Matching is in quasi-NC

Jakub Tarnawski

joint work with Ola Svensson

September 28, 2017 am Mittag
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

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, distributed, streaming/sublinear, ...

Matching is in quasi-NC
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, distributed, streaming/sublinear, ...

Matching is in quasi-NC
Parallel complexity

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

**poly** \( n \) processors

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

Matching is in quasi-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

it's in \textsc{Randomized} $\mathcal{NC}$

poly log $n$ time

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

Matching is in \textit{RANDOMIZED NC} [Lovász 1979]: has \textit{randomized} algorithm that uses:
- polynomially many processors
- polylog time

Search version is in \textit{RANDOMIZED NC}:
- [Karp, Upfal, Wigderson 1986]
- [Mulmuley, Vazirani, Vazirani 1987]
Parallel complexity

- Matching is in \textit{Randomized NC} [Lovász 1979]: has randomized algorithm that uses:
  - polynomially many processors
  - polylog time
- Search version is in \textit{Randomized NC}:
  - [Karp, Upfal, Wigderson 1986]
  - [Mulmuley, Vazirani, Vazirani 1987]

led to understanding of computational relationship between search and decision problems

first matching algorithm to use Tutte’s matrix and Zippel-Schwartz Lemma

introduced the Isolation Lemma

Matching is in \textit{quasi-NC}
Matching is in $\textit{RANDOMIZED }\mathcal{NC}$ [Lovász 1979]: has randomized algorithm that uses:
- polynomially many processors
- polylog time
Search version is in $\textit{RANDOMIZED }\mathcal{NC}$:
- [Karp, Upfal, Wigderson 1986]
- [Mulmuley, Vazirani, Vazirani 1987]
Parallel complexity

- Matching is in \textbf{Randomized NC} [Lovász 1979]:
  - has randomized algorithm that uses:
    - polynomially many processors
    - polylog time
- Search version is in \textbf{Randomized NC}:
  - [Karp, Upfal, Wigderson 1986]
  - [Mulmuley, Vazirani, Vazirani 1987]

Can we derandomize all efficient computation?

THE SEQUENTIAL ALGORITHM WAS NOT RANDOMIZED

Ola Svensson, Jakub Tarnawski
Parallel complexity

Matching is in \textit{Randomized NC} [Lovász 1979]:

- has randomized algorithm that uses:
  - polynomially many processors
  - polylog time

Search version is in \textit{Randomized NC}:

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

Can we derandomize all efficient computation?

\text{THE SEQUENTIAL ALGORITHM}

\text{NO IDEA}

\text{WAS NOT RANDOMIZED}
Parallel complexity

- Matching is in \textsc{Randomized NC} [Lovász 1979]:
  - has randomized algorithm that uses:
    - polynomially many processors
    - polylog time
- Search version is in \textsc{Randomized NC}:
  - [Karp, Upfal, Wigderson 1986]
  - [Mulmuley, Vazirani, Vazirani 1987]

Can we derandomize all efficient computation?

\textbf{The Sequential Algorithm}

\textbf{Was not randomized}
Parallel complexity

- Matching is in \textbf{RANDOMIZED }\mathcal{NC} [Lovász 1979]:
  - has randomized algorithm that uses:
    - polynomially many processors
    - polylog time
- Search version is in \textbf{RANDOMIZED }\mathcal{NC}:
  - [Karp, Upfal, Wigderson 1986]
  - [Mulmuley, Vazirani, Vazirani 1987]

Can we derandomize one of these algorithms?

\textbf{THE SEQUENTIAL ALGORITHM}

\textbf{WAS NOT RANDOMIZED}
Parallel complexity

- Matching is in **RANDOMIZED NC** [Lovász 1979]: has randomized algorithm that uses:
  - polynomially many processors
  - polylog time
- Search version is in **RANDOMIZED NC**:
  - [Karp, Upfal, Wigderson 1986]
  - [Mulmuley, Vazirani, Vazirani 1987]

Can we derandomize one of these algorithms?

Is matching in **NC**?

**THE SEQUENTIAL ALGORITHM**

**WAS NOT RANDOMIZED**
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]
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:

- general
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:

- general
- bipartite
Fenner, Gurjar and Thierauf [2015] showed:

- **Bipartite** matching is in **QUASI-NC**
  \[ (n^{\text{poly log } n} \text{ processors, poly log } n \text{ time, deterministic}) \]
Is matching in $\mathcal{NC}$?

