of perfect matchings [Grigoriev, Karpinski 1987]
- claw-free [Chrobak, Naor, Novick 1989]
- $K_{3,3}$-free (decision version) [Vazirani 1989]
- planar bipartite [Miller, Naor 1989]
- dense [Dahlhaus, Hajnal, Karpinski 1993]
- strongly chordal [Dahlhaus, Karpinski 1998]
- $P_4$-tidy [Parfenoff 1998]
- bipartite small genus [Mahajan, Varadarajan 2000]
- graphs with small number of perfect matchings [Agrawal, Hoang, Thierauf 2006]
- planar (search version) [Anari, Vazirani 2017]
Is matching in $\mathcal{NC}$?

Yes, for restricted graph classes:

- bipartite regular [Lev, Pippenger, Valiant 1981]
- bipartite convex [Dekel, Sahni 1984]
- incomparability graphs [Kozen, Vazirani, Vazirani 1985]
- bipartite graphs with small number of perfect matchings [Grigoriev, Karpinski 1987]
- claw-free [Chrobak, Naor, Novick 1989]
- $K_{3,3}$-free (decision version) [Vazirani 1989]
- planar bipartite [Miller, Naor 1989]
- dense [Dahlhaus, Hajnal, Karpinski 1993]
- strongly chordal [Dahlhaus, Karpinski 1998]
- $P_4$-tidy [Parfenoff 1998]
- bipartite small genus [Mahajan, Varadarajan 2000]
- graphs with small number of perfect matchings [Agrawal, Hoang, Thierauf 2006]
- planar (search version) [Anari, Vazirani 2017]

but not known for:

- bipartite
Theorem

Fenner, Gurjar and Thierauf [2015]

Bipartite matching is in QUASI-NC

\((n^{\text{poly log } n \text{ processors, poly log } n \text{ time, deterministic}})\)
**Theorem**  

Fenner, Gurjar and Thierauf [2015]

Bipartite matching is in **QUASI-NC**

\( n^{\text{poly log } n} \) processors, \( \text{poly log } n \) time, deterministic

- Approach fails for non-bipartite graphs

\[ \begin{array}{ccc}
\bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet \\
\end{array} \]  
much harder than  

\[ \begin{array}{cc}
\bullet & \bullet \\
\bullet & \bullet \\
\end{array} \]
Theorem S. and Tarnawski [2017]

**General** matching is in **QUASI-NC**

\( (n^{\text{poly log } n} \text{ processors, poly log } n \text{ time, deterministic}) \)
Theorem  S. and Tarnawski [2017]

**General matching is in QUASI-\(\mathcal{NC}\)**

\((n^{poly \log n} \text{ processors, } poly \log n \text{ time, deterministic})\)

with quasi-polynomial \# processors
Outline

1. Basic approach for derandomization
2. Bipartite case [Fenner, Gurjar, Thierauf 2015]
3. Difficulties of general case & our approach
Basic approach for derandomization
Basic approach for derandomization

(Derandomize one of the randomized algorithms)
Algorithm of Mulmuley, Vazirani, Vazirani’87
Algorithm

1. For each edge $e$ select weight $w(e) \in \{1, 2, \ldots, n^2\}$ at random
2. Calculate determinant of Tutte matrix where $X_e$ is replaced by $2^{w(e)}$
Algorithm

1. For each edge $e$ select weight $w(e) \in \{1, 2, \ldots, n^2\}$ at random
2. Calculate determinant of Tutte matrix where $X_e$ is replaced by $2^{w(e)}$

*Important that $w$ is polynomially bounded*
Algorithm of Mulmuley, Vazirani, Vazirani’87

Algorithm

1. For each edge \(e\) select weight \(w(e) \in \{1, 2, \ldots, n^2\}\) at random

2. Calculate determinant of Tutte matrix where \(X_e\) is replaced by \(2^{w(e)}\)

Important that \(w\) is polynomially bounded
**Algorithm**

1. For each edge \( e \) select weight \( w(e) \in \{1, 2, \ldots, n^2\} \) at random.

2. Calculate determinant of Tutte matrix where \( X_e \) is replaced by \( 2^{w(e)} \).

*Important that \( w \) is polynomially bounded*
Algorithm of Mulmuley, Vazirani, Vazirani’87

Algorithm

1. For each edge $e$ select weight $w(e) \in \{1, 2, \ldots, n^2\}$ at random
2. Calculate determinant of Tutte matrix where $X_e$ is replaced by $2^{w(e)}$

Important that $w$ is polynomially bounded

Step 2 guaranteed to work if weight function $w$ is isolating: unique min-weight matching
Algorithm

1. For each edge \( e \) select weight \( w(e) \in \{1, 2, \ldots, n^2\} \) at random
2. Calculate determinant of Tutte matrix where \( X_e \) is replaced by \( 2^{w(e)} \)

*Important that \( w \) is polynomially bounded*

### random sampling (Step 1)

**Isolation Lemma:**
\[
\Pr[w \text{ isolating}] \geq 0.9
\]

Step 2 guaranteed to work if weight function \( w \) is **isolating**: unique min-weight matching
Algorithm of Mulmuley, Vazirani, Vazirani'87

Algorithm

1. For each edge $e$ select weight $w(e) \in \{1, 2, \ldots, n^2\}$ at random

2. Calculate determinant of Tutte matrix where $X_e$ is replaced by $2^{w(e)}$

Important that $w$ is polynomially bounded

**Isolation Lemma:**

$$\Pr[w \text{ isolating}] \geq 0.9$$

Step 2 guaranteed to work if weight function $w$ is **isolating**: unique min-weight matching

random sampling (Step 1)

something deterministic?

