Distributed Computing
11 - LOCAL Algorithms

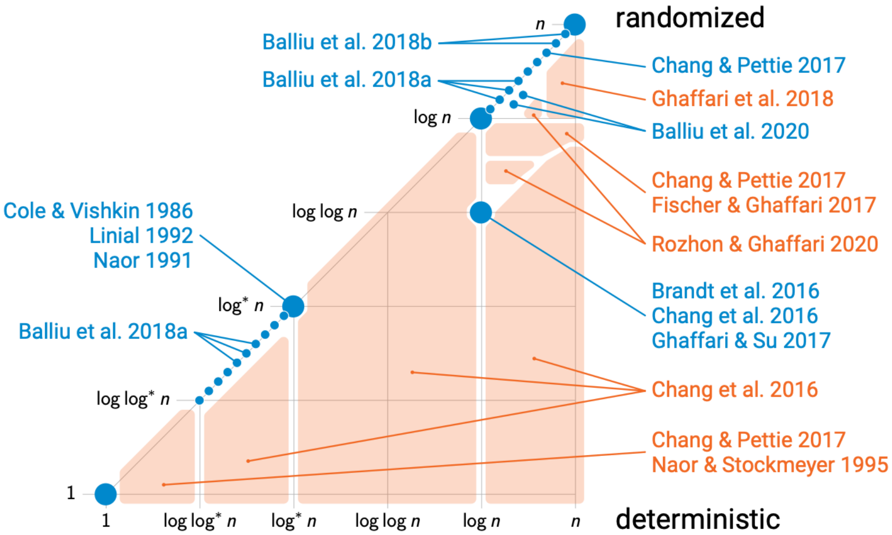
Mikaël Rabie
Université Paris-Cité, IRIF

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Université de Paris

Creating Complexities

Complexities on Graphs with the LOCAL Model

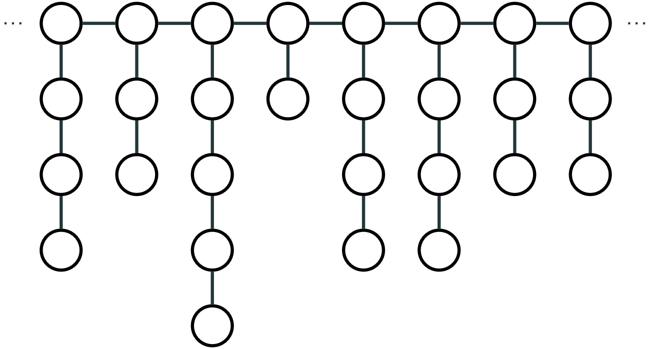


Figures from https ://jukkasuomela.fi/landscape-of-locality/

Infinite Number of Complexities

Chang, Pettie (2019)
For any $k > 0$, there exists a problem P_k that is solvable in time $\Theta(n^{1/k})$.

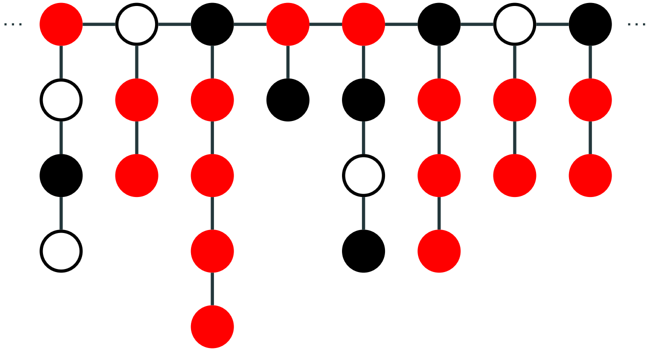
$2\frac{1}{2}$-coloring Example



Comb Tree : A path V_2 of degree 3 nodes. To each of those nodes there is a path appended.
For each node $v \in V_2$ either :

- Its appended path is 2-colored ($v \in D_2$)
- It is 2-colored with its $V_2 \setminus D_2$ neighbors

$2\frac{1}{2}$-coloring Example



Comb Tree : A path V_2 of degree 3 nodes. To each of those nodes there is a path appended.

For each node $v \in V_2$ either :

- Its appended path is 2-colored ($v \in D_2$)
- It is 2-colored with its $V_2 \setminus D_2$ neighbors

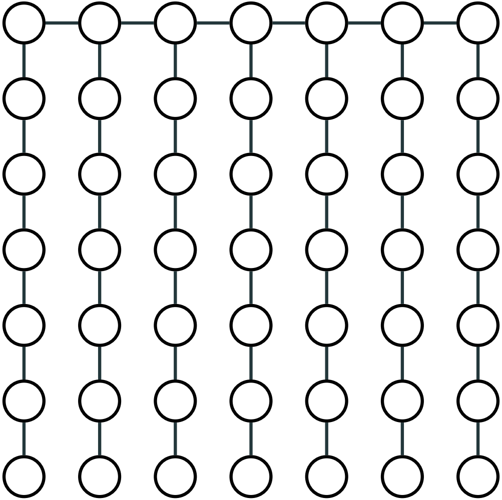
Question : Find a $O(\sqrt{n})$ algorithm.