Fenner, Gurjar and Thierauf [2015] showed:

- **Bipartite** matching is in \textsc{quasi-NC} (\(n^{\text{poly log } n}\) processors, \(\text{poly log } n\) time, deterministic)

- Approach fails for non-bipartite graphs

\begin{itemize}
  \item much harder than
\end{itemize}
Our result

We show: general matching is in $\text{QUASI-NC}$:

- $n^{\text{poly log } n}$ processors
- $\text{poly log } n$ time
- deterministic
Outline

1. Isolating weight functions
   [Mulmuley, Vazirani, Vazirani 1987]

2. Bipartite case
   [Fenner, Gurjar, Thierauf 2015]

3. Difficulties of general case
   & our approach
1. Isolating weight functions
[Mulmuley, Vazirani, Vazirani 1987]
Isolating weight functions

**Difficulty:**
too many possible perfect matchings
Isolating weight functions

Difficulty:
too many possible perfect matchings

Weight function $w: E \to \mathbb{Z}^+$ is isolating if there is a unique min-weight perfect matching.
Isolating weight functions

**Difficulty:**
too many possible perfect matchings
Isolating weight functions

Difficulty:

too many possible perfect matchings

Solution: look for a min-weight perfect matching

Tried weights?
Isolating weight functions

Difficulty:
too many possible perfect matchings

Solution: look for a min-weight perfect matching

Weight function \( w : E \rightarrow \mathbb{Z}_+ \) is isolating if there is a unique min-weight perfect matching
Matching is in quasi-NC

isolating weight function

determinant computation in \( \mathcal{NC} \)

matching
Matching is in quasi-NC

Ola Svensson, Jakub Tarnawski
Isolation Lemma

random sampling

isolating weight function

determinant computation in $\mathcal{NC}$

something deterministic?

matching
Isolation Lemma

Weight function $w : E \rightarrow \mathbb{Z}_+$ is isolating if there is a unique min-weight perfect matching.

Isolation Lemma [MVV 1987]

If each $w(e)$ picked randomly from $\{1, 2, \ldots, n^3\}$, then $P[w$ isolating$] \geq 1 - \frac{1}{n}$.

Matching is in quasi-NC
Isolation Lemma

Weight function $w : E \to \mathbb{Z}_+$ is isolating if there is a unique min-weight perfect matching.

Isolation Lemma [MVV 1987]

If each $w(e)$ picked randomly from $\{1, 2, \ldots, n^3\}$, then $P[w \text{ isolating}] \geq 1 - \frac{1}{n}$

- holds more generally, for any set family in place of matchings!
- many applications in complexity theory
- related to Polynomial Identity Testing
Derandomize the Isolation Lemma

- **Challenge:**
  get an isolating weight function deterministically in $\text{NC}$

- **We prove:**
  can construct $n^{O(\log^2 n)}$ weight functions in $\text{QUASI-NC}$
such that one of them is isolating

- **We do it without looking at the graph**

- **Implies:** matching is in $\text{QUASI-NC}$

Special case of derandomizing Polynomial Identity Testing
– for the polynomial being $\det T(G)$
Derandomize the Isolation Lemma

- **Challenge:**
  get an isolating weight function deterministically in $\mathcal{NC}$

- **We prove:**
  can construct $n^{O(\log^2 n)}$ weight functions in $\text{QUASI-NC}$
such that one of them is isolating

- **We do it without looking at the graph**

- **Implies: matching is in $\text{QUASI-NC}$**

Special case of derandomizing Polynomial Identity Testing
– for the polynomial being $\det T(G)$
Derandomize the Isolation Lemma

- **Challenge:**
  get an isolating weight function
deterministically in $\mathcal{NC}$

- **We prove:**
  can construct $n^{O(\log^2 n)}$ weight functions in $\text{QUASI-NC}$
such that one of them is isolating

- **We do it without looking at the graph**

- **Implies:** matching is in $\text{QUASI-NC}$

Special case of derandomizing Polynomial Identity Testing
– for the polynomial being $\det T(G)$
2. Bipartite case
[Fenner, Gurjar, Thierauf 2015]

**Goal:** how to construct $n^{O(\log n)}$ weight functions such that one of them is isolating?
Isolating matchings

What if \( w \) is not isolating?

- there are perfect matchings \( M, M' \)
  with \( w(M) = w(M') \) minimum

Define discrepancy of a cycle:
\[
\text{discrepancy of a cycle: } d_w(C) = w(\text{GREEN}) - w(\text{RED})
\]

If \( \forall C \quad d_w(C) \neq 0 \), then \( w \) isolating!
Isolating matchings

What if \( w \) is not isolating?

- there are perfect matchings \( M, M' \) with \( w(M) = w(M') \) minimum

\[ d_w(C) := w(GREEN) - w(RED) \]

If \( \forall C \thinspace d_w(C) \neq 0 \), then \( w \) isolating!
Isolating matchings

What if $w$ is not isolating?

