Symmetry Breaking in Static and Dynamic Networks

Leonid Barenboim
Open University of Israel
Rate of Updates in Networks

(Estimation)

**Vertex addition/removal, edge addition/removal**

- **Social Networks (hundreds of millions of users)**
  - 10 vertices per second
  - 200 edges per second
- **Social GPS (millions of users)**
  - 5 vertices per second
  - 2,000 edges per second
- **The brain (hundred of billions of neurons)**
  - 10,000 vertices per second
  - 200,000 edges per second
A communication network is represented by a graph. Vertices have unique IDs of size $O(\log n)$ each. A message traverses an edge within one round. Running time = number of rounds to provide a solution. Update time = number of rounds to update a solution.
Network Models

Model #0  Static: Network does not change

Model #1  Dynamic single change

Model #2  Dynamic restricted change

Model #3  Dynamic unrestricted change

Step-by-step changes
Network Models

Model #0  **Static**: Network does not change

Model #1  **Dynamic single change**

Model #2  **Dynamic restricted change**

Model #3  **Dynamic unrestricted change**

Model #4  **Dynamic changes during execution**
Symmetry Breaking Problems

- Coloring
  \((\Delta + 1)\)-vertex-coloring, \((2\Delta - 1)\)-edge-coloring, defective-coloring,…
- Maximal Independent Set (MIS)
- Maximal Matching (MM)
Symmetry Breaking Problems

- Coloring
  \((\Delta+1)\)-vertex-coloring, \((2\Delta-1)\)-edge-coloring, defective-coloring,…
- Maximal Independent Set (MIS)
- Maximal Matching (MM)
Symmetry Breaking Problems

- Coloring
  \((\Delta+1)\)-vertex-coloring, \((2\Delta-1)\)-edge-coloring, defective-coloring,…
- Maximal Independent Set (MIS)
- Maximal Matching (MM)
Symmetry Breaking Problems

- Coloring
  \((\Delta+1)\)-vertex-coloring, \((2\Delta-1)\)-edge-coloring, defective-coloring,…
- Maximal Independent Set (MIS)
- Maximal Matching (MM)
Symmetry Breaking Problems

Coloring, MIS and MM belong to the class of locally-checkable problems

(Local Decision Class, Fraigniaud, Korman and Peleg 2011)
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge
- Removing a vertex or an edge
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge
- Removing a vertex or an edge
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge
- Removing a vertex or an edge
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge
- Removing a vertex or an edge
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge
- Removing a vertex or an edge
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge

- Removing a vertex or an edge
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge

- Removing a vertex or an edge
Dynamic Single Change - Coloring

Local fixing in $O(1)$ rounds

König and Wattenhofer 2013

- Adding a vertex or an edge
- Removing a vertex or an edge

This is a proper coloring, but is it a $(\Delta+1)$-coloring?
Dynamic Single Change - Coloring

Possible solution:
Delete all colors out of range \(\{1, 2, \ldots, \Delta + 1\}\), recompute solution for colorless vertices.

If a vertex leaves “gracefully” then \(O(1)\)-time solution is possible.
An MIS may consist of a single vertex.

Vertex removal may require recomputation for the entire graph.

If a vertex leaves “gracefully”, it can communicate new solution within $O(1)$ rounds.
What if vertices do not leave “gracefully”? 

- Expected O(1)-time solution
  
  Censor-hillel, Haramaty and Karnin 2016

Simulation of a greedy sequential MIS with a random ordering.
Dynamic Unrestricted Change
Theorem:
Suppose that we have a static algorithm for a locally-checkable problem on graphs with partial solution with time $T$.
Then we have a dynamic algorithm for the problem with update time $T$. 
Obtaining Dynamic Algorithms

Static Algorithm

Static Algorithm for Partial Solution

Dynamic Algorithm
Obtaining Dynamic Algorithms

Static Algorithm

Static Algorithm for Partial Solution

Dynamic Algorithm

Obtaining Static Algorithms

Dynamic Algorithm

Static Algorithm for Partial Solution

Static Algorithm
Static $O(\Delta^2)$-Coloring

Linial 1987

Running time: $O(\log^* n)$.