Construct isolating $w$ in $NC$?
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $NC$.
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
2. For $w \in \mathcal{W}^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
3. The number of weight functions are polynomial $|\mathcal{W}^*| \leq \text{poly}(n)$

The oblivious algorithm simply checks all weight functions in parallel.
**Oblivious derandomization**

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
2. For $w \in \mathcal{W}^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
2. For $w \in \mathcal{W}^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
3. The number of weight functions are polynomial $|\mathcal{W}^*| \leq \text{poly}(n)$
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
2. For $w \in \mathcal{W}^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
3. The number of weight functions are polynomial $|\mathcal{W}^*| \leq \text{poly}(n)$

*The oblivious algorithm simply checks all weight functions in parallel*
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $NC$

**Oblivious challenge:** On input $n$, construct a family $W^*$ of weight functions that can be computed in $NC$ such that

1. For any $n$-vertex graph, there is an isolating $w \in W^*$
2. For $w \in W^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
3. The number of weight functions are polynomial $|W^*| \leq \text{poly}(n)$

*The oblivious algorithm simply checks all weight functions in parallel*
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $NC$

**Oblivious challenge:** On input $n$, construct a family $W^*$ of weight functions that can be computed in $NC$ such that

1. For any $n$-vertex graph, there is an isolating $w \in W^*$
2. For $w \in W^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
3. The number of weight functions are polynomial $|W^*| \leq \text{poly}(n)$

*The oblivious algorithm simply checks all weight functions in parallel*

Easy even with $|W^*| \leq 1$
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
2. For $w \in \mathcal{W}^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
3. The number of weight functions are polynomial $|\mathcal{W}^*| \leq \text{poly}(n)$

*The oblivious algorithm simply checks all weight functions in parallel*
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
2. For $w \in \mathcal{W}^*$ and edge $e$, we have $w(e) \leq \text{poly}(n)$
3. The number of weight functions are polynomial $|\mathcal{W}^*| \leq \text{poly}(n)$

*The oblivious algorithm simply checks all weight functions in parallel*

Easy, but best known bound on $|\mathcal{W}^*|$ is exponential in $n$
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $NC$

**Oblivious challenge:** On input $n$, construct a family $W^*$ of weight functions that can be computed in $NC$ such that

1. For any $n$-vertex graph, there is an isolating $w \in W^*$
2. For $w \in W^*$ and edge $e$, we have $w(e) \leq n^{\text{poly}(\log n)}$
3. The number of weight functions are polynomial $|W^*| \leq n^{\text{poly}(\log n)}$

*The oblivious algorithm simply checks all weight functions in parallel*
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}$
2. For $w \in \mathcal{W}$ and edge $e$, we have $w(e) \leq n^{\text{poly}(\log n)}$
3. The number of weight functions are polynomial $|\mathcal{W}| \leq n^{\text{poly}(\log n)}$

*The oblivious algorithm simply checks all weight functions in parallel*

Thm[FGT’15]: $\mathcal{W}$ exists for bipartite graphs
Oblivious derandomization

**Challenge:** On input $G$, construct an isolating weight function in $\mathcal{NC}$

**Oblivious challenge:** On input $n$, construct a family $\mathcal{W}^*$ of weight functions that can be computed in $\mathcal{NC}$ such that

1. For any $n$-vertex graph, there is an isolating $w \in \mathcal{W}^*$
2. For $w \in \mathcal{W}^*$ and edge $e$, we have $w(e) \leq n^{\text{poly}(\log n)}$
3. The number of weight functions are polynomial $|\mathcal{W}^*| \leq n^{\text{poly}(\log n)}$

---

The oblivious algorithm simply checks all weight functions in parallel

**Thm[FGT’15]:** $\mathcal{W}^*$ exists for bipartite graphs

**Thm[ST’17]:** $\mathcal{W}^*$ exists for general graphs
Bipartite case

[Fenner, Gurjar, Thierauf 2015]
Bipartite case

[Fenner, Gurjar, Thierauf 2015]
“Greed is good. Greed is right. Greed works. Greed clarifies, cuts through and captures the essence of the evolutionary spirit.”

- Gordon Gecko

Bipartite case

[Fenner, Gurjar, Thierauf 2015]
Bipartite case

[Fenner, Gurjar, Thierauf 2015]
Make progress step-by-step

Construct isolating function iteratively

\[
W = \{ w_k : w_k(e_i) = 2i \text{ mod } k \text{ for } k = 2, 3, \ldots, n^4 \}
\]

Let \( w_1 \in W \) and let \( M_1 \) be perfect matchings minimizing \( w_1 \)

Let \( w_2 \in W \) and let \( M_2 \subseteq M_1 \) be PMs in \( M_1 \) minimizing \( w_2 \)

Let \( w_3 \in W \) and let \( M_3 \subseteq M_2 \) be PMs in \( M_2 \) minimizing \( w_3 \)

...
Make progress step-by-step

**Construct isolating function iteratively**

Let $\mathcal{W} = \{ w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4 \}$ be a polynomial set of simple weight functions.
Make progress step-by-step

Construct isolating function iteratively

Let $W = \{ w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4 \}$ be a polynomial set of simple weight functions.
Construct isolating function iteratively

Let \( \mathcal{W} = \{ w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4 \} \) be a polynomial set of simple weight functions.

- Select \( w_1 \in \mathcal{W} \) and let \( \mathcal{M}_1 \) be perfect matchings minimizing \( w_1 \).
Construct isolating function iteratively

Let $\mathcal{W} = \{w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4\}$ be a polynomial set of simple weight functions

- Select $w_1 \in \mathcal{W}$ and let $\mathcal{M}_1$ be perfect matchings minimizing $w_1$
- Select $w_2 \in \mathcal{W}$ and let $\mathcal{M}_2 \subseteq \mathcal{M}_1$ be PMs in $\mathcal{M}_1$ minimizing $w_2$
Make progress step-by-step

Construct isolating function iteratively

Let \( \mathcal{W} = \{ w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4 \} \) be a polynomial set of simple weight functions

- Select \( w_1 \in \mathcal{W} \) and let \( \mathcal{M}_1 \) be perfect matchings minimizing \( w_1 \)
- Select \( w_2 \in \mathcal{W} \) and let \( \mathcal{M}_2 \subseteq \mathcal{M}_1 \) be PMs in \( \mathcal{M}_1 \) minimizing \( w_2 \)
- Select \( w_3 \in \mathcal{W} \) and let \( \mathcal{M}_3 \subseteq \mathcal{M}_2 \) be PMs in \( \mathcal{M}_2 \) minimizing \( w_3 \)
  
  \[ \vdots \]
Construct isolating function iteratively

Let $W = \{w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4\}$ be a polynomial set of simple weight functions

- Select $w_1 \in W$ and let $M_1$ be perfect matchings minimizing $w_1$
- Select $w_2 \in W$ and let $M_2 \subseteq M_1$ be PMs in $M_1$ minimizing $w_2$
- Select $w_3 \in W$ and let $M_3 \subseteq M_2$ be PMs in $M_2$ minimizing $w_3$
  
  $\vdots$

**How many** $w_1, \ldots, w_\ell \in W$ **necessary for** $|M_\ell| = 1$?
Construct isolating function iteratively

Let \( \mathcal{W} = \{ w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4 \} \) be a polynomial set of simple weight functions

- Select \( w_1 \in \mathcal{W} \) and let \( \mathcal{M}_1 \) be perfect matchings minimizing \( w_1 \)

- Select \( w_2 \in \mathcal{W} \) and let \( \mathcal{M}_2 \subseteq \mathcal{M}_1 \) be PMs in \( \mathcal{M}_1 \) minimizing \( w_2 \)

- Select \( w_3 \in \mathcal{W} \) and let \( \mathcal{M}_3 \subseteq \mathcal{M}_2 \) be PMs in \( \mathcal{M}_2 \) minimizing \( w_3 \)

\[ \vdots \]

How many \( w_1, \ldots, w_\ell \in \mathcal{W} \) necessary for \( |\mathcal{M}_\ell| = 1 \)?