- there are perfect matchings $M, M'$ with $w(M) = w(M')$ minimum

- define discrepancy of a cycle:
  $$d_w(C) := w(\text{GREEN}) - w(\text{RED})$$

- $d_w(C) = 0 \iff C$ is symmetric difference of alternating cycles

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

New objective: assign $\neq 0$ discrepancy to every cycle
Isolating matchings

What if \( w \) is \textbf{not} isolating?

- there are perfect matchings \( M, M' \) with \( w(M) = w(M') \) minimum
- symmetric difference
  = alternating cycles

\[
d_w(C) = w(\text{GREEN}) - w(\text{RED})
\]

If \( \forall C : d_w(C) \neq 0 \), then \( w \) isolating!
Isolating matchings

What if $w$ is not isolating?

- there are perfect matchings $M, M'$ with $w(M) = w(M')$ minimum
- symmetric difference
  = alternating cycles
- in each cycle $C$,
  $w($GREEN$) = w($RED$)$
  (otherwise could get lighter matching)
Isolating matchings

What if $w$ is not isolating?

- there are perfect matchings $M, M'$ with $w(M) = w(M')$ minimum
- symmetric difference $= \text{alternating cycles}$
- in each cycle $C$,
  $w(\text{GREEN}) = w(\text{RED})$
  (otherwise could get lighter matching)
- define discrepancy of a cycle:
  $d_w(C) := w(\text{GREEN}) - w(\text{RED})$
Isolating matchings

What if \( w \) is not isolating?

- there are perfect matchings \( M, M' \) with \( w(M) = w(M') \) minimum
- symmetric difference
  \( = \) alternating cycles
- in each cycle \( C \),
  \( w(\text{GREEN}) = w(\text{RED}) \)
  (otherwise could get lighter matching)
- define \textbf{discrepancy} of a cycle:
  \[
  d_w(C) := w(\text{GREEN}) - w(\text{RED})
  \]
- \( d_w(C) = 0 \)
Isolating matchings

What if $w$ is not isolating?

- there are perfect matchings $M, M'$ with $w(M) = w(M')$ minimum
- symmetric difference $= \text{alternating cycles}$
- in each cycle $C$, $w(\text{GREEN}) = w(\text{RED})$ (otherwise could get lighter matching)
- define discrepancy of a cycle:
  $d_w(C) := w(\text{GREEN}) - w(\text{RED})$
- $d_w(C) = 0$

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

What if \( w \) is not isolating?

- there are perfect matchings \( M, M' \) with \( w(M) = w(M') \) minimum
- symmetric difference = alternating cycles
- in each cycle \( C \),
  \[ w(\text{GREEN}) = w(\text{RED}) \]
  (otherwise could get lighter matching)
- define discrepancy of a cycle:
  \[ d_w(C) := w(\text{GREEN}) - w(\text{RED}) \]
- \( d_w(C) = 0 \)

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

New objective: assign \( \neq 0 \) discrepancy to every cycle
New objective: assign $\neq 0$ discrepancy to every cycle
New objective: assign \( \neq 0 \) discrepancy to every cycle

Lemma

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:
- for any \( n^4 \) cycles,
- some \( w \in \mathcal{W} \)
- assigns all of them \( \neq 0 \) discrepancy.
Removing cycles

**New objective:** assign $\neq 0$ discrepancy to every cycle

**Lemma**
There is a poly-sized set $\mathcal{W}$ of weight functions such that:
for any $n^4$ cycles,
some $w \in \mathcal{W}$
assigns all of them $\neq 0$ discrepancy.
Removing cycles

**New objective:** assign $\neq 0$ discrepancy to every cycle

**Lemma**

There is a poly-sized set $\mathcal{W}$ of weight functions such that:

for any $n^4$ cycles,

some $w \in \mathcal{W}$ assigns all of them $\neq 0$ discrepancy.

If $\leq n^4$ cycles in the graph: done!
New objective: assign $\neq 0$ discrepancy to every cycle

Lemma

There is a poly-sized set $\mathcal{W}$ of weight functions such that:

For any $n^4$ 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.
Removing cycles

Active subgraph: 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.

$d_w(C_1) = 1 \neq 0$

$d_w(C_2) = 1 \neq 0$

That is, any perfect matching in the active subgraph is min-weight.

By assigning $\neq 0$ discrepancy to 4-cycles, we can remove them.

Then continue restricted to the smaller active subgraph!
Removing cycles

Active subgraph: 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.

\[
d_{w}(C_1) = 1 \neq 0
d_{w}(C_2) = 1 \neq 0
\]

Thus, any perfect matching in the active subgraph is min-weight. By assigning \( \neq 0 \) discrepancy to 4-cycles, we can remove them.

