Distributed Computing
13 - LOCAL Variants

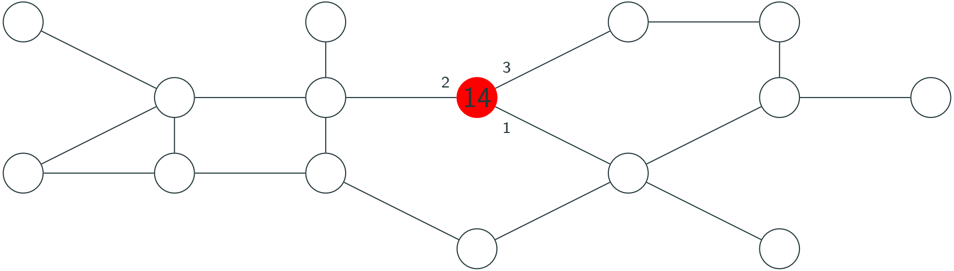
Mikaël Rabie
Université Paris Cité, IRIF

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

Volume Complexity

CentLOCAL Model

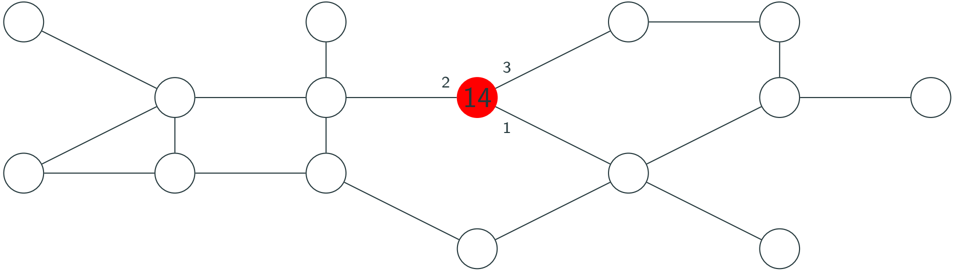
- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node



CentLOCAL Model

- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node

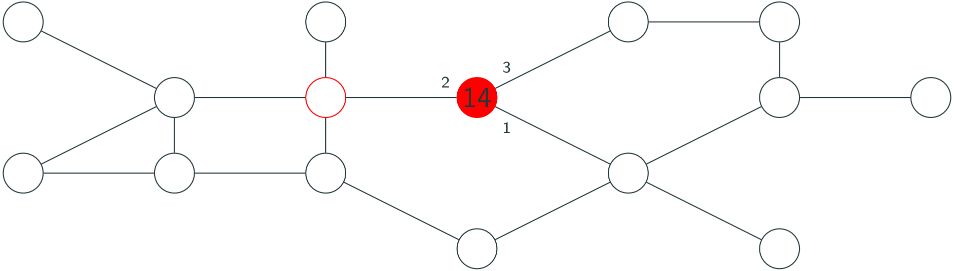
Request : (14,2)



CentLOCAL Model

- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node

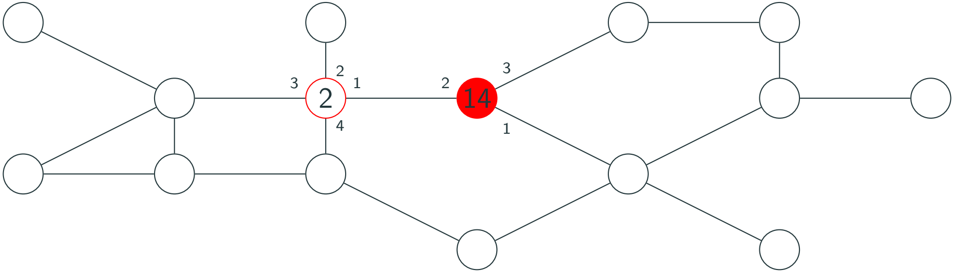
Request : (14,2)