**Thm [FGT’15]:**

For any \( G \), there is \( w_1, \ldots, w_{\log_2(n)} \in \mathcal{W} \) so that \( |\mathcal{M}_{\log_2(n)}| = 1 \)
Make progress step-by-step

Construct isolating function iteratively

Let $\mathcal{W} = \{w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4\}$ be a polynomial set of simple weight functions

- Select $w_1 \in \mathcal{W}$ and let $\mathcal{M}_1$ be perfect matchings minimizing $w_1$
- Select $w_2 \in \mathcal{W}$ and let $\mathcal{M}_2 \subseteq \mathcal{M}_1$ be PMs in $\mathcal{M}_1$ minimizing $w_2$
- Select $w_3 \in \mathcal{W}$ and let $\mathcal{M}_3 \subseteq \mathcal{M}_2$ be PMs in $\mathcal{M}_2$ minimizing $w_3$
  
  $\vdots$

How many $w_1, \ldots, w_\ell \in \mathcal{W}$ necessary for $|\mathcal{M}_\ell| = 1$?

**Thm [FGT’15]:**

For any $G$, there is $w_1, \ldots, w_{\log_2(n)} \in \mathcal{W}$ so that $|\mathcal{M}_{\log_2(n)}| = 1$

\[ \downarrow \]

$\mathcal{W}^* = \{n^{9(\log(n))}w_1 + n^{9(\log(n)-1)}w_2 + \cdots + 1 \cdot w_{\log(n)} : w_1, \ldots, w_{\log_2(n)} \in \mathcal{W}\}$
gives oblivious quasi-polynomial derandomization
**GOAL:** For *any* $n$-vertex graph $G$, show that there is

$$w_1, \ldots, w_{\log n} \in \mathcal{W} = \{w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4\}$$

so that $|\mathcal{M}_{\log n}| = 1$
**GOAL:** For any $n$-vertex graph $G$, show that there is

$$w_1, \ldots, w_{\log n} \in \mathcal{W} = \{w_k : w_k(e_i) = 2^i \mod k \text{ for } k = 2, 3, \ldots, n^4\}$$

so that $|\mathcal{M}_{\log n}| = 1$

We need good progress measure
Consider min-weight perfect matchings $M, M'$ with $w(M) = w(M')$.
Consider min-weight perfect matchings \( M, M' \) with \( w(M) = w(M') \).
Minimum perfect matchings of the same weight

- Consider min-weight perfect matchings \( M, M' \) with \( w(M) = w(M') \)

\[
\begin{align*}
\text{Progress: assign, 0 discrepancy to "many" cycles}
\end{align*}
\]
Consider min-weight perfect matchings $M, M'$ with $w(M) = w(M')$

Symmetric difference = alternating cycles

Define discrepancy of a cycle:
$$d_w(C) := w(e_1) - w(e_2) + w(e_3) - w(e_4)$$

If $\forall C d_w(C) = 0$, then $w$ isolating!

Progress: assign 0 discrepancy to "many" cycles
Consider min-weight perfect matchings \( M, M' \) with \( w(M) = w(M') \).

Symmetric difference

Alternating cycles

In each cycle \( C \),

\[ w(M \cap C) = w(M' \cap C) \]

(otherwise could get lighter matching)
Minimum perfect matchings of the same weight

- Consider min-weight perfect matchings $M, M'$ with $w(M) = w(M')$
- symmetric difference
  $\Rightarrow$ alternating cycles
- in each cycle $C$,
  \[ w(M \cap C) = w(M' \cap C) \]
  (otherwise could get lighter matching)

Define discrepancy of a cycle $C$:
\[ d_w(C) := w(e_1) - w(e_2) + w(e_3) - w(e_4) \]

If $\forall C \, d_w(C) = 0$, then $w$ isolating!

Progress: assign $0$ discrepancy to "many" cycles
Consider min-weight perfect matchings $M, M'$ with $w(M) = w(M')$

- symmetric difference $= \text{alternating cycles}$

- in each cycle $C$,
  $w(M \cap C) = w(M' \cap C)$
  (otherwise could get lighter matching)

- define discrepancy of a cycle:
  $d_w(C) := w(e_1) - w(e_2) + w(e_3) - w(e_4)$
Consider min-weight perfect matchings $M, M'$ with $w(M) = w(M')$

- symmetric difference
  $= \text{alternating cycles}$

- in each cycle $C$,
  $w(M \cap C) = w(M' \cap C)$
  (otherwise could get lighter matching)

- define **discrepancy** of a cycle:
  $d_w(C) := w(e_1) - w(e_2) + w(e_3) - w(e_4)$

- $d_w(C) = 0$
Consider min-weight perfect matchings $M, M'$ with $w(M) = w(M')$

- symmetric difference
  $= \text{alternating cycles}$
  in each cycle $C$
  $w(M \cap C) = w(M' \cap C)$
  (otherwise could get lighter matching)

- define discrepancy of a cycle:
  \begin{align*}
d_w(C) &:= w(e_1) - w(e_2) + w(e_3) - w(e_4) \\
d_w(C) &= 0
\end{align*}

If $\forall C \ d_w(C) \neq 0$, then $w$ isolating!
Consider min-weight perfect matchings $M, M'$ with $w(M) = w(M')$

- symmetric difference
  - alternating cycles
  - in each cycle $C$, $w(M \cap C) = w(M' \cap C)$
    (otherwise could get lighter matching)
- define discrepancy of a cycle:
  $$d_w(C) := w(e_1) - w(e_2) + w(e_3) - w(e_4)$$
- $d_w(C) = 0$

If $(\forall C) d_w(C) \neq 0$, then $w$ isolating!