Then continue restricted to the smaller active subgraph.
Removing cycles

Active subgraph: those edges that are in a min-weight perfect matching

Bipartite key property

Once we assign a cycle \( C \neq 0 \) discrepancy, it will disappear from the active subgraph.

\[
d_w(C_1) = 1 \neq 0
\]
\[
d_w(C_2) = 1 \neq 0
\]

That is, any perfect matching in the active subgraph is min-weight.

By assigning \( \neq 0 \) discrepancy to 4-cycles, we can remove them.

Then continue restricted to the smaller active subgraph!
Removing cycles

Active subgraph:
those edges that are in a min-weight perfect matching

\[ d_w(C_1) = 1 \neq 0 \]
\[ d_w(C_2) = 1 \neq 0 \]
Removing cycles

Active subgraph:
those edges that are in a min-weight perfect matching

\[ d_w(C_1) = 1 \neq 0 \]
\[ d_w(C_2) = 1 \neq 0 \]
Removing cycles

Active subgraph:
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.

That is, any perfect matching in the active subgraph is min-weight.

By assigning \( \neq 0 \) discrepancy to cycles, we can remove them. Then continue restricted to the smaller active subgraph!
Removing cycles

Active subgraph:
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.

That is, any perfect matching in the active subgraph is min-weight.

\[
\begin{array}{c}
\begin{array}{c}
\text{C}_{1} \\
\uparrow \\
\text{C}_{2}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
0 \\
1 \\
3 \\
1
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{d}_w(\text{C}_{1}) = 1 \neq 0 \\
\text{d}_w(\text{C}_{2}) = 1 \neq 0
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{d}_w(\text{C}_{1}) = 1 \neq 0 \\
\text{d}_w(\text{C}_{2}) = 1 \neq 0
\end{array}
\end{array}
\end{array}

\]
Removing cycles

Active subgraph:
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.

That is, any perfect matching in the active subgraph is min-weight.

\[
\begin{align*}
d_w(C_1) &= 1 \neq 0 \\
d_w(C_2) &= 1 \neq 0
\end{align*}
\]

By assigning $\neq 0$ discrepancy to 4-cycles, we can remove them. Then continue restricted to the smaller active subgraph!
Lemma

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

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

- Active subgraph has \( \leq n^4 4\)-cycles
  - Apply \( w_1 \in \mathcal{W} \)
    - Active subgraph has no 4-cycles
- Active subgraph has \( \leq n^4 8\)-cycles
  - Apply \( w_2 \in \mathcal{W} \)
    - Active subgraph has no 8-cycles
- Active subgraph has \( \leq n^4 16\)-cycles
  - Apply \( w_3 \in \mathcal{W} \)
    - Active subgraph has no 16-cycles
- ...
  - Apply \( w_{\log n} \in \mathcal{W} \)
    - Active subgraph has no cycles whatsoever
  - Success!
Lemma

There is a poly-sized set $\mathcal{W}$ of weight functions such that:
for any $n^4$ cycles, some $w \in \mathcal{W}$ removes all of them.

- active subgraph has $\leq n^4$ 4-cycles
  - apply $w_1 \in \mathcal{W}$
    - active subgraph has no 4-cycles
  - apply $w_2 \in \mathcal{W}$
    - active subgraph has $\leq n^4$ 8-cycles
  - apply $w_3 \in \mathcal{W}$
    - active subgraph has no 8-cycles
  - apply $w_4 \in \mathcal{W}$
    - active subgraph has $\leq n^4$ 16-cycles
  - apply $w_{\log n} \in \mathcal{W}$
    - active subgraph has no 16-cycles
  - ...

Matching is in quasi-NC
Lemma

There is a poly-sized set $\mathcal{W}$ of weight functions such that:

for any $n^4$ cycles, some $w \in \mathcal{W}$ removes all of them.