CentLOCAL Model

- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node

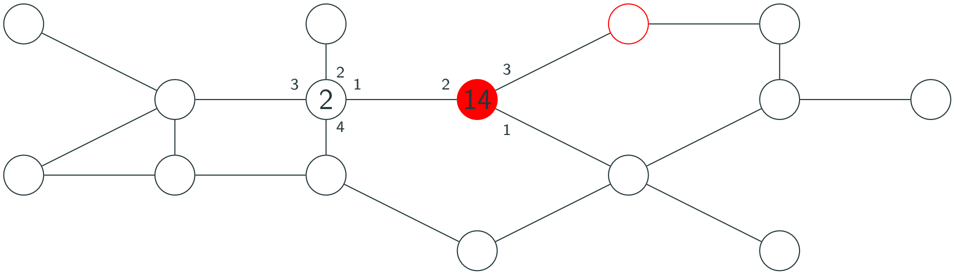
Request : (14,2) $\Rightarrow$ (2,4,1)



CentLOCAL Model

- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node

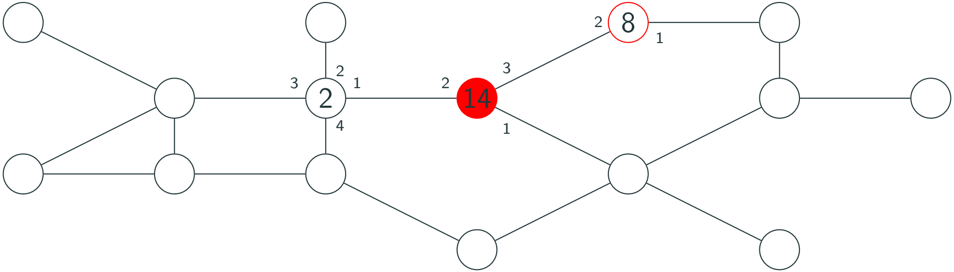
Request : (14,3)



CentLOCAL Model

- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node

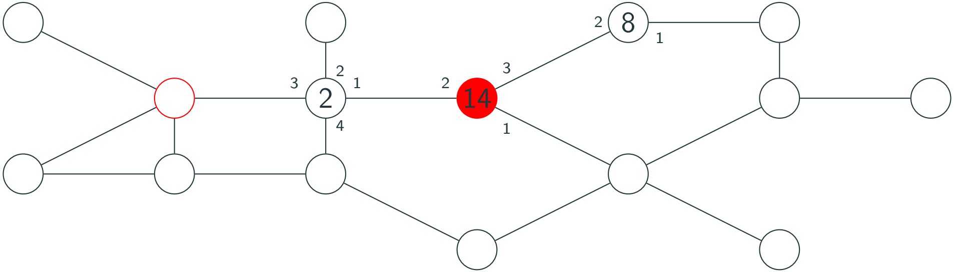
Request : (14,3) $\Rightarrow$ (8,2,2)



CentLOCAL Model

- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node

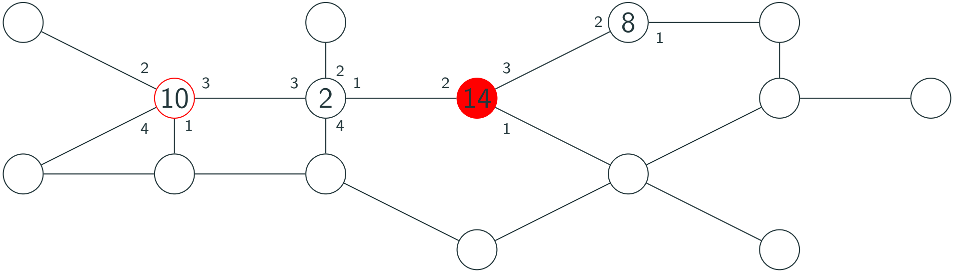
Request : (2,3)



CentLOCAL Model

- In parallel, each node v :
  - Knows its own Id_v and degree d_{Id_v}
  - At each step, they send a request (Id_u, k), with $k \leq d_{Id_u}$
  - They get (Id_w, d_{Id_w}, k') such that $(u, v) \in E$ are connected by port k from u and k' from w
- Complexity : maximal number of requests from a node

Request : (2,3) $\Rightarrow$ (10,4,3)



Greedy Problems

Problem A can be solved in time $\Theta(f(n))$ in the LOCAL model

$\Rightarrow$ A can be solved in time in the CentLOCAL model

Greedy Problems

Problem A can be solved in time $\Theta(f(n))$ in the LOCAL model

$$\Rightarrow A \text{ can be solved in time } \Omega(f(n)) \text{ and } O\left(\Delta^{f(n)}\right) \text{ in the CentLOCAL model}$$

Greedy Problems

Problem A can be solved in time $\Theta(f(n))$ in the LOCAL model

$\Rightarrow$ A can be solved in time $\Omega(f(n))$ and $O\left(\Delta^{f(n)}\right)$ in the CentLOCAL model

Even *et. al* (2018)

There is a CentLOCAL algorithm in time $O(\Delta \times log^* n + \Delta^3)$ for $\leq \Delta^2$-coloring a graph.