**Progress:** assign $\neq 0$ discrepancy to “many” cycles
Removing cycles

A graph may have exponentially many cycles ⇒ seems hard to find $w$ so that all of them have non-zero discrepancy.
Removing cycles

A graph may have exponentially many cycles \( \Rightarrow \) seems hard to find \( w \) so that all of them have non-zero discrepancy

Don’t be greedy!

**Old Lemma:**

For any collection of \( n^4 \) cycles, some \( w \in \mathcal{W} \) assigns all of them \( \neq 0 \) discrepancy
Removing cycles

A graph may have exponentially many cycles ⇒ seems hard to find $w$ so that all of them have non-zero discrepancy

Don’t be greedy!

Old Lemma:

For any collection of $n^4$ cycles, some $w \in \mathcal{W}$ assigns all of them $\neq 0$ discrepancy

If $\leq n^4$ cycles in the graph: done!
A graph may have exponentially many cycles $\Rightarrow$ seems hard to find $w$ so that all of them have non-zero discrepancy

Don’t be greedy!

Old Lemma:

For any collection of $n^4$ cycles, some $w \in \mathcal{W}$ assigns all of them $\neq 0$ discrepancy

If $\leq n^4$ cycles in the graph: done!

Not so easy, but we can cope with all 4-cycles
Select $w_1 \in \mathcal{W}$ so that all 4-cycles have $\neq 0$ discrepancy.
Select \( w_1 \in \mathcal{W} \) so that all 4-cycles have \( \neq 0 \) discrepancy.

What can we say about the active subgraph \( G_1 \) that contains those edges that are in a min-weight perfect matching?
Select $w_1 \in \mathcal{W}$ so that all 4-cycles have $\neq 0$ discrepancy

What can we say about the active subgraph $G_1$ that contains those edges that are in a min-weight perfect matching?
Select $w_1 \in \mathcal{W}$ so that all 4-cycles have $\neq 0$ discrepancy.
Select $w_1 \in \mathcal{W}$ so that all 4-cycles have $\neq 0$ discrepancy.

All matchings of $G$

$\mathcal{M}_1 = \{M, M'\}$
Select \( w_1 \in \mathcal{W} \) so that all 4-cycles have \( \neq 0 \) discrepancy.

All matchings of \( G \)

\[ \mathcal{M}_1 = \{ M, M' \} \]

\[ G_1 = (V, \bigcup_{M \in \mathcal{M}_1} M) \]
Select $w_1 \in \mathcal{W}$ so that all 4-cycles have $\neq 0$ discrepancy.

All matchings of $G$

$\mathcal{M}_1 = \{M, M'\}$

$G_1 = (\mathcal{V}, \cup_{M \in \mathcal{M}_1} M)$

What can we say about the active subgraph $G_1$ that contains those edges that are in a min-weight perfect matching?
Bipartite key property: Once we assign a cycle \( \neq 0 \) discrepancy, it will disappear from the active subgraph.
Bipartite key property: Once we assign a cycle $\neq 0$ discrepancy, it will disappear from the active subgraph.
**Bipartite key property:** Once we assign a cycle $\neq 0$ discrepancy, it will disappear from the active subgraph.
**Bipartite key property:** Once we assign a cycle \( \neq 0 \) discrepancy, it will disappear from the active subgraph

\[
d_w(C_1) = 1 \neq 0 \\
d_w(C_2) = 1 \neq 0
\]
Bipartite key property: Once we assign a cycle $\neq 0$ discrepancy, it will disappear from the active subgraph.

\[ d_w(C_1) = 1 \neq 0 \]
\[ d_w(C_2) = 1 \neq 0 \]
**Bipartite key property:** Once we assign a cycle $\neq 0$ discrepancy, it will disappear from the active subgraph

\[
d_w(C_1) = 1 \neq 0 \\
d_w(C_2) = 1 \neq 0
\]
**Bipartite key property:** Once we assign a cycle \( \neq 0 \) discrepancy, it will disappear from the active subgraph

**Proof:** Let \( \mathcal{M} \) be the set of perfect matchings minimizing \( w \)

- Consider the convex hull of \( \mathcal{M} \) (face \( F \) of the bipartite matching polytope):

\[
\text{PM} : \text{perfect matching polytope (convex hull of matchings)}
\]
**Bipartite key property:** Once we assign a cycle \( \neq 0 \) discrepancy, it will disappear from the active subgraph.

**Proof:** Let \( \mathcal{M} \) be the set of perfect matchings minimizing \( w \).

- Consider the convex hull of \( \mathcal{M} \) (face \( F \) of the bipartite matching polytope):

\[
PM : \text{perfect matching polytope (convex hull of matchings)}
\]
Bipartite key property: Once we assign a cycle ≠ 0 discrepancy, it will disappear from the active subgraph

Proof: Let $\mathcal{M}$ be the set of perfect matchings minimizing $w$

- Consider the convex hull of $\mathcal{M}$ (face $F$ of the bipartite matching polytope):

PM : perfect matching polytope (convex hull of matchings)
**Bipartite key property:** Once we assign a cycle ≠ 0 discrepancy, it will disappear from the active subgraph

**Proof:** Let $\mathcal{M}$ be the set of perfect matchings minimizing $w$

- Consider the convex hull of $\mathcal{M}$ (face $F$ of the bipartite matching polytope):

$\mathbf{PM}$: perfect matching polytope (convex hull of matchings)

<table>
<thead>
<tr>
<th>Bipartite PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(\delta(v)) = 1$ for every $v \in V$</td>
</tr>
<tr>
<td>$x_e \geq 0$ for every $e \in E$</td>
</tr>
</tbody>
</table>
**Bipartite key property:** Once we assign a cycle $\neq 0$ discrepancy, it will disappear from the active subgraph

**Proof:** Let $\mathcal{M}$ be the set of perfect matchings minimizing $w$

- Consider the convex hull of $\mathcal{M}$ (face $F$ of the bipartite matching polytope):

$\mathcal{M}$: perfect matching polytope (convex hull of matchings)

<table>
<thead>
<tr>
<th>Bipartite PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(\delta(v)) = 1$ for every $v \in V$</td>
</tr>
<tr>
<td>$x_e \geq 0$ for every $e \in E$</td>
</tr>
</tbody>
</table>

$F$ is simply a subgraph
**Bipartite key property:** Once we assign a cycle \( \neq 0 \) discrepancy, it will disappear from the active subgraph

**Proof:** Let \( \mathcal{M} \) be the set of perfect matchings minimizing \( w \)

- Consider the convex hull of \( \mathcal{M} \) (face \( F \) of the bipartite matching polytope):

  \[ F \]  