Worst Case Graph

Question : On which graphs the problem needs $\Omega(\sqrt{n})$?

Worst Case Graph

Question : On which graphs the problem needs $\Omega(\sqrt{n})$?



Generalization to any graphs

- G_1 : input graph
- $V_1 = \{v \in V(G_1) : degree(v) \leq 2\}$
- $G_2 = G_1 \setminus V_1$
- $V_2 = \{v \in V(G_2) : degree(v) \leq 2\}$
- V_1 nodes accept if, in regard to its neighbors in V_1 :
  - They are 2-colored with colors 1 and 2
  - They all are in color 3
  - They are on an oriented cycle
- $D_2 = \{v \in V_2 : \text{its } V_1 \text{ neighbors are 2-colored}\}$
  - D_2 nodes accept if they are in color 4
- $V_2 \setminus D_2$ nodes accept if, in regard to its neighbors in $V_2 \setminus D_2$:
  - They are 2-colored with colors 1 and 2
  - They have 2 neighbors and they all are in color 3
  - They are on an oriented cycle

Generalization to $n^{1/k}$

- G_1 : input graph
- $V_i = \{v \in V(G_i) : degree(v) \leq 2\}$
- $G_i = G_{i-1} \setminus V_{i-1}$
- $V_{k+1} = V(G_{k+1})$
- A vertex in $V(G)$ is **exempted** if
  - It has a lower level exempted neighbor
  - It has a lower level 2-colored neighbor
  - It is on an oriented cycle on its level
  - It is in V_{k+1}
- $D_i \subseteq V_i$ are the exempted nodes of level i
- $V_i \setminus D_i$ nodes ($i \leq k$) accept if, in regard to its neighbors in $V_i \setminus D_i$:
  - They are 2-colored with colors 1 and 2
  - They all are in color 3 (and have degree=2 if $i = k$)
  - They are on an oriented cycle

Maximal Matchings

Lower Bounds

Balliu *et. al* (2019)

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

Linial (1992)

An algorithm which colors the n-cycle with three colors requires at least $\frac{1}{2}(\log^* n - 3)$ communication rounds.

$\Rightarrow$ Maximal Matching needs $\Omega(\log^* n)$ rounds in the LOCAL Model.

Lower Bounds

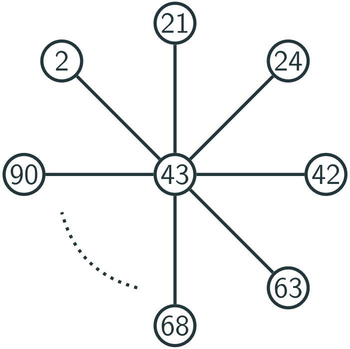
Balliu *et. al* (2019)
Maximal Matching needs $\Omega(min\{\Delta, \log n / \log \log n\})$ rounds in the LOCAL Model.

Linial (1992)
An algorithm which colors the n-cycle with three colors requires at least $\frac{1}{2}(\log^* n - 3)$ communication rounds.

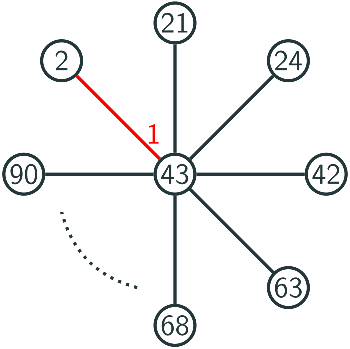
$\Rightarrow$ Maximal Matching needs $\Omega(\log^* n)$ rounds in the LOCAL Model.

Panconesi, Rizzi (2001)
There exists a LCL algorithm to produce a Maximal Matching in $O(\log^* n + \Delta)$ communications.