There is a CentLOCAL algorithm in time $O(\Delta \times log^* n + \Delta^3)$ for orienting a graph where the longer oriented path is of length $\leq \Delta^2$.

Any greedy problem can be solved in time $O(f(\Delta) \times log^* n)$.

Complexity Gap

Rosenbaum and Suomela (2020)
In the CentLOCAL model, if n is not given in advance and identifiers do not require to be polynomial in n, there is no problem whose time complexity is in $\omega(log^* n) \cap o(n)$.

Complexity Gap

Rosenbaum and Suomela (2020)
In the CentLOCAL model, if n is not given in advance and identifiers do not require to be polynomial in n, there is no problem whose time complexity is in $\omega(log^* n) \cap o(n)$.

- Take N such that $T(N) \ll N$
- Do a distance N-coloring
- Simulate the algorithm with the new identifiers

Mendability

Mendable Problems

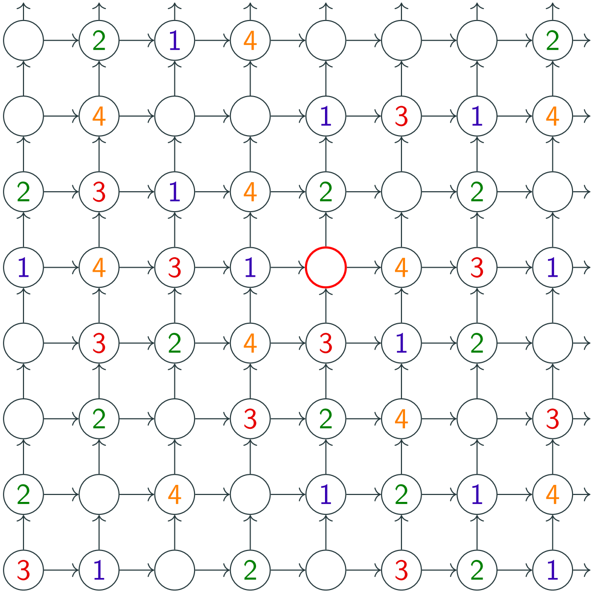
$\Gamma^* : V \to \mathcal{O} \cup \{\perp\}$ is a **Partial Solution** if :

- $\mathcal{O}$ is the Output Set,
- $\forall u \in V : \Gamma^*(u) \neq \perp \Rightarrow$ we can complete the labels of the neighbors of u.

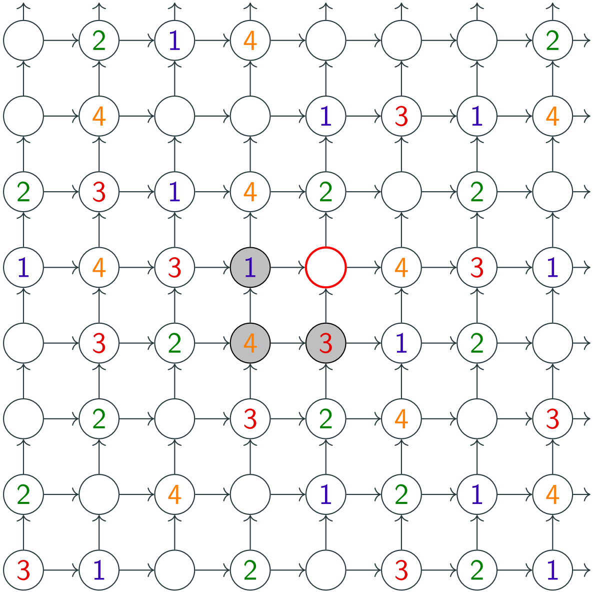
A problem is T-**Mendable** if, from any partial solution Γ^* and any $v \in V$ such that $\Gamma^*(v) = \perp$, there exists Γ' :