  **PM**: perfect matching polytope (convex hull of matchings)

  \[ x(\delta(v)) = 1 \quad \text{for every } v \in V \]

  \[ x_e \geq 0 \quad \text{for every } e \in E \]

  *F is simply a subgraph*

- What can we say about the weight of points in \( F \)?
**Bipartite key property:** Once we assign a cycle \( \neq 0 \) discrepancy, it will disappear from the active subgraph

**Proof:** Let \( \mathcal{M} \) be the set of perfect matchings minimizing \( w \)

- Consider the convex hull of \( \mathcal{M} \) (face \( F \) of the bipartite matching polytope):

\[
F \quad \text{PM} : \text{perfect matching polytope (convex hull of matchings)}
\]

<table>
<thead>
<tr>
<th>Bipartite PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x(\delta(v)) = 1 ) for every ( v \in V )</td>
</tr>
<tr>
<td>( x_e \geq 0 ) for every ( e \in E )</td>
</tr>
<tr>
<td>( F ) is simply a subgraph</td>
</tr>
</tbody>
</table>

- What can we say about the weight of points in \( F \)?

Every \( x, y \in F \) have same weight: \( \sum_e w(e)x_e = \sum_e w(e)y_e \)
$F$ is the convex hull of $\mathcal{M} \Rightarrow$ every $x, y \in F$ have same weight

$F$ is simply a subgraph

PM: perfect matching polytope (convex hull of matchings)

Bipartite PM

$x(\delta(v)) = 1$ for every $v \in V$

$x_e \geq 0$ for every $e \in E$
$F$ is the convex hull of $\mathcal{M} \Rightarrow$ every $x, y \in F$ have same weight

PM : perfect matching polytope (convex hull of matchings)

$F$ is simply a subgraph

Bipartite PM

\[
\begin{align*}
  x(\delta(v)) &= 1 \quad \text{for every } v \in V \\
  x_e &\geq 0 \quad \text{for every } e \in E
\end{align*}
\]

Suppose active subgraph has cycle $C$ of $\neq 0$ discrepancy

$w(\text{green edges}) \neq w(\text{red edges})$
$F$ is the convex hull of $\mathcal{M}$ $\Rightarrow$ every $x, y \in F$ have same weight

PM : perfect matching polytope (convex hull of matchings)

Bipartite PM

\[
\begin{aligned}
x(\delta(v)) &= 1 & \text{for every } v \in V \\
x_e &\geq 0 & \text{for every } e \in E
\end{aligned}
\]

$F$ is simply a subgraph

(\text{edge set } \bigcup_{M \in \mathcal{M}} M)

$\triangleright$ Suppose active subgraph has cycle $C$ of $\neq 0$ discrepancy

\[
C
\]

$w(\text{green edges}) \neq w(\text{red edges})$

$\triangleright$ Let $x = \frac{1}{|\mathcal{M}|} \sum_{M \in \mathcal{M}} 1_M$ be the mean of the face $F$
\( F \) is the convex hull of \( \mathcal{M} \) \( \Rightarrow \) every \( x, y \in F \) have same weight

PM : perfect matching polytope (convex hull of matchings)

Bipartite PM

\[
\begin{align*}
x(\delta(v)) &= 1 \quad \text{for every } v \in V \\
x_e &\geq 0 \quad \text{for every } e \in E
\end{align*}
\]

\( F \) is simply a subgraph

(\text{edge set } \bigcup_{M \in \mathcal{M}} M)

▶ Suppose active subgraph has cycle \( C \) of \( \neq 0 \) discrepancy

\[
 x = \frac{1}{|\mathcal{M}|} \sum_{M \in \mathcal{M}} 1_M \quad \text{be the mean of the face } F
\]

▶ Let \( x_e > 0 \) for every \( e \in C \) (since support of \( x \) equals \( \bigcup_{M \in \mathcal{M}} M \))

▶ Increasing red edges while decreasing green maintain degrees

▶ So we obtain a new point \( y \in F \) of different weight; contradiction
\( F \) is the convex hull of \( \mathcal{M} \) \( \Rightarrow \) every \( x, y \in F \) have same weight

\[
\begin{align*}
\text{PM} & : \text{perfect matching polytope (convex hull of matchings)} \\
\text{Bipartite PM} & \\
& x(\delta(v)) = 1 \quad \text{for every } v \in V \\
& x_e \geq 0 \quad \text{for every } e \in E \\
\end{align*}
\]

\( F \) is simply a subgraph

(\text{edge set } \cup_{M \in \mathcal{M}} M)

- Suppose active subgraph has cycle \( C \) of \( \neq 0 \) discrepancy

\[
\begin{align*}
\text{Let } x = \frac{1}{|\mathcal{M}|} \sum_{M \in \mathcal{M}} 1_M \text{ be the mean of the face } F \\
\text{Then } x_e > 0 \text{ for every } e \in C \quad (\text{since support of } x \text{ equals } \cup_{M \in \mathcal{M}} M) \\
\text{Increasing red edges while decreasing green maintain degrees}
\end{align*}
\]

\[
\begin{align*}
w(\text{green edges}) & \neq w(\text{red edges}) \\
\end{align*}
\]
$F$ is the convex hull of $\mathcal{M}$ $\Rightarrow$ every $x, y \in F$ have same weight

**Perfect Matching Polytope** (convex hull of matchings)

\[ x(\delta(v)) = 1 \quad \text{for every } v \in V \]
\[ x_e \geq 0 \quad \text{for every } e \in E \]

$F$ is simply a subgraph

Suppose active subgraph has cycle $C$ of $\neq 0$ discrepancy

\[ w(\text{green edges}) \neq w(\text{red edges}) \]

Let $x = \frac{1}{|\mathcal{M}|} \sum_{M \in \mathcal{M}} 1_M$ be the mean of the face $F$

Then $x_e > 0$ for every $e \in C$ (since support of $x$ equals $\bigcup_{M \in \mathcal{M}} M$)

Increasing red edges while decreasing green maintain degrees

So we obtain a new point $y \in F$ of different weight; contradiction
Old Lemma:

For any collection of $n^4$ cycles, some $w \in \mathcal{W}$ assigns all of them $\neq 0$ discrepancy

Bipartite key property:

Once we assign a cycle $\neq 0$ discrepancy, it will disappear from the active subgraph
Select $w_1 \in \mathcal{W}$ so that all 4-cycles in $G$ have $\neq 0$ discrepancy