- active subgraph has $\leq n^4$ 4-cycles
  - apply $w_1 \in \mathcal{W}$
    - active subgraph has no 4-cycles
  - active subgraph has $\leq n^4$ 8-cycles
    - apply $w_2 \in \mathcal{W}$
      - active subgraph has no 8-cycles
  - active subgraph has $\leq n^4$ 16-cycles
    - apply $w_3 \in \mathcal{W}$
      - active subgraph has no 16-cycles
  - ...  
    - apply $w_{\log n} \in \mathcal{W}$  
      - active subgraph has no cycles whatsoever
      - success!
Lemma

There is a poly-sized set $\mathcal{W}$ of weight functions such that:

for any $n^4$ cycles, some $w \in \mathcal{W}$ removes all of them.

\[
W = W_1
\]

- active subgraph has $\leq n^4$ 4-cycles
- apply $w_1 \in \mathcal{W}$
  - active subgraph has no 4-cycles
  - active subgraph has $\leq n^4$ 8-cycles
  - apply $w_2 \in \mathcal{W}$
    - active subgraph has no 8-cycles
    - active subgraph has $\leq n^4$ 16-cycles
    - apply $w_3 \in \mathcal{W}$
      - active subgraph has no 16-cycles
      - ...  
      - apply $w_{\log n} \in \mathcal{W}$
        - active subgraph has no cycles whatsoever
        - success!
Isolating in stages

\[ w = w_1 \]

Lemma

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

For any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles
  - active subgraph has \( \leq n^4 \) 8-cycles
    - apply \( w_2 \in \mathcal{W} \)
      - active subgraph has no 8-cycles
  - active subgraph has \( \leq n^4 \) 16-cycles
    - apply \( w_3 \in \mathcal{W} \)
      - active subgraph has no 16-cycles
  - ...
    - apply \( w_{\log n} \in \mathcal{W} \)
      - active subgraph has no cycles whatsoever
      - success!
Isolating in stages

Lemma
There is a poly-sized set \( \mathcal{W} \) of weight functions such that:
for any \( n^4 \) cycles,
some \( w \in \mathcal{W} \) removes all of them.

Counting argument
No cycles of length \( \leq r \)
\( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

\[ w = w_1 \]

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles
  - active subgraph has \( \leq n^4 \) 8-cycles
    - apply \( w_2 \in \mathcal{W} \)
      - active subgraph has no 8-cycles
    - active subgraph has \( \leq n^4 \) 16-cycles
      - apply \( w_3 \in \mathcal{W} \)
        - active subgraph has no 16-cycles
      - ... 
        - apply \( w_{\log n} \in \mathcal{W} \)
          - active subgraph has no cycles whatsoever
          - success!
Isolating in stages

\[ w = w_1 \]

**Lemma**

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

**for any** \( n^4 \) cycles,

**some** \( w \in \mathcal{W} \) 

removes all of them.

**Counting argument**

No cycles of length \( \leq r \) 
\[ \implies \] only \( n^4 \) cycles of length \( \leq 2r \)
Isolating in stages

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

**Lemma**

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

- for any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

**Counting argument**

- No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles
  - active subgraph has \( \leq n^4 \) 8-cycles
    - apply \( w_2 \in \mathcal{W} \)
      - active subgraph has no 8-cycles
    - active subgraph has \( \leq n^4 \) 16-cycles
      - apply \( w_3 \in \mathcal{W} \)
        - active subgraph has no 16-cycles
      - ...
    - apply \( w_{\log n} \in \mathcal{W} \)
      - active subgraph has no cycles whatsoever
      - success!
Isolating in stages

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

Lemma
There is a poly-sized set \( \mathcal{W} \) of weight functions such that:
for any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

Counting argument
No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

active subgraph has \( \leq n^4 \) 4-cycles
apply \( w_1 \in \mathcal{W} \)
active subgraph has no 4-cycles

active subgraph has \( \leq n^4 \) 8-cycles
apply \( w_2 \in \mathcal{W} \)
active subgraph has no 8-cycles

active subgraph has \( \leq n^4 \) 16-cycles
apply \( w_3 \in \mathcal{W} \)
active subgraph has no 16-cycles

... apply \( w_{\log n} \in \mathcal{W} \)
active subgraph has no cycles whatsoever
success!
Isolating in stages

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

**Lemma**
There is a poly-sized set \( \mathcal{W} \) of weight functions such that:
for any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

**Counting argument**
No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles
- active subgraph has \( \leq n^4 \) 8-cycles
  - apply \( w_2 \in \mathcal{W} \)
    - active subgraph has no 8-cycles
- active subgraph has \( \leq n^4 \) 16-cycles
  - apply \( w_3 \in \mathcal{W} \)
    - active subgraph has no 16-cycles
- ...
  - apply \( w_{\log n} \in \mathcal{W} \)
    - active subgraph has no cycles whatsoever
- success!
Isolating in stages

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

**Lemma**

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

- for any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

**Counting argument**

No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
  - active subgraph has no 4-cycles
- active subgraph has \( \leq n^4 \) 8-cycles
  - apply \( w_2 \in \mathcal{W} \)
  - active subgraph has no 8-cycles
- active subgraph has \( \leq n^4 \) 16-cycles
  - apply \( w_3 \in \mathcal{W} \)
  - active subgraph has no 16-cycles
- ... (\( \log n \) steps)
  - apply \( w_{\log n} \in \mathcal{W} \)
  - active subgraph has no cycles whatsoever
  - success!
Isolating in stages