- $\Gamma'(v) \neq \perp$
- $\forall u \neq v,\ \Gamma'(u) = \perp \Leftrightarrow \Gamma^*(u) = \perp$
- $\forall u \in V,\ dist(u, v) > T \Rightarrow \Gamma'(u) = \Gamma^*(u)$

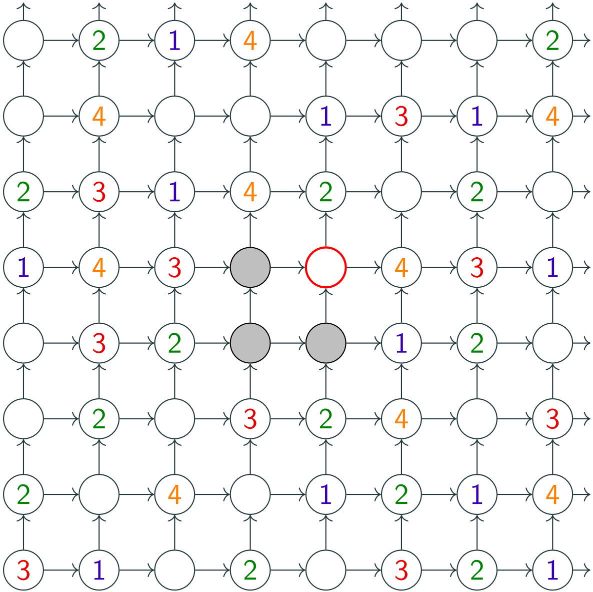
4-coloring the Grid



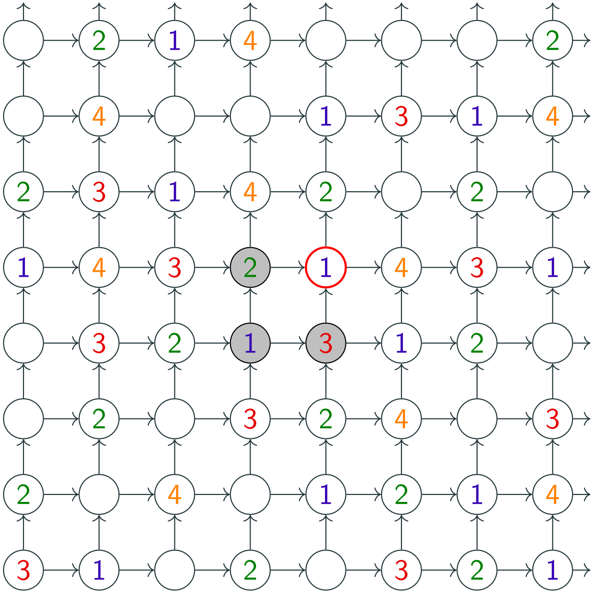
4-coloring the Grid



4-coloring the Grid



4-coloring the Grid



Mendable into LOCAL

Balliu *et. al* (2022)

Let Π be a T-mendable LCL problem. Π can be solved in rounds in the LOCAL model if we are given a distance-$(2T+1)$ coloring.

Mendable into LOCAL

Balliu ***et. al*** **(2022)**

Let Π be a T-mendable LCL problem. Π can be solved in $O\left(T\Delta^{2T}\right)$ rounds in the LOCAL model if we are given a distance-$(2T+1)$ coloring.

Mendable into LOCAL

Balliu *et. al* (2022)
Let Π be a T-mendable LCL problem. Π can be solved in $O\left(T\Delta^{2T}\right)$ rounds in the LOCAL model if we are given a distance-$(2T+1)$ coloring.

Balliu *et. al* (2022)
Let Π be a $O(1)$-mendable LCL problem. Π can be solved in $O\left(\log^* n\right)$ rounds in the LOCAL model on bounded degree graphs.

From $\log^* n$ to Mendability

On paths and cycles, are all $O(\log^* n)$ problems mendable?

From $\log^* n$ to Mendability

On paths and cycles, are all $O(\log^* n)$ problems mendable ?

No : 3-color with $\{1, 2, 3\}$ or 2-color with $\{A, B\}$.

From $\log^* n$ to Mendability

On paths and cycles, are all $O(\log^* n)$ problems mendable ?

No : 3-color with $\{1, 2, 3\}$ or 2-color with $\{A, B\}$.