A graph has at most $n^4$ cycles of length 4
Select $w_1 \in \mathcal{W}$ so that all 4-cycles in $G$ have $\neq 0$ discrepancy

- Bipartite key property: $G_1 = (V, \cup_{M \in \mathcal{M}_1} M)$ has no cycles of length $\leq 4$
Select $w_1 \in \mathcal{W}$ so that all 4-cycles in $G$ have $\neq 0$ discrepancy

- Bipartite key property: $G_1 = (V, \bigcup_{M \in \mathcal{M}_1} M)$ has no cycles of length $\leq 4$

Select $w_2 \in \mathcal{W}$ so that all $\leq 8$-cycles in $G_1$ have $\neq 0$ discrepancy

A graph with no $\leq 4$-cycles has at most $n^4$ cycles of length $\leq 8$
Select $w_1 \in \mathcal{W}$ so that all 4-cycles in $G$ have $\neq 0$ discrepancy

- Bipartite key property: $G_1 = (V, \cup_{M \in \mathcal{M}_1} M)$ has no cycles of length $\leq 4$

Select $w_2 \in \mathcal{W}$ so that all $\leq 8$-cycles in $G_1$ have $\neq 0$ discrepancy

- Bipartite key property: $G_2 = (V, \cup_{M \in \mathcal{M}_2} M)$ has no cycles of length $\leq 8$
Select $w_1 \in \mathcal{W}$ so that all 4-cycles in $G$ have $\neq 0$ discrepancy

- Bipartite key property: $G_1 = (V, \cup_{M \in \mathcal{M}_1} M)$ has no cycles of length $\leq 4$

Select $w_2 \in \mathcal{W}$ so that all $\leq 8$-cycles in $G_1$ have $\neq 0$ discrepancy

- Bipartite key property: $G_2 = (V, \cup_{M \in \mathcal{M}_2} M)$ has no cycles of length $\leq 8$

Select $w_3 \in \mathcal{W}$ so that all $\leq 16$-cycles in $G_2$ have $\neq 0$ discrepancy

A graph with no $\leq 8$-cycles has at most $n^4$ cycles of length $\leq 16$
Select $w_1 \in \mathcal{W}$ so that all 4-cycles in $G$ have $\neq 0$ discrepancy
  - Bipartite key property: $G_1 = (V, \cup_{M \in \mathcal{M}_1} M)$ has no cycles of length $\leq 4$

Select $w_2 \in \mathcal{W}$ so that all $\leq 8$-cycles in $G_1$ have $\neq 0$ discrepancy
  - Bipartite key property: $G_2 = (V, \cup_{M \in \mathcal{M}_2} M)$ has no cycles of length $\leq 8$

Select $w_3 \in \mathcal{W}$ so that all $\leq 16$-cycles in $G_2$ have $\neq 0$ discrepancy
  - Bipartite key property: $G_3 = (V, \cup_{M \in \mathcal{M}_3} M)$ has no cycles of length $\leq 16$

\[ \vdots \]
Select \( w_1 \in W \) so that all 4-cycles in \( G \) have \( \neq 0 \) discrepancy

- Bipartite key property: \( G_1 = (V, \cup_{M \in M_1} M) \) has no cycles of length \( \leq 4 \)

Select \( w_2 \in W \) so that all \( \leq 8 \)-cycles in \( G_1 \) have \( \neq 0 \) discrepancy

- Bipartite key property: \( G_2 = (V, \cup_{M \in M_2} M) \) has no cycles of length \( \leq 8 \)

Select \( w_3 \in W \) so that all \( \leq 16 \)-cycles in \( G_2 \) have \( \neq 0 \) discrepancy

- Bipartite key property: \( G_3 = (V, \cup_{M \in M_3} M) \) has no cycles of length \( \leq 16 \)

\[ \vdots \]

\( G_{\log n} = (V, \cup_{M \in M_{\log n}} M) \) have no cycles so \( |M_{\log n}| = 1 \) as required
A graph with no $\leq 4$-cycles has at most $n^4$ cycles of length 8.
A graph with no $\leq 4$-cycles has at most $n^4$ cycles of length 8

- Associate a signature $(a, b, c, d)$ with each 8-cycle
  - $a$ is the first vertex, $b$ is the third vertex, $c$ is the fifth vertex, $d$ is the seventh vertex
A graph with no \( \leq 4 \)-cycles has at most \( n^4 \) cycles of length 8

- Associate a signature \((a, b, c, d)\) with each 8-cycle
  - \( a \) is the first vertex, \( b \) is the third vertex, \( c \) is the fifth vertex, \( d \) is the seventh vertex

Two cycles cannot have the same signature as that would imply a 4-cycle:
Final argument

A graph with no \( \leq 4 \)-cycles has at most \( n^4 \) cycles of length 8

▪ Associate a signature \((a, b, c, d)\) with each 8-cycle
  ▪ \( a \) is the first vertex, \( b \) is the third vertex, \( c \) is the fifth vertex, \( d \) is the seventh vertex

\[
\begin{align*}
&a & b \\
&d & c
\end{align*}
\]

▪ Two cycles cannot have the same signature as that would imply a 4-cycle:

\[
\begin{align*}
&a & b \\
&d & c
\end{align*}
\]
A graph with no $\leq 4$-cycles has at most $n^4$ cycles of length 8

- Associate a signature $(a, b, c, d)$ with each 8-cycle
  - $a$ is the first vertex, $b$ is the third vertex, $c$ is the fifth vertex, $d$ is the seventh vertex

- Two cycles cannot have the same signature as that would imply a 4-cycle:

- So $\# 8$-cycles is at most $\#$ signatures which is at most $n^4$
Some perspective
Polyhedral perspective

isolating in stages

= decreasing sequence of faces
isolating in stages
= decreasing sequence of faces

Polyhedral perspective
isolating in stages

\begin{align*}
F_1 &\quad w = w_1 \\
F_1 &\quad w = \langle w_1, w_2 \rangle \\
F_1 &\quad w = \langle w_1, w_2, w_3 \rangle \\
\end{align*}

= decreasing sequence of faces

Polyhedral perspective
Polyhedral perspective

isolating in stages
  =
  decreasing sequence of faces

$F_1 = w_1$

$F_2 = \langle w_1, w_2 \rangle$

$F_3 = \langle w_1, w_2, w_3 \rangle$

$w$ is isolating
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

$F_1$

$w_1$

$F_2$

$F_3$

$w_2$

$w = w_1$
isolating in stages
= decreasing sequence of faces

Polyhedral perspective
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

$F_1$  
$F_2$

$w_1$  
$w_2$

$w = \langle w_1, w_2 \rangle$
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