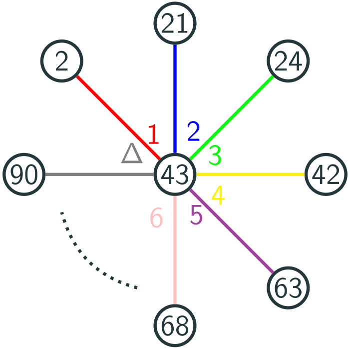
Partition into Rooted Trees



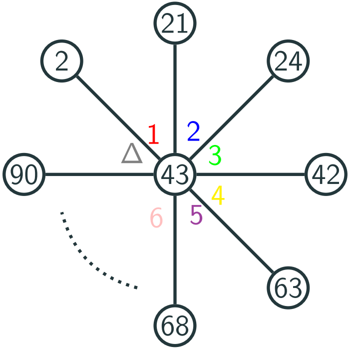
Partition into Rooted Trees

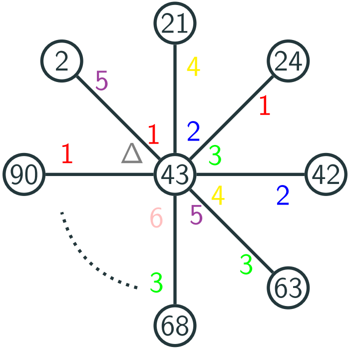


Partition into Rooted Trees

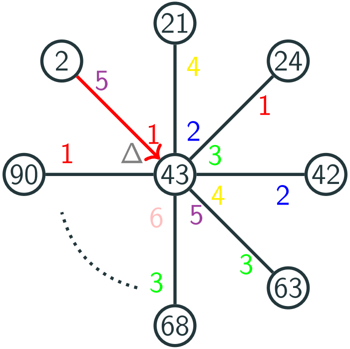


Partition into Rooted Trees

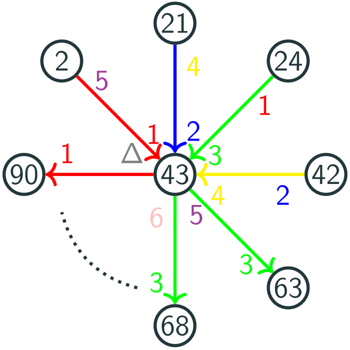



21
2
24
90
43
42
63
68
5
4
1
1
2
1
3
2
4
5
6
3
3
△

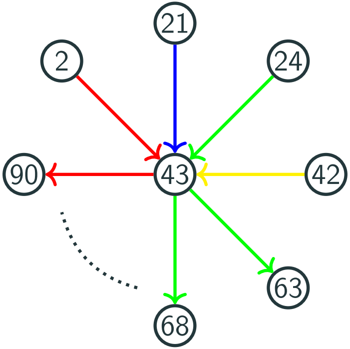
Partition into Rooted Trees



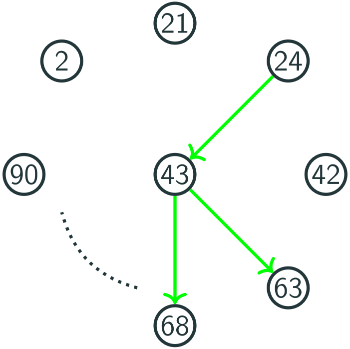
Partition into Rooted Trees



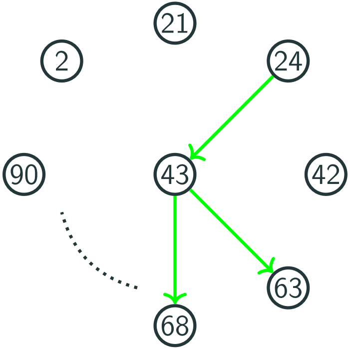
Partition into Rooted Trees



Partition into Rooted Trees

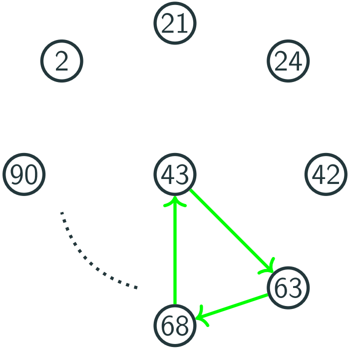


Partition into Rooted Trees



- At most one parent

Partition into Rooted Trees



- At most one parent
- No cycles

Maximal Matching Algorithm

1. Partition G into Δ Rooted Forests.

2. For each Forest, 3-color it in $O(\log^* n)$ communications (in parallel)

Maximal Matching Algorithm

1. Partition G into Δ Rooted Forests.

2. For each Forest, 3-color it in $O(\log^* n)$ communications (in parallel)

3. For Forest $f = 1$ to Δ do :
   - For Color $c = 1$ to 3 do :
     - For all node, add if possible an edge connecting it to one of its children of Color c in Forest f.

3-Coloring Trees

3-Coloring in $O(\log n)$

Bonamy *et. al* (2018)
There exists an algorithm that 3-colors trees in $O(\log n)$ rounds.

Tree Shattering
There exists an algorithm that computes an MIS in trees in $O(\log n)$ rounds. This MIS is such that, after its removal, connected components have size 1 or 2.

Bonamy *et. al* (2018)
There exists an algorithm that 3-colors trees in $O(\log n)$ rounds.

Tree Shattering
There exists an algorithm that computes an MIS in trees in $O(\log n)$ rounds. This MIS is such that, after its removal, connected components have size 1 or 2.

- Build a $\mathcal{O}(\log n)$ light labeling of the nodes :
  1. Any node labeled i has at most two neighbors with label $\geq i$, at most one of which with label $\geq i+1$
  2. No two adjacent nodes labeled i both have a neighbor with label $\geq i+1$

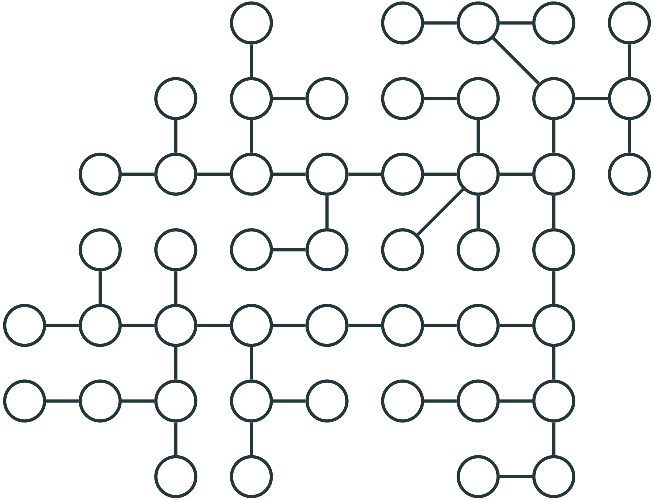
3-Coloring in $O(\log n)$

Bonamy *et. al* (2018)
There exists an algorithm that 3-colors trees in $O(\log n)$ rounds.