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

**Lemma**
There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

for any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

**Counting argument**
No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles

- active subgraph has \( \leq n^4 \) 8-cycles
  - apply \( w_2 \in \mathcal{W} \)
    - active subgraph has no 8-cycles

- active subgraph has \( \leq n^4 \) 16-cycles
  - apply \( w_3 \in \mathcal{W} \)
    - active subgraph has no 16-cycles

- ... 
  - apply \( w_{\log n} \in \mathcal{W} \)
    - active subgraph has no cycles whatsoever
    - success!
Isolating in stages

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

**Lemma**

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

- for any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

**Counting argument**

No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles
  - active subgraph has \( \leq n^4 \) 8-cycles
    - apply \( w_2 \in \mathcal{W} \)
      - active subgraph has no 8-cycles
  - active subgraph has \( \leq n^4 \) 16-cycles
    - apply \( w_3 \in \mathcal{W} \)
      - active subgraph has no 16-cycles
- ...  
  - apply \( w_{\log n} \in \mathcal{W} \)
    - active subgraph has no cycles whatsoever
    - success!
Isolating in stages

\[ w = \langle w_1, w_2, w_3, \ldots, w_{\log n} \rangle \]

**Lemma**

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

- for any \( n^4 \) cycles, some \( w \in \mathcal{W} \) removes all of them.

**Counting argument**

No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles
- active subgraph has \( \leq n^4 \) 8-cycles
  - apply \( w_2 \in \mathcal{W} \)
    - active subgraph has no 8-cycles
- active subgraph has \( \leq n^4 \) 16-cycles
  - apply \( w_3 \in \mathcal{W} \)
    - active subgraph has no 16-cycles
  - ...  
  - apply \( w_{\log n} \in \mathcal{W} \)
    - active subgraph has no cycles whatsoever
    - success!
Isolating in stages

\[ w = \langle w_1, w_2, w_3, \ldots, w_{\log n} \rangle \]

Lemma

There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

for any \( n^4 \) cycles,

some \( w \in \mathcal{W} \) removes all of them.

Counting argument

No cycles of length \( \leq r \)

\[ \implies \text{only } n^4 \text{ cycles of length } \leq 2r \]

active subgraph has \( \leq n^4 \) 4-cycles

apply \( w_1 \in \mathcal{W} \)

active subgraph has no 4-cycles

active subgraph has \( \leq n^4 \) 8-cycles

apply \( w_2 \in \mathcal{W} \)

active subgraph has no 8-cycles

active subgraph has \( \leq n^4 \) 16-cycles

apply \( w_3 \in \mathcal{W} \)

active subgraph has no 16-cycles

... \( \implies \)

apply \( w_{\log n} \in \mathcal{W} \)

active subgraph has no cycles whatsoever

success!
Isolating in stages

\[ w = \langle w_1, w_2, w_3, \ldots, w_{\log n} \rangle \]

**Lemma**
There is a poly-sized set \( \mathcal{W} \) of weight functions such that:

**for any** \( n^4 \) **cycles,**

**some** \( w \in \mathcal{W} \) **removes all of them.**

**Counting argument**

No cycles of length \( \leq r \) \( \implies \) only \( n^4 \) cycles of length \( \leq 2r \)

- active subgraph has \( \leq n^4 \) 4-cycles
  - apply \( w_1 \in \mathcal{W} \)
    - active subgraph has no 4-cycles
- active subgraph has \( \leq n^4 \) 8-cycles
  - apply \( w_2 \in \mathcal{W} \)
    - active subgraph has no 8-cycles
- active subgraph has \( \leq n^4 \) 16-cycles
  - apply \( w_3 \in \mathcal{W} \)
    - active subgraph has no 16-cycles
- ... 
  - apply \( w_{\log n} \in \mathcal{W} \)
    - active subgraph has no cycles whatsoever
    - success!
Isolating in stages

\[ w = \langle w_1, w_2, \ldots, w_{\log n} \rangle \]

- For each stage \( i \), some \( w_i \in \mathcal{W} \) removes the wanted cycles
- So some concatenation \( \langle w_1, w_2, \ldots, w_{\log n} \rangle \) is isolating
- But not sure how to check in \( \mathcal{NC} \) if given \( w_i \) is good...