$F_1$, $F_2$, $F_3$

$w = \langle w_1, w_2 \rangle$
isolating in stages

= decreasing sequence of faces

$w = \langle w_1, w_2 \rangle$
Polyhedral perspective

1. $F_1$

2. $F_2$

3. $F_2$

isolating in stages

= decreasing sequence of faces

$w = \langle w_1, w_2 \rangle$
isolating in stages
= decreasing sequence of faces

$w = \langle w_1, w_2 \rangle$
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

$w = \langle w_1, w_2, w_3 \rangle$

$F_1 \rightarrow w_1 \rightarrow F_2 \rightarrow w_2 \rightarrow F_3 \rightarrow w_3$
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

\[ w = \langle w_1, w_2, w_3 \rangle \]
isolating in stages

= decreasing sequence of faces

\[ w = \langle w_1, w_2, w_3 \rangle \]

\( w \) is isolating
isolating in stages
= decreasing sequence of faces

Fast decrease due to bipartite matching polytope:
- every face is a subgraph
- Key property: girth doubles in every step

\[ w = \langle w_1, w_2, w_3 \rangle \]

\( w \) is isolating
Difficulties of general case & our approach

Bipartite key property: Once we assign a cycle, it will disappear from the active subgraph
Difficulties of general case & our approach

**Bipartite key property:** Once we assign a cycle ≠ 0 discrepancy, it will disappear from the active subgraph.
General graphs are “exponentially” harder

Edmonds [1965] Perfect matching polytope description on $x \in \mathbb{R}^E$:

- $x_e \geq 0$ for every edge $e$
- $x(\delta(v)) = 1$ for every vertex $v$
- $x(\delta(S)) \geq 1$ for every odd set $S$ of vertices

$(\delta(S) = \text{edges crossing } S)$
Edmonds [1965] Perfect matching polytope description on $x \in \mathbb{R}^E$:

- $x_e \geq 0$ for every edge $e$
- $x(\delta(v)) = 1$ for every vertex $v$
- $x(\delta(S)) \geq 1$ for every odd set $S$ of vertices

So every face $F$ is given as:

$$F = \{ x \in \text{PM} : x_e = 0 \text{ for some edges } e, \quad x(\delta(S)) = 1 \text{ for some odd sets } S \}$$
General graphs are “exponentially” harder

Edmonds [1965] Perfect matching polytope description on $x \in \mathbb{R}^E$:

- $x_e \geq 0$ for every edge $e$
- $x(\delta(v)) = 1$ for every vertex $v$
- $x(\delta(S)) \geq 1$ for every odd set $S$ of vertices

So every face $F$ is given as:

$$F = \{x \in \text{PM} : x_e = 0 \text{ for some edges } e, \quad x(\delta(S)) = 1 \text{ for some odd sets } S\}$$

- In bipartite case:
  $F = \{x \in \text{PM} : x_e = 0 \text{ for some edges } e\}$
  ($F$ given by the active subgraph)

- Now, faces are exponentially harder

- Need $2^{\Omega(n)}$ inequalities [Rothvoss 2013]
General graphs are “exponentially” harder

Edmonds [1965] Perfect matching polytope description on $x \in \mathbb{R}^E$:

- $x_e \geq 0$ for every edge $e$
- $x(\delta(v)) = 1$ for every vertex $v$

So every face $F$ is given as:

$$F = \{x \in \text{PM} : x_e = 0 \text{ for some edges } e, x(\delta(S)) = 1 \text{ for some odd sets } S\}$$

Girth does not make sense as progress measure and bipartite key property fails!

- In bipartite case:
  $F = \{x \in \text{PM} : x_e = 0 \text{ for some edges } e\}$
  ($F$ given by the active subgraph)
- Now, faces are exponentially harder
- Need $2^{\Omega(n)}$ inequalities [Rothvoss 2013]
How bipartite key property fails

$S_1 \subseteq C$ want:
$\text{d} \omega(C), 0 \leq \text{d} \omega(C) \leq 2$, $0$

PM: convex hull of all four matchings:
$F$: convex hull of matchings of weight 1:
$F \subseteq \text{PM}$ but still has all edges...
$F \subseteq \text{PM}$ but still has all edges...
$F = \{ x \in \text{PM}: x(\delta(S)) = 1 \}$
How bipartite key property fails

PM: convex hull of all four matchings:
How bipartite key property fails

want:
\[ d_w(C) \neq 0 \]

\[ d_w(C) = 2 \]

PM: convex hull of all four matchings:

\[ F \subset \text{PM} \]

but still has all edges...
How bipartite key property fails

\[ d_w(C) = 2 \neq 0 \]

PM: convex hull of all four matchings:
How bipartite key property fails

PM: convex hull of all four matchings:

\[
\begin{align*}
F : \text{convex hull of matchings of weight 1:} & \\
\end{align*}
\]

\[
d_w(C) = 2 \neq 0
\]
How bipartite key property fails

PM: convex hull of all four matchings:

\[ d_w(C) = 2 \neq 0 \]

\[ F = \{ x \in PM : x(\delta(S)) = 1 \} \]

\[ F \subsetneq PM \] but still has all edges... 😞
How bipartite key property fails

\[ d_w(C) = 2 \neq 0 \]

**PM**: convex hull of all four matchings:

![Matchings](image)

**F**: convex hull of matchings of weight 1:

![Matchings](image)

\[ F \subset PM \text{ but still has all edges... 😞} \]

\[ F = \{ x \in PM : x(\delta(S)) = 1 \} \]
How bipartite key property fails

PM: convex hull of all four matchings:

$F$: convex hull of matchings of weight 1:

$F \subset PM$ but still has all edges...

