Maximal Matching
Maximal Matching in $\Delta$-regular graphs

- Maximal Matching, $\Sigma = \{M, O, P\}$
- $P_0 = (M^{\Delta-1} \mid P^\Delta), B_0 = (M[P^\Delta]^{\Delta-1} \mid O^\Delta)$

Balliu et al. (2019)
Maximal Matching needs $\Omega(\min\{\Delta, \log n / \log \log n\})$ rounds in the LOCAL Model.
Parametrized Matching

Parametrized problem $\Pi_\Delta(x, y)$

- Addition of joker output $X$.
- $B_\Delta(x, y) = \left(MO^{d-1} \mid P^d\right) O^y X^x$
- $W_\Delta(x, y) = \left([MX][POX]^{d-1} \mid [OX]^d\right) [POX]^y [MPOX]^x$

$$d = \Delta - x - y$$

- Maximal Matching in $T$ rounds, $\Rightarrow \Pi_\Delta(0, 0)$ in $T$ rounds.
- $\Pi_\Delta(x, y)$ in $T$ rounds $\Rightarrow \Pi_\Delta(x + 1, y + x)$ in $T - 1$ rounds.
- What lower bound on the complexity can we deduce?
Parametrized Matching

Parametrized problem $\Pi_\Delta(x, y)$

- Addition of joker output $X$.
- $B_\Delta(x, y) = \left( \text{MO}^{d-1} \right| P^d \right) O^y X^x$
- $W_\Delta(x, y) = \left( \left[ \text{MX} \right] \left[ \text{POX} \right]^{d-1} \right| \left[ \text{OX} \right]^d \right) \left[ \text{POX} \right]^y \left[ \text{MPOX} \right]^x$

$$d = \Delta - x - y$$

- Maximal Matching in $T$ rounds, $\Rightarrow \Pi_\Delta(0, 0)$ in $T$ rounds.
- $\Pi_\Delta(x, y)$ in $T$ rounds $\Rightarrow \Pi_\Delta(x + 1, y + x)$ in $T - 1$ rounds.
- What lower bound on the complexity can we deduce?

$$\Omega(\sqrt{\Delta})$$
Weak $k$-matching

**Weak $k$-matching**: For each node $v$,

- $v$ is matched to $l \in [1, k]$ nodes
- $v$ is unmatched and all the neighbors of $v$ are matched

Suppose that we can solve Maximal Matching in $o(\Delta)$ rounds:

$\Rightarrow$ we solve weak "$\sqrt{\Delta}$-Matching" in $o(\sqrt{\Delta})$ rounds.
Weak $\sqrt{\Delta}$-matching
Weak $k$-matching

**Weak $k$-matching** : For each node $v$,
- $v$ is matched to $l \in [1, k]$ nodes
- $v$ is unmatched and all the neighbors of $v$ are matched

Suppose that we can solve Maximal Matching in $o(\Delta)$ rounds :

$\Rightarrow$ we solve weak "$\sqrt{\Delta}$-Matching" in $o(\sqrt{\Delta})$ rounds.
$\Rightarrow$ we solve $\Pi_{\Delta}(\sqrt{\Delta} - 1, 0)$ in $o(\sqrt{\Delta})$ rounds.

Contradiction $\Rightarrow$ Maximal Matching cannot be solved in $o(\Delta)$ rounds.

How can we deduce that Maximal Matching cannot be solved in $o(\log n / \log \log n)$ rounds?

$\Delta = \Theta(\log n / \log \log n)$

$\Rightarrow$ $\Delta = \Theta(n)$
Weak $k$-matching

Weak $k$-matching: For each node $v$,
- $v$ is matched to $l \in [1, k]$ nodes
- $v$ is unmatched and all the neighbors of $v$ are matched

Suppose that we can solve Maximal Matching in $o(\Delta)$ rounds:

$\Rightarrow$ we solve weak "$\sqrt{\Delta}$-Matching" in $o(\sqrt{\Delta})$ rounds.
$\Rightarrow$ we solve $\Pi_{\Delta}(\sqrt{\Delta} - 1, 0)$ in $o(\sqrt{\Delta})$ rounds.
$\Rightarrow$ we solve $\Pi_{\Delta}(\sqrt{\Delta} + o(\sqrt{\Delta}), o(\Delta))$ in 0 rounds.
Weak $k$-matching

**Weak $k$-matching**: For each node $v$,

- $v$ is matched to $l \in [1, k]$ nodes
- $v$ is unmatched and all the neighbors of $v$ are matched

Suppose that we can solve Maximal Matching in $o(\Delta)$ rounds:

$\Rightarrow$ we solve weak "$\sqrt{\Delta}$-Matching" in $o(\sqrt{\Delta})$ rounds.

$\Rightarrow$ we solve $\Pi_{\Delta}(\sqrt{\Delta} - 1, 0)$ in $o(\sqrt{\Delta})$ rounds.

$\Rightarrow$ we solve $\Pi_{\Delta}(\sqrt{\Delta} + o(\sqrt{\Delta}), o(\Delta))$ in 0 rounds.