Very high-level description:

1. Initial $n$-coloring is obtained using IDs

2. In each round the number of colors is reduced from $k$ to $O(\Delta^2 \log k)$.

\[ n \rightarrow \Delta^2 \log n \rightarrow \Delta^2 (\log \Delta + \log \log n) \rightarrow \cdots \rightarrow \Delta^2 \log \Delta \rightarrow \Delta^2 \]
Static $O(\Delta^2)$-Coloring

- Each vertex constructs a list of colors using its current color
Static $O(\Delta^2)$-Coloring

- Each vertex constructs a list of colors using its current color
- Each list must have a color that does not appear in the neighbors lists
Static $O(\Delta^2)$-Coloring

- Each vertex constructs a list of colors using its current color.
- Each list must have a color that does not appear in the neighbors' lists.

This color is selected as the new color. New coloring is proper!
Implementing One Round

$O(\Delta^3)$ colors $\rightarrow O(\Delta^2)$ colors

Let $q = O(\Delta)$ be a prime, such that the number of colors is at most $q^3$.

There are $q^3$ distinct polynomials over the field $\mathbb{Z}_q$:

$$a + bx + cx^2 \quad \quad 0 \leq a, b, c \leq q - 1$$

Each of the $q^3$ colors is assigned a distinct polynomial.
Implementing One Round

Diagram: A network of connections with nodes labeled 105, 105, 95, 203, and 89.
Implementing One Round

Diagram:

- 95
- 105
- 105
- 203
- 89
Implementing One Round

95

105

105

203

89
Implementing One Round

For each vertex:
- At most 2 intersections with each neighbor
- At most $2\Delta$ intersections with all neighbors

Choose $q \geq 2\Delta + 1$

There is $t, 0 \leq t \leq q - 1$:
\[ < t, P(t) > \neq < t, Q(t) > \]

for all neighbors’ $Q$. 
Implementing One Round

There is \( t, 0 \leq t \leq q - 1 \):
\[
<t, P(t) > \neq < t, Q(t) >
\]
for all neighbors’ \( Q \).

\(< t, P(t) > \) is the new color.

For each pair of neighbors:
\[
<t, P(t) > \neq < r, Q(r) >
\]

Number of colors:
\[
q^2 = O(\Delta^2).
\]
Using less than $\Delta^2$ Colors

Suppose we have an orientation with out-degree $d$
Using less than $\Delta^2$ Colors

Suppose we have an orientation with out-degree $d$

Look only on outgoing neighbors. Select a color that is not in their lists.
Using less than $\Delta^2$ Colors

Suppose we have an orientation with out-degree $d$

Look only on outgoing neighbors. Select a color that is not in their lists.

$O(d^2)$-coloring is computed in $O(\log^* n)$ time.
Using less than $\Delta^2$ Colors

Suppose we have an orientation with out-degree $d$.

Look only on outgoing neighbors. Select a color that is not in their lists.

$O(d^2)$-coloring is computed in $O(\log^* n)$ time.

Arboricity $a$ is the minimum number of forests.
Using less than $\Delta^2$ Colors

Suppose we have an orientation with out-degree $d$

Look only on outgoing neighbors. Select a color that is not in their lists.

$O(d^2)$-coloring is computed in $O(\log^* n)$ time.

Arboricity $a$ is the minimum number of forests.

$O(a)$-orientation in $O(\log n)$ time.  Barenboim and Elkin 08.
Orientations with Small Out-Degree

If we have an orientation with \( d \leq \sqrt{\Delta} \), we can compute \( O(\Delta) \)-coloring in \( O(\log^* n) \) time!

Small out-degree orientation does not always exist.

Partition the graph into \( \sim\sqrt{\Delta} \) vertex-disjoint subgraphs, each subgraph with out-degree \( O(\sqrt{\Delta}) \).