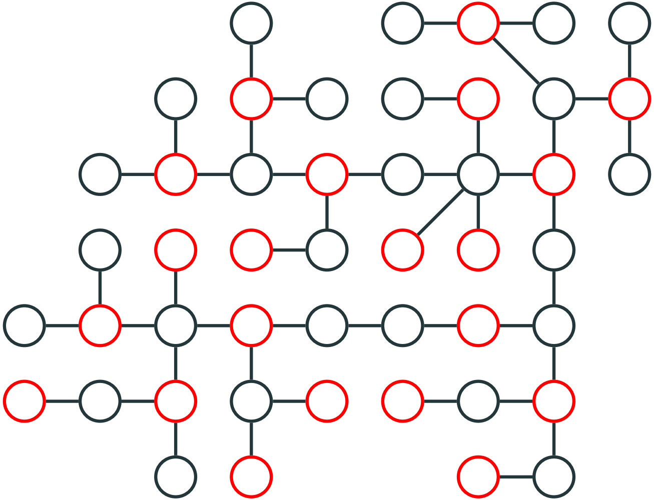
Tree Shattering
There exists an algorithm that computes an MIS in trees in $O(\log n)$ rounds. This MIS is such that, after its removal, connected components have size 1 or 2.

- Build a $\mathcal{O}(\log n)$ light labeling of the nodes :
  1. Any node labeled i has at most two neighbors with label $\geq i$, at most one of which with label $\geq i+1$
  2. No two adjacent nodes labeled i both have a neighbor with label $\geq i+1$
- Compute a 3-coloring for each paths of nodes with the same label
- Build the MIS by considering labels decreasingly.