The oblivious algorithm checks all concatenations:

\[ |\mathcal{W}|^{\log n} = n^{O(\log n)} \]
3. Difficulties of general case & our approach
Bipartite key property fails

Once we assign a cycle $\neq 0$ discrepancy, it will disappear from the active subgraph.
Polyhedral perspective

- PM: perfect matching polytope
  (convex hull of all perfect matchings)
PM: perfect matching polytope
(convex hull of all perfect matchings)
Polyhedral perspective

- **PM**: perfect matching polytope (convex hull of all perfect matchings)
- **F**: set of points in PM that minimize \( w \)
  - F is a face of PM

\[ F \text{ is a face of PM} \iff |F| = 1 \]
Polyhedral perspective

- **PM**: perfect matching polytope (convex hull of all perfect matchings)
- **$F$**: set of points in PM that minimize $w$
  - $F$ is a face of PM
- $w$ isolating $\iff |F| = 1$ ($F$ is a vertex)

Matching is in quasi-NC
Polyhedral perspective

- **PM**: perfect matching polytope (convex hull of all perfect matchings)
- **$F$**: set of points in $\text{PM}$ that minimize $w$
  - $F$ is a face of $\text{PM}$
- $w$ isolating $\iff |F| = 1$ ($F$ is a vertex)

$F$ is not isolating $\iff$ want to avoid a zero-measure set deterministically (similar to Polynomial Identity Testing)
Polyhedral perspective

- **PM**: perfect matching polytope (convex hull of all perfect matchings)
- **$F$**: set of points in PM that minimize $w$
  - $F$ is a face of PM
- $w$ isolating $\iff |F| = 1$ ($F$ is a vertex)

$w$ isolating

Matching is in quasi-NC
Polyhedral perspective

- **PM**: perfect matching polytope (convex hull of all perfect matchings)
- **$F$**: set of points in PM that minimize $w$
  - $F$ is a face of PM
- $w$ isolating $\iff |F| = 1$ ($F$ is a vertex)

Want to avoid a zero-measure set deterministically (similar to Polynomial Identity Testing)
**Polyhedral perspective**

isolating in stages

= decreasing sequence of faces

Ola Svensson, Jakub Tarnawski

Matching is in quasi-NC
Polyhedral perspective

isolating in stages

= decreasing sequence of faces

$w = w_1$
Polyhedral perspective

isolating in stages

= 

decreasing sequence of faces

\[ w = w_1 \]

\[ F_1 \]

\[ w_1 \]
Polyhedral perspective

Isolating in stages

= decreasing sequence of faces

$F_1$

$w_1$

$F_1$

$w = w_1$
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

\( w = w_1 \)
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

w = w₁

Ola Svensson, Jakub Tarnawski
isolating in stages
= decreasing sequence of faces

\[ w = \langle w_1, w_2 \rangle \]
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

$F_1$

$w_1$

$F_2$

$w_2$

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

isolating in stages
= decreasing sequence of faces

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

isolating in stages
= decreasing sequence of faces

\[ w = \langle w_1, w_2 \rangle \]
Polyhedral perspective

isolating in stages

= decreasing sequence of faces

\[ w = \langle w_1, w_2 \rangle \]
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

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

Ola Svensson, Jakub Tarnawski

Matching is in quasi-NC
Polyhedral perspective

isolating in stages
= decreasing sequence of faces

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

isolating in stages
=

decreasing sequence of faces

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

$w$ is isolating
Polyhedral perspective

1. $F_1$

2. $F_2$

3. $F_3$

isolating in stages

= 

decreasing sequence of faces

decreasing fast due to the bipartite matching polytope:

- bipartite key property: every face is a subgraph
- so girth doubles at every step

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

$w$ is isolating

Ola Svensson, Jakub Tarnawski

Matching is in quasi-NC
Edmonds [1965]

PM described as set of $x \in \mathbb{R}^E$ such that:

- $x_e \geq 0$ for every edge $e$
- $x(\delta(v)) = 1$ for every vertex $v$ \hspace{1cm} ($\delta(S) = \text{edges crossing } S$)
- $x(\delta(S)) \geq 1$ for every odd set $S$ of vertices
Edmonds [1965]

PM described as set of $x \in \mathbb{R}^E$ such that:

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

So every face $F$ is given as:

$$F = \{x \in \text{PM} : x_e = 0 \text{ for some edges } e, \quad x(\delta(S)) = 1 \text{ for some odd sets } S\}$$
Edmonds [1965]

PM described as set of $x \in \mathbb{R}^E$ such that:

- $x_e \geq 0$ for every edge $e$
- $x(\delta(v)) = 1$ for every vertex $v$ \hspace{1cm} ($\delta(S) =$ edges crossing $S$)
- $x(\delta(S)) \geq 1$ for every odd set $S$ of vertices