Balliu *et. al* **(2022)**
Suppose Π is an LCL problem on directed cycles with no input. If Π is $O(\log^* n)$-solvable, we can define a new LCL problem Π' with the same round complexity, such that a solution for Π' is also a solution for Π, and Π' is $O(1)$-mendable.

The Case of Trees

Balliu *et. al* **(2022)**
In trees, there are exactly three classes : $O(1)$-mendable, $\Theta(\log n)$-mendable, and $\Theta(n)$-mendable problems.

3-coloring the rooted tree is

The Case of Trees

Balliu *et. al* **(2022)**
In trees, there are exactly three classes : $O(1)$-mendable, $\Theta(\log n)$-mendable, and $\Theta(n)$-mendable problems.

3-coloring the rooted tree is $O(n)$-mendable.
There exists a $O(1)$-mendable problem Π' that projects its solutions to a 3-coloring :

- A node is **monochromatic** if both its children have the same color.
- Otherwise, the node is **mixed**.
- Π' only accept connected components of mixed nodes of height $\leq k$.

Landscape of LOCAL Variants

LOCAL Variants landscape

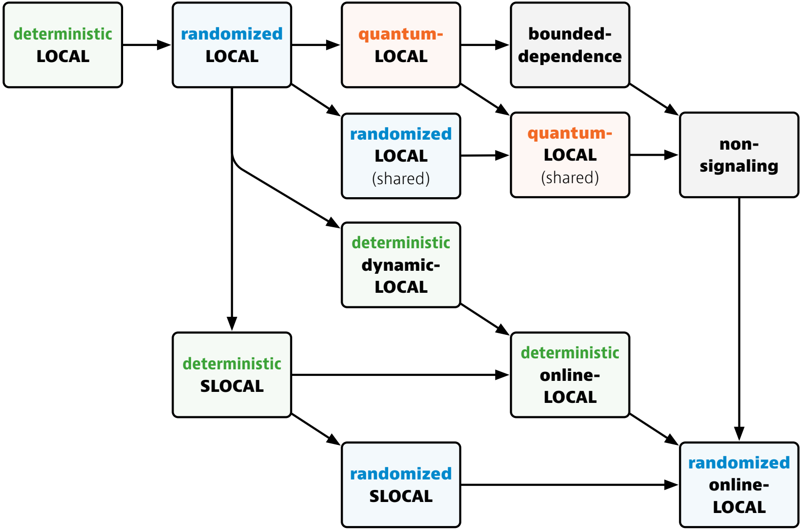


Figure from https ://jukkasuomela.fi/doc/meascomb-2024-06-27.pdf

LOCAL Variants landscape

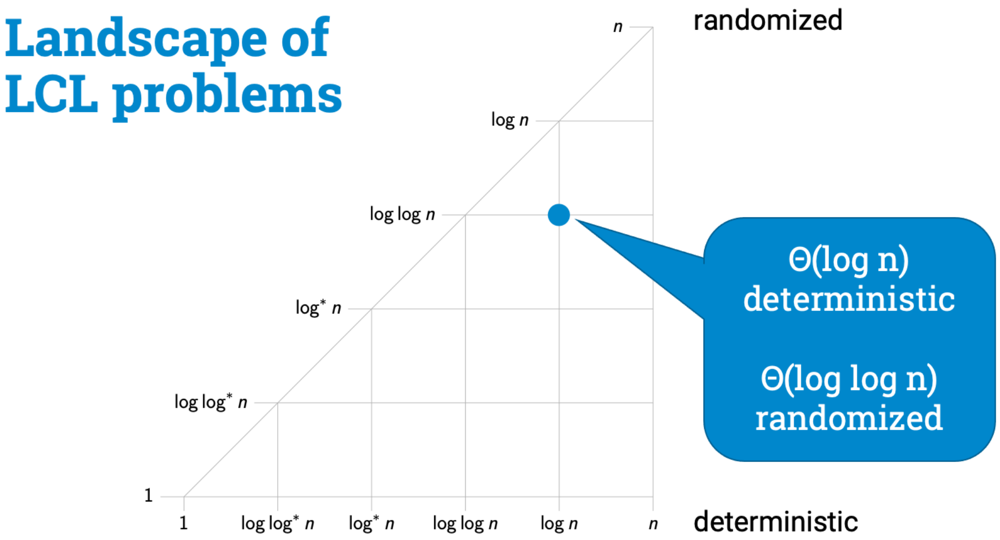