$F = \{ x \in PM : x(\delta(S)) = 1 \}$

$d_w(C) = 2 \neq 0$
Main ingredients:

▶ Laminar family of tight constraints (at most $2n - 1$ constraints instead of exponential)

▶ Tight cut constraints decompose the instance ⇒ divide-and-conquer approach
Main ingredients:
- Laminar family of tight constraints (at most $2^n - 1$ constraints instead of exponential)
- Tight cut constraints decompose the instance ⇒ divide-and-conquer approach
Main ingredients:

- Laminar family of tight constraints (at most $2n - 1$ constraints instead of exponential)
- Tight cut constraints decompose the instance
  \[ \Rightarrow \text{divide-and-conquer approach} \]

quite technical path
Every face $F$ is given as:

$$F = \{ x \in PM : x_e = 0 \text{ for some edges } e, \quad x(\delta(S)) = 1 \text{ for some odd sets } S \}$$
Every face $F$ is given as:

$$F = \{ x \in PM : x_e = 0 \text{ for some edges } e, \quad x(\delta(S)) = 1 \text{ for some odd sets } S \}$$

Great news: “some” can be chosen to be a laminar family!
Laminarity

face \sim (edge subset, laminar family)
Laminarity

face $\sim$ (edge subset, laminar family)
Tight odd cuts decomposes instance

- exactly one edge crossing

- once we fix a boundary edge...
Tight odd cuts decomposes instance exactly one edge crossing

- once we fix a boundary edge...
Tight odd cuts decomposes instance

exactly one edge crossing

- once we fix a boundary edge...
Tight odd cuts decomposes instance

- once we fix a boundary edge...
- ... the instance decomposes into two independent ones
Tight odd cuts decomposes instance

- once we fix a boundary edge...
- ... the instance decomposes into two independent ones
**Between friends:** cycles that do not cross tight odd sets behave like in the bipartite case and can thus be removed

**Simplest case:** only one tight odd set
Between friends: cycles that do not cross tight odd sets behave like in the bipartite case and can thus be removed

Simplest case: only one tight odd set

- then every boundary edge determines entire matching
**Between friends:** cycles that do not cross tight odd sets behave like in the bipartite case and can thus be removed

**Simplest case:** only one tight odd set

- then every *boundary edge* determines entire matching
**Between friends:** cycles that do not cross tight odd sets behave like in the bipartite case and can thus be removed

**Simplest case:** only one tight odd set

- then every **boundary edge** determines entire matching
**Between friends:** cycles that do not cross tight odd sets behave like in the bipartite case and can thus be removed

**Simplest case:** only one tight odd set

- then every boundary edge determines entire matching
- so: at most $n^2$ perfect matchings
**Between friends:** cycles that do not cross tight odd sets behave like in the bipartite case and can thus be removed

**Simplest case:** only one tight odd set

- then every **boundary edge** determines entire matching
- so: at most $n^2$ perfect matchings
- some $w \in \mathcal{W}$ will give them different weights
Divide & conquer: chain case

As before, we isolate the whole instance in $O(\log n)$ phases...

...$n^2$ choices $f$

Instance where both sides of the cut are isolated, one $w \in W'$ makes the whole subinstance isolated
Divide & conquer: chain case

As before, we isolate the whole instance in $O(\log n)$ phases...

Instance where both sides of the cut are isolated, one $w \in W'$ makes the whole subinstance isolated.
As before, we isolate the whole instance in $O(\log n)$ phases...

Instance where both sides of the cut are isolated, one $w \in W'$ makes the whole subinstance isolated.
Divide & conquer: chain case

As before, we isolate the whole instance in $O(\log n)$ phases...
Divide & conquer: chain case

As before, we isolate the whole instance in $O(\log n)$ phases.

Instance where both sides of the cut are isolated, one $w \in W'$ makes the whole subinstance isolated.
Divide & conquer: chain case

Instance where both sides of the cut are isolated, one \( w \in \mathcal{W'} \) makes the whole subinstance isolated.
Divide & conquer: chain case

Now instance where both sides of the cut are isolated, one $w \in \mathcal{W}'$ makes the whole subinstance isolated

Instance where both sides of the cut are isolated, one $w \in \mathcal{W}'$ makes the whole subinstance isolated
Divide & conquer: chain case

Now instance where both sides of the cut are isolated, one \( w \in \mathcal{W}' \) makes the whole instance isolated :)
Divide & conquer: chain case

As before we isolate the whole instance in $O(\log n)$ phases

Now instance where both sides of the cut are isolated, one $w \in \mathcal{W}'$ makes the whole instance isolated :) 

Now instance where both sides of the cut are isolated, one $w \in \mathcal{W}'$ makes the whole subinstance isolated

Instance where both sides of the cut are isolated, one $w \in \mathcal{W}'$ makes the whole subinstance isolated
quite technical path

harder than

Removing cycles similar to bipartite case

The chain case (divide-and-conquer)

Theorem S. and Tarnawski [2017]

General matching is in quasi-NC with quasi-polynomial #processors
Carefully selected progress measure allows us to reduce laminar case to

- Removing cycles similar to bipartite case
- The chain case (divide-and-conquer)
Carefully selected progress measure allows us to reduce laminar case to

- Removing cycles similar to bipartite case
- The chain case (divide-and-conquer)

**Theorem**

S. and Tarnawski [2017]

**General** matching is in **QUASI-NC**

with quasi-polynomial \# processors
Future work

- go down to $NC$
  - even for bipartite graphs
  ✓ for planar graphs: [Anari, Vazirani 2017]
Future work

- go down to $\mathcal{NC}$
  - even for bipartite graphs
  - ✓ for planar graphs: [Anari, Vazirani 2017]

- derandomize Isolation Lemma in other cases (any efficiently solvable $\{0, 1\}$ polytope?)
  - ✓ matroid intersection: [Gurjar, Thierauf 2017]
  - ✓ totally unimodular polytopes: [Gurjar, Thierauf, Vishnoi 2017]
Future work

- go down to NC
  - even for bipartite graphs
  - for planar graphs: [Anari, Vazirani 2017]

- derandomize Isolation Lemma in other cases (any efficiently solvable \(\{0, 1\}\) polytope?)
  - matroid intersection: [Gurjar, Thierauf 2017]
  - totally unimodular polytopes: [Gurjar, Thierauf, Vishnoi 2017]

Exact Matching Problem

Given: graph with some edges red, number \(k\).
Is there a perfect matching with exactly \(k\) red edges?

- randomized complexity: even Randomized NC
- deterministic complexity: is it in \(\mathcal{P}\)?
Future work

- go down to $\mathcal{NC}$
  - even for bipartite graphs
  ✓ for planar graphs: [Anari, Vazirani 2017]

- derandomize Isolation Lemma in other cases (any efficiently solvable $\{0, 1\}$ polytope?)
  ✓ matroid intersection: [Gurjar, Thierauf 2017]
  ✓ totally unimodular polytopes: [Gurjar, Thierauf, Vishnoi 2017]

**Exact Matching Problem**

Given: graph with some edges red, number $k$.
Is there a perfect matching with exactly $k$ red edges?

- randomized complexity: even $\text{Randomized } \mathcal{NC}$
- deterministic complexity: is it in $\mathcal{P}$?

Thank you!