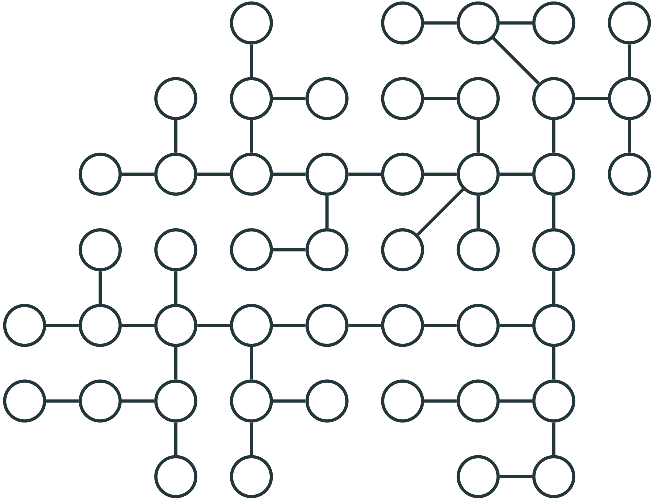
Tree Shattering



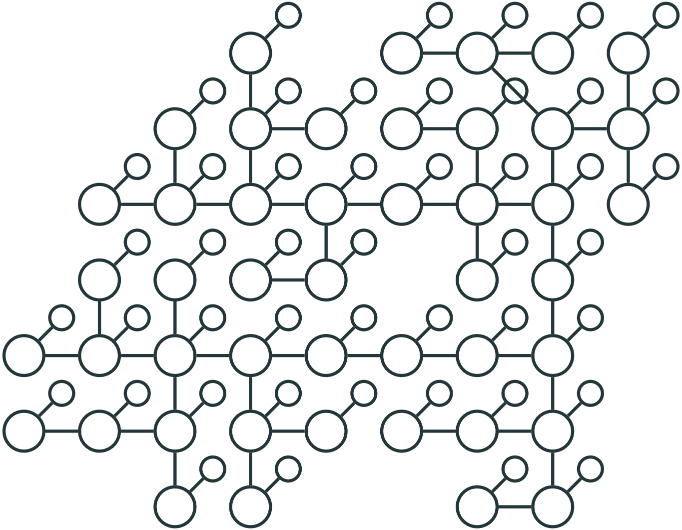
Tree Shattering



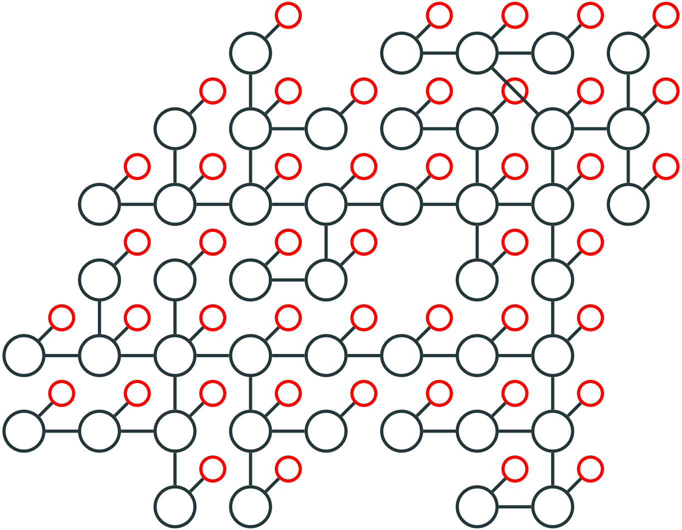
Tree Shattering



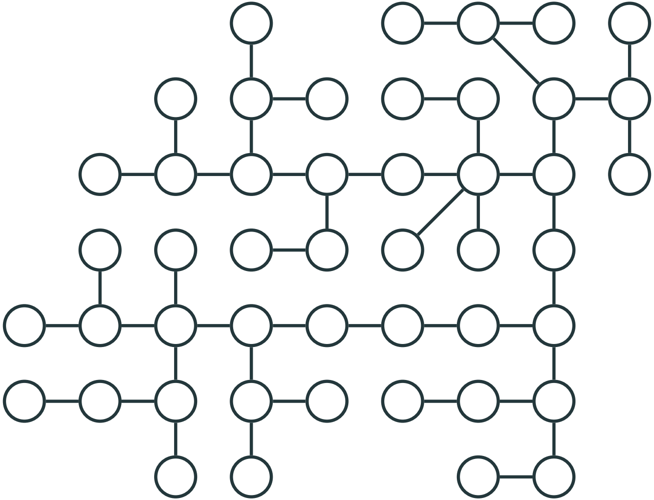
Tree Shattering

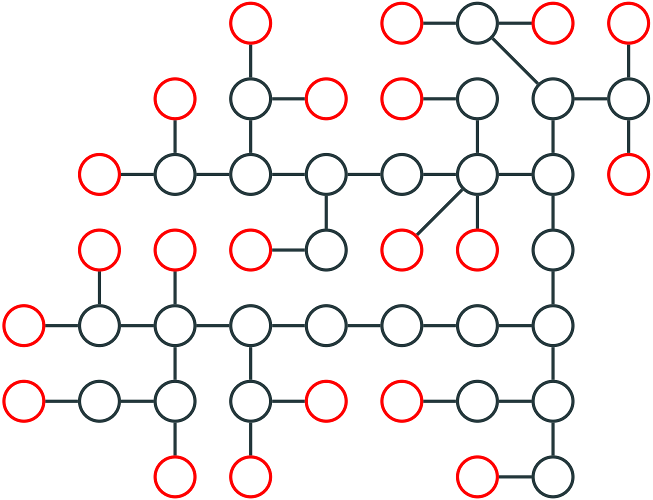


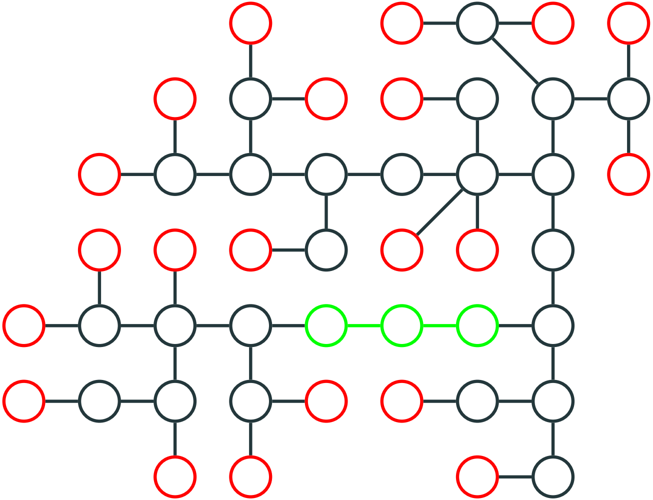
Tree Shattering



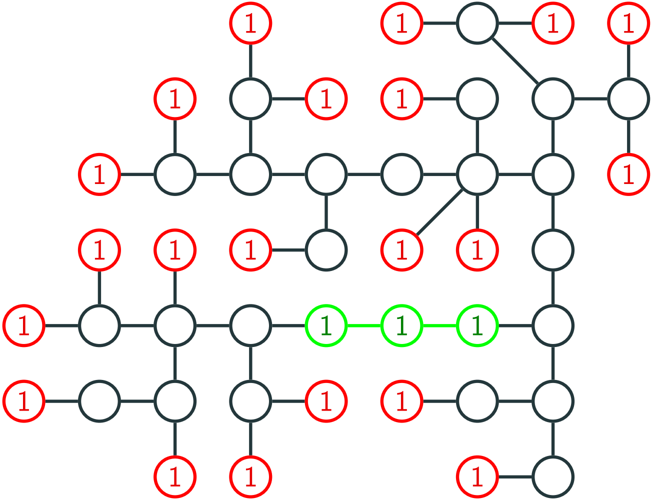
Tree Shattering



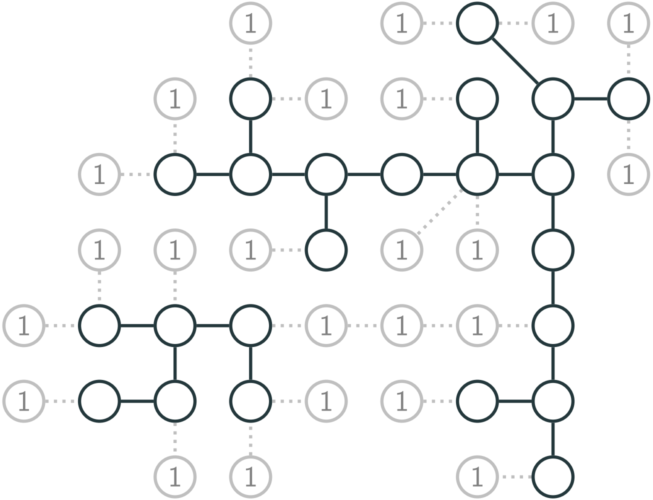




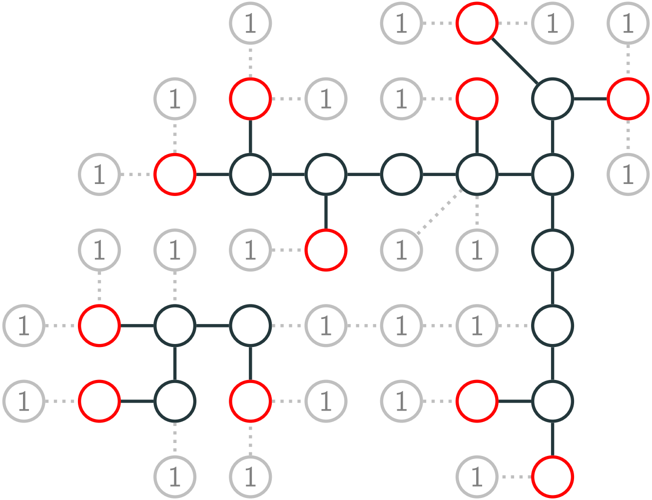
Tree Shattering



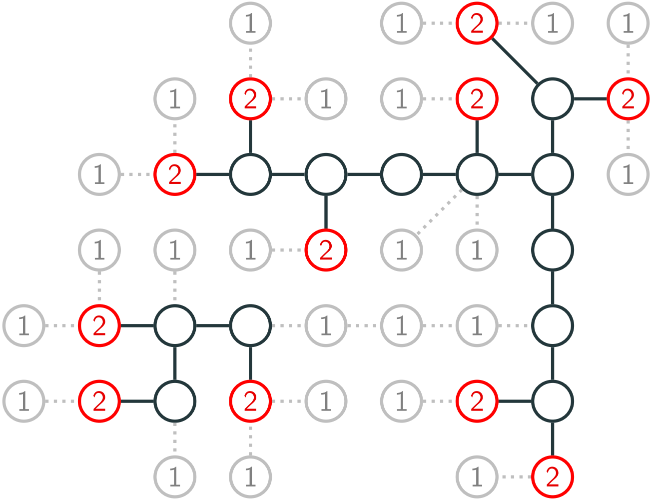
Tree Shattering



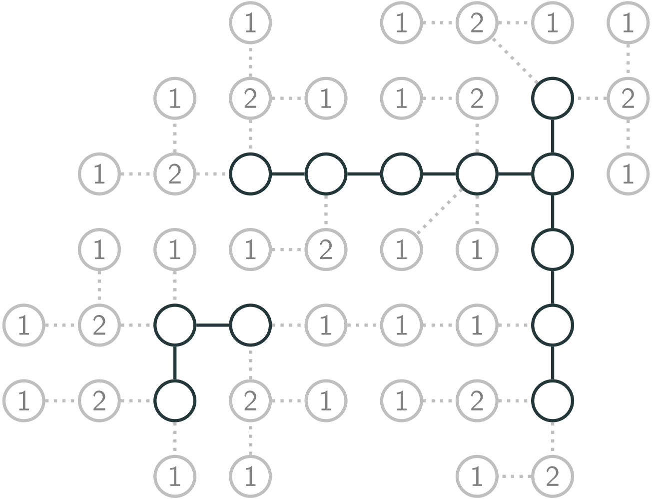
Tree Shattering