Contradiction $\Rightarrow$ Maximal Matching cannot be solved in $o(\Delta)$ rounds.
Weak $k$-matching

**Weak $k$-matching**: For each node $v$,
- $v$ is matched to $l \in [1, k]$ nodes
- $v$ is unmatched and all the neighbors of $v$ are matched

Suppose that we can solve Maximal Matching in $o(\Delta)$ rounds:

$\Rightarrow$ we solve weak "$\sqrt{\Delta}$-Matching" in $o(\sqrt{\Delta})$ rounds.

$\Rightarrow$ we solve $\Pi_\Delta(\sqrt{\Delta} - 1, 0)$ in $o(\sqrt{\Delta})$ rounds.

$\Rightarrow$ we solve $\Pi_\Delta(\sqrt{\Delta} + o(\sqrt{\Delta}), o(\Delta))$ in 0 rounds.

Contradiction $\Rightarrow$ Maximal Matching cannot be solved in $o(\Delta)$ rounds.

How can we deduce that Maximal Matching cannot be solved in $o(\log n / \log \log n)$ rounds?
Weak $k$-matching

**Weak $k$-matching**: For each node $v$,
- $v$ is matched to $l \in [1, k]$ nodes
- $v$ is unmatched and all the neighbors of $v$ are matched

Suppose that we can solve Maximal Matching in $o(\Delta)$ rounds:

$\Rightarrow$ we solve weak "$\sqrt{\Delta}$-Matching" in $o(\sqrt{\Delta})$ rounds.
$\Rightarrow$ we solve $\Pi_\Delta(\sqrt{\Delta} - 1, 0)$ in $o(\sqrt{\Delta})$ rounds.
$\Rightarrow$ we solve $\Pi_\Delta(\sqrt{\Delta} + o(\sqrt{\Delta}), o(\Delta))$ in 0 rounds.

Contradiction $\Rightarrow$ Maximal Matching cannot be solved in $o(\Delta)$ rounds.

How can we deduce that Maximal Matching cannot be solved in $o(\log n / \log \log n)$ rounds?

$\Delta = \Theta(\log n / \log \log n) \Rightarrow \Delta^\Delta = \Theta(n)$
Complexity Landscapes
Landscape of LCL problems

Randomized time complexity

Deterministic time complexity

Figures from https://jukkasuomela.fi/landscape-of-locality/
Landscape of LCL problems

\[ \Theta(\log n) \]
deterministic

\[ \Theta(\log \log n) \]
randomized
Landscape of LCL problems

- Maximal matching
- Maximal independent set
- 3-coloring paths and cycles
- Deterministic
- Randomized

Figures from https://jukkasuomela.fi/landscape-of-locality/
Complexities on Graphs with the LOCAL Model

2020

Figures from https://jukkasuomela.fi/landscape-of-locality/
Path and Cycle Complexities
### 3-coloring neighborhoods

<table>
<thead>
<tr>
<th>213</th>
<th>121</th>
<th>212</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>321</td>
<td>232</td>
<td>323</td>
<td>231</td>
</tr>
<tr>
<td>132</td>
<td>313</td>
<td>131</td>
<td>312</td>
</tr>
</tbody>
</table>
3-coloring neighborhoods

213  121  212  123
321  232  323  231
132  313  131  312
3-coloring neighborhoods
3-coloring neighborhoods

Diagram showing the 3-coloring neighborhoods with numbers arranged in a circular manner, connected by lines to represent adjacency.
3-coloring neighborhoods
3-coloring neighborhoods
3-coloring neighborhoods

\begin{align*}
\text{13} & \rightarrow \text{31} \\
\text{21} & \rightarrow \text{12} \\
\text{32} & \rightarrow \text{23} \\
\text{1} & \rightarrow \text{3} \\
\text{2} & \rightarrow \text{1}
\end{align*}
3-coloring

Transition Automata

1

2

3
3-coloring  Independent Set

Transition Automata
Transition Automata

3-coloring

Independent Set

1

2

3

0

1
Transition Automata

3-coloring  Independent Set  MIS
Transition Automata

3-coloring

Independent Set

MIS
Transition Automata

3-coloring  Independent Set  MIS  2-coloring

\[ \begin{array}{cccc}
1 & \rightarrow & 2 & \rightarrow \\
2 & \rightarrow & 3 & \rightarrow \\
\end{array} \]

\[ \begin{array}{cccc}
0 & \rightarrow & 1 & \rightarrow \\
\end{array} \]

\[ \begin{array}{cccc}
00 & \rightarrow & 01 & \rightarrow \\
10 & \rightarrow & 01 & \rightarrow \\
\end{array} \]
Transition Automata

3-coloring

Independent Set

MIS

2-coloring

1

0

00

1

01

10

2

9 / 22
Complexity Separation on Paths

**Naor, Stockmeyer (1995)**
If the input graph is an unlabeled path or cycle, the time complexity is decidable.

The different time complexities are $O(1)$, $\Theta(\log^* n)$ and $\Omega(n)$. 
How to Decide the Complexity

- $O(1)$
How to Decide the Complexity

- \( O(1) \): There exists a self-loop.
How to Decide the Complexity

- $O(1)$: There exists a self-loop.