Color subgraphs one by one - \( O(\log^* n) \) time per subgraph.
Graph Partition

$G_1$, $G_2$, $G_3$, ..., $G_{\sqrt{\Delta}}$
Graph Partition

Each subgraph is properly colored.
Graph Partition

Each subgraph is properly colored.
Graph Partition

Each subgraph is properly colored.
Graph Partition

Each subgraph is properly colored.

Problem: monochromatic edges between subgraphs.
Solution: make it work in partially colored graphs.
Coloring Partially-Colored Graphs

\[ G \leq \sqrt{\Delta} \]

\[ G_i \]
Each vertex may have up to $\Delta$ colored neighbors.

Each color is a forbidden coordinate $< x, f(x) >$.

**Problem:** The size of the field is only $O(\sqrt{\Delta})$.

**Solution:**
Each vertex defines $O(\sqrt{\Delta})$ non-intersecting polynomials.
Then we can find a polynomial with a good coordinate.
Coloring Partially-Colored Graphs

Find a polynomial with minimum number of conflicts

\[ \sqrt{\Delta} \leq q = O(\sqrt{\Delta}) \]
Coloring Partially-Colored Graphs

How to determine the coefficients $a$ and $b$?

Using a helper temporary $O(\Delta)$-coloring of $G_i$. 
Coloring Partially-Colored Graphs

\[ u \leq \Delta \]

\[ G_1 \]

\[ G_{i-1} \]

\[ G_i \]

\[ G_{i+1} \]
Coloring Partially-Colored Graphs

\[ u \leq \sqrt{\Delta} \]

\[ 3x + 4x^2 \]
\[ 1 + 3x + 4x^2 \]
\[ 2 + 3x + 4x^2 \]
\[ \ldots \]

\[ 7x + 4x^2 \]
\[ 1 + 7x + 4x^2 \]
\[ 2 + 7x + 4x^2 \]
\[ \ldots \]
Coloring Partially-Colored Graphs

- Let \( G_0 = (V_0, E_0) \) denote the subgraph of colored vertices.
- Execute our algorithm on \( V \setminus V_0 \), and avoid conflicts with \( V_0 \).
Dynamic Algorithm

In each step (addition of vertices or edges, removal of vertices or edges):

1. Perform local fixing to obtain a partial solution
2. Invoke static algorithm for partial solution
Dynamic Algorithm

In each step (addition of vertices or edges, removal of vertices or edges):

1. Perform local fixing to obtain a partial solution

2. Invoke static algorithm for partial solution
Dynamic Algorithm

In each step (addition of vertices or edges, removal of vertices or edges):

1. Perform local fixing to obtain a partial solution

2. Invoke static algorithm for partial solution
Dynamic Algorithm

In each step (addition of vertices or edges, removal of vertices or edges):

1. Perform local fixing to obtain a partial solution
2. Invoke static algorithm for partial solution
Static Algorithm for List-Coloring

Input:
Each vertex receives as input a list of at least $\Delta + 1$ colors from a range of size $D = O(\Delta)$.

Output:
Each vertex selects a color from its list to obtain a proper coloring.
Solution: a reduction from list coloring to coloring partially-colored graphs

Add neighbors with colors that are not in the lists

New maximum degree: at most D-1

Static Algorithm for List-Coloring
Conclusion

• Static algorithms for graphs with partial solution yield dynamic algorithms.

• Static algorithms for graphs with partial solution are known for:
  • Coloring: $\sim O(\sqrt{\Delta} + \log^* n)$ time.
  • Maximal Independent Set: $O(\Delta + \log^* n)$ time.
  • Maximal Matching: $O(\Delta + \log^* n)$ time.
  • ...

• We obtain dynamic algorithms for these problems with the same update time.

Can we do better than that?
Conclusion

• In these dynamic settings changes occur in steps.

• During an execution of an algorithm no changes occur.

Can algorithms cope with changes during their execution?
Thank you!