So every face $F$ is given as:

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

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

Edmonds [1965]

PM described as set of $x \in \mathbb{R}^E$ such that:

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

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

\[ d_{\text{w}}(C) \neq 0 \]

PM: convex hull of all four matchings:

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

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

\[ \text{PM: convex hull of all four matchings:} \]

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

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

want: $d_w(C) \neq 0$

$PM$: convex hull of all four matchings:

Ola Svensson, Jakub Tarnawski

Matching is in quasi-NC
How bipartite key property fails

$\begin{array}{c}
\begin{array}{c}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
\end{array} \\
\begin{array}{c}
C \\
\end{array} \\
\begin{array}{c}
1 \\
\end{array} \\
\end{array}$

$d_w(C) = 2 \neq 0$

$PM$: convex hull of all four matchings:

Ola Svensson, Jakub Tarnawski

Matching is in quasi-NC
How bipartite key property fails

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

\[ \text{PM: convex hull of all four matchings:} \]

\[ \text{F: convex hull of matchings of weight 1:} \]
How bipartite key property fails

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

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

- Top left
- Top middle
- Top right
- Bottom middle

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

- Top left
- Top middle
- Bottom middle

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

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

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

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

PM: convex hull of all four matchings:

\[
\begin{align*}
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix} \\
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix} \\
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix} \\
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix}
\end{align*}
\]

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

F: convex hull of matchings of weight 1:

\[
\begin{align*}
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix} \\
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix} \\
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix} \\
&\begin{bmatrix}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{bmatrix}
\end{align*}
\]

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

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

Main ingredients:

▶ Laminar family of tight cut constraints

▶ Tight cut constraints decompose the instance ⇒ divide-and-conquer approach

Ola Svensson, Jakub Tarnawski

Matching is in quasi-NC
How we cope

Main ingredients:
- Laminar family of tight cut constraints
- Tight cut constraints decompose the instance ⇒ divide-and-conquer approach

Matching is in quasi-NC
How we cope

Main ingredients:

- Laminar family of tight cut constraints
- Tight cut constraints decompose the instance
  ⇒ divide-and-conquer approach
Laminarity

Every face $F$ is given as:

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

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!

(at most $n/2$ constraints instead of exponentially many to describe a face)
face $\sim$ (edge subset, laminar family)
Laminarity

\[ F_2 \sim (\text{edge subset, laminar family}) \]
Tight odd cuts are not all bad

exactly one edge crossing

once we fix a boundary edge...
Tight odd cuts are not all bad

exactly one edge crossing

once we fix a boundary edge...
Tight odd cuts are not all bad

exactly one edge crossing

▶ once we fix a boundary edge...
Tight odd cuts are not all bad

Exactly one edge crossing

Once we fix a boundary edge...
Tight odd cuts are not all bad

exactly one edge crossing

▶ once we fix a boundary edge...
Tight odd cuts are not all bad

▶ once we fix a boundary edge...
Tight odd cuts are not all bad

▶ once we fix a boundary edge...
▶ ... the instance decomposes into two independent ones
Tight odd cuts are not all bad

once we fix a **boundary edge**...

... the instance decomposes into two **independent** ones
Simplest case of laminar family: 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
Divide & conquer

Simplest case of laminar family: 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

▶ then every boundary edge determines entire matching
Simplest case of laminar family: 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

▶ then every boundary edge determines entire matching
Divide & conquer

Simplest case of laminar family: 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

▶ then every boundary edge determines entire matching
Simplest case of laminar family: 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

▶ then every boundary edge determines entire matching
▶ so: at most $n^2$ perfect matchings
Simplest case of laminar family: 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

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

- remove cycles *not crossing tight odd-sets*

- use tight odd-sets to decompose problem (divide & conquer)

Details: see paper or talk to me :)
Our dichotomy

Dichotomy:

- remove cycles *not crossing tight odd-sets*
- *use tight odd-sets* to decompose problem (divide & conquer)

Details: see paper or talk to me :)

Ola Svensson, Jakub Tarnawski
Future work

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

Thank you!

Ola Svensson, Jakub Tarnawski

Matching is in quasi-NC
Future work

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

▶ derandomize Isolation Lemma in other cases
  ✓ matroid intersection: [Gurjar, Thierauf 2017]
  ✓ totally unimodular polytopes: [Gurjar, Thierauf, Vishnoi 2017]
  ▶ any efficiently solvable 0/1-polytope?
Future work

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

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

Exact Matching

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 $\mathcal{P}$?
Future work

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

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

### Exact Matching

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!

Ola Svensson, Jakub Tarnawski

Matching is in quasi-$\mathcal{NC}$