**Naor, Stockmeyer (1995)**
If a problem on paths can be computed in $o(\log^* n)$ rounds, there exists an algorithm with the same complexity stable under relative order.
How to Decide the Complexity

- $O(1)$: There exists a self-loop.
- $\Theta(\log^* n)$
How to Decide the Complexity

- $O(1)$: There exists a self-loop.

- $\Theta(\log^* n)$: There exists a node $v$ and $k > 0$ such as $\forall i \geq k$, there exists a path of length $i$ from $v$ to $v$.

- $\Omega(n)$
How to Decide the Complexity

- $O(1)$: There exists a self-loop.

- $\Theta(\log^* n)$: There exists a node $v$ and $k > 0$ such as $\forall i \geq k$, there exists a path of length $i$ from $v$ to $v$.

- $\Omega(n)$: For all nodes $v$, either $v$ cannot be reached from $v$, or there exists $k > 1$ such as any path of length $i > 0$ from $v$ to $v$ is such as $k$ divides $i$. 
Decidability on Paths with Inputs
Problem on Paths with Inputs

- 3-color the red nodes.
- Carry the color through the blue nodes.
Problem on Paths with Inputs

• 3-color the red nodes.
• Carry the color through the blue nodes.
3-color the red nodes.

Carry the color through the blue nodes.
- 3-color the red nodes.
- Carry the color through the blue nodes.
**Balliu et al. (2019)**

For any LCL problem on cycle graphs, its complexity is either $\Omega(n)$ or $O(\log^* n)$. Moreover, there is an algorithm that decides whether the problem has complexity $\Omega(n)$ or $O(\log^* n)$ on cycle graphs; for the case the complexity is $O(\log^* n)$, the algorithm outputs a description of an $O(\log^* n)$-round deterministic LOCAL algorithm that solves it.

For any LCL problem on cycle graphs, its complexity is either $\Omega(\log^* n)$ or $O(1)$. Moreover, there is an algorithm that decides whether the problem has complexity $\Omega(\log^* n)$ or $O(1)$ on cycle graphs; for the case the complexity is $O(1)$, the algorithm outputs a description of an $O(1)$-round deterministic LOCAL algorithm that solves it.
It is PSPACE-hard to distinguish whether a given LCL problem with input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on globally oriented path graphs.
Balliu et. al (2019)
It is PSPACE-hard to distinguish whether a given LCL problem with input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on globally oriented path graphs.

- Input: nodes in states $a$, $b$, or representing a Turing Machine tape on space $k$
- Output:
  - Tape nodes carry the $a$ or $b$ closest from the left
  - Tape nodes prove that the tape encodes a bad TM simulation and carry state $E$ after the mistake
Turing Machine Encoding
Error Detection
Balliu et al (2019) It is PSPACE-hard to distinguish whether a given LCL problem without input labels can be solved in $O(1)$ time or needs $\Omega(n)$ time on trees with degree $\Delta = 3$. 
Encoding a Number in a Tree

Encoding $6 = 0110_2$
Encoding a Number in a Tree

Encoding $6=0110_2$
Encoding a Number in a Tree

Encoding $6 = 0110_2$
Encoding a Number in a Tree

Encoding $6 = 0110_2$
Encoding a Number in a Tree

Encoding $6=0110_2$
Encoding a Number in a Tree

Encoding $6 = 0110_2$
Encoding a Number in a Tree

Encoding $6_{10} = 0110_2$
Encoding a Number in a Tree

Encoding $6=0110_2$
Encoding the Input of the Path

\[ \ldots a_1 a_2 a_3 a_4 a_5 a_6 \ldots \]
Encoding the Input of the Path

\[ a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \]
Toroidal Grids
Toroidal Grid Graphs

Grid Graphs:
- Toroidal $n \times n$ grid
- Consistent orientation N/S/E/W

Brandt et al (2017)
Given any LCL problem $P$ with an algorithm $A$ that solves $P$ in time $T = o(n)$, there exists an algorithm $B$ that solves $P$ and has running time $O(\log^* n)$. 
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm A in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$:
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm $A$ in time $T(n) = o(n)$:
- Take $k$ such that $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Speed Up Algorithm

Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
- New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
Algorithm $A$ in time $T(n) = o(n)$:

- Take $k$ such as $T(k) < k/4 - 4$
- Compute Independent set $I$ of $G^{k/2}$ in time $O(\log^* n)$
- Compute new identifiers in $[k^2]$
  - Each node $u$ finds its closest anchor $i_u \in I$
  - $id'_u = (x_{i_u} - x_u, y_{i_u} - y_u)$
  - New identifiers form a $k/2$-distance coloring
- Simulate $A$ with the new identifiers in time $T(k)$
- https://jukkasuomela.fi/landscape-of-locality/