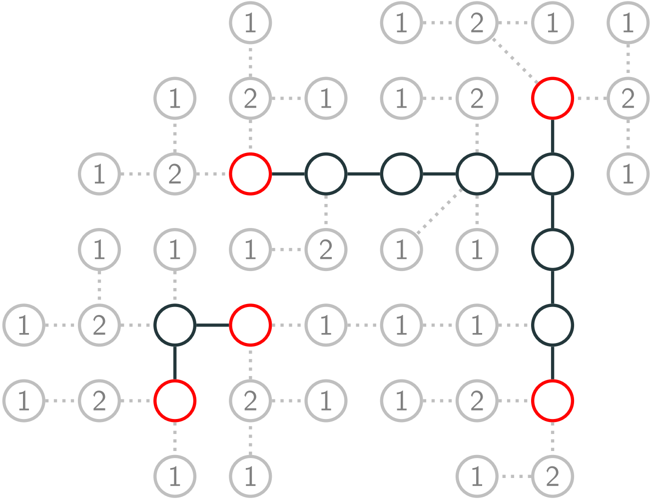
Tree Shattering



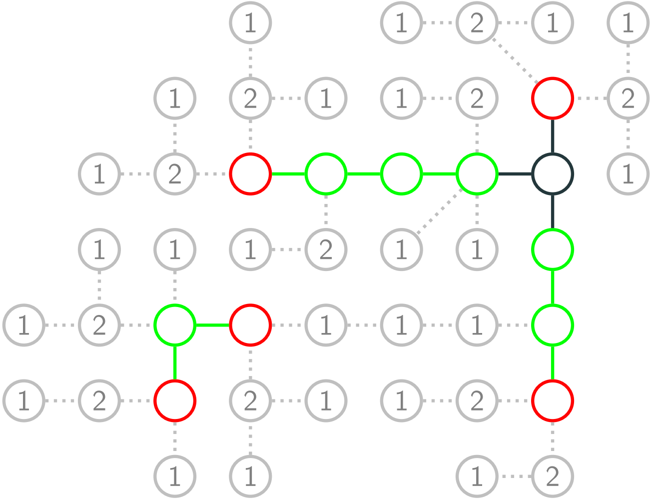
Tree Shattering



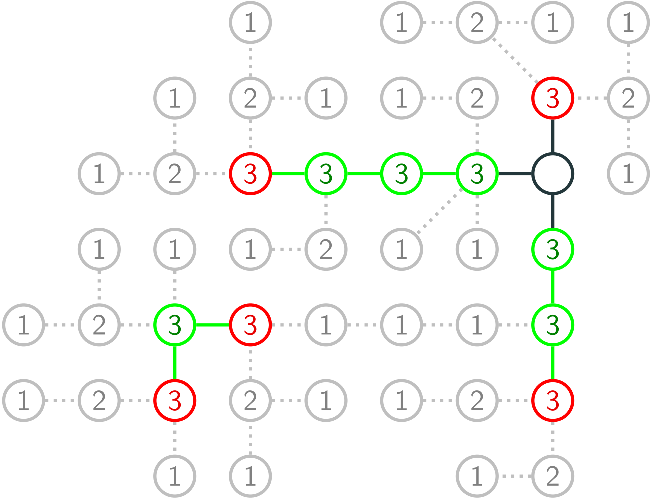
Tree Shattering

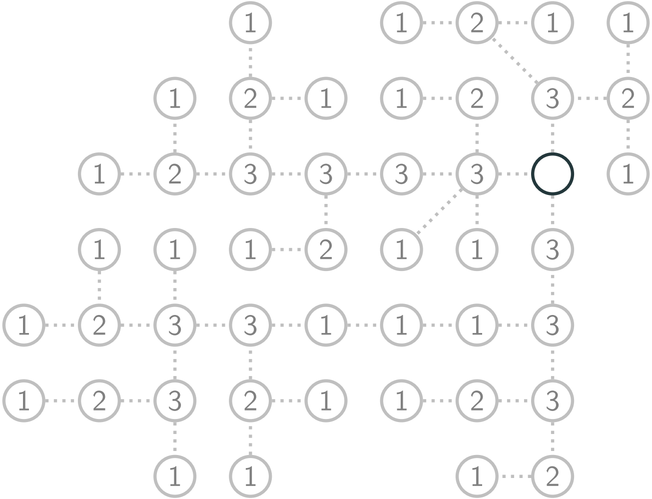


Tree Shattering

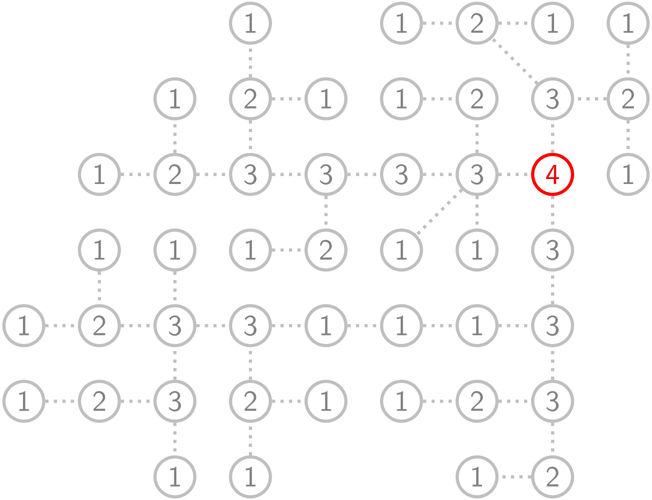


Tree Shattering



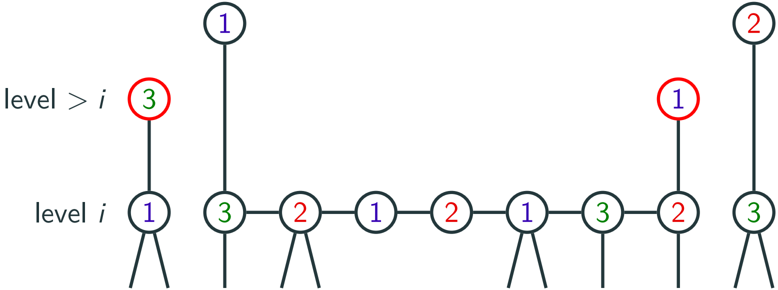

1
1
2
1
1
1
2
1
1
2
1
1
2
3
2
1
2
3
3
3
3
1
1
1
1
2
1
1
3
1
2
3
3
1
1
1
3
1
2
3
2
1
1
2
3
1
1
1
2

Tree Shattering



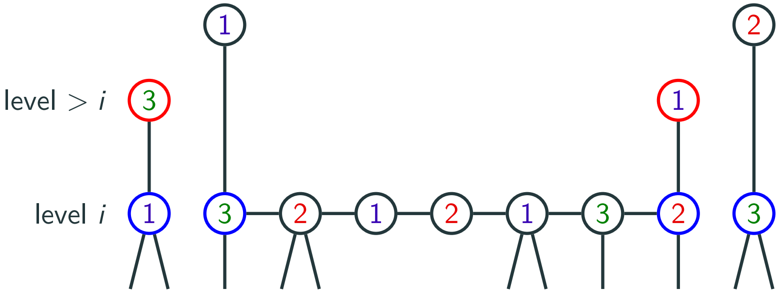
Constructing the Maximal Independent Set

- In parallel, compute a 3-coloring on each level
- From higher to lower level
- Add, if possible, neighbors to higher level nodes
- Greedily complete the MIS



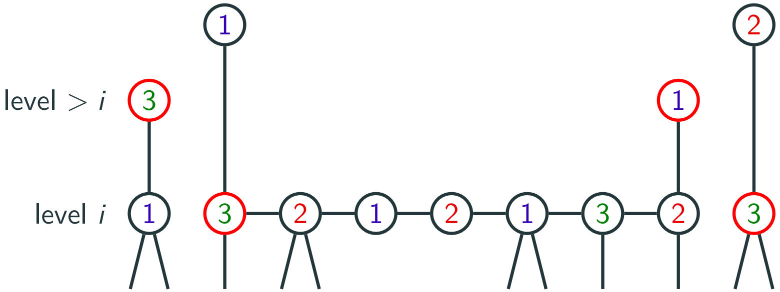
Constructing the Maximal Independent Set

- In parallel, compute a 3-coloring on each level
- From higher to lower level
- Add, if possible, neighbors to higher level nodes
- Greedily complete the MIS



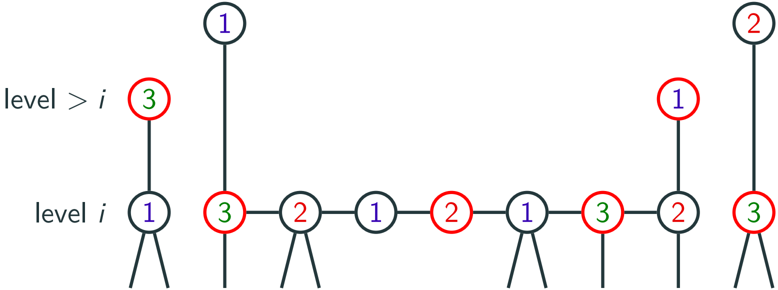
Constructing the Maximal Independent Set

- In parallel, compute a 3-coloring on each level
- From higher to lower level
- Add, if possible, neighbors to higher level nodes
- Greedily complete the MIS



Constructing the Maximal Independent Set

- In parallel, compute a 3-coloring on each level
- From higher to lower level
- Add, if possible, neighbors to higher level nodes
- Greedily complete the MIS



Bibliography

- https ://jukkasuomela.fi/landscape-of-locality/
- Marthe Bonamy, Paul Ouvrard, Mikaël Rabie, Jukka Suomela, Jara Uitto. **Distributed Recoloring**. DISC 2018.
- Yi-Jun Chang, Seth Pettie. **A Time Hierarchy Theorem for the LOCAL Model**. In SIAM J. Comput. 48, 2019.
- Nathan Linial. **Locality in distributed graph algorithms**. SIAM Journal on Computing, 1992.
- Alessandro Panconesi, Romeo Rizzi. **Some simple distributed algorithms for sparse networks**. In Distributed Comput., 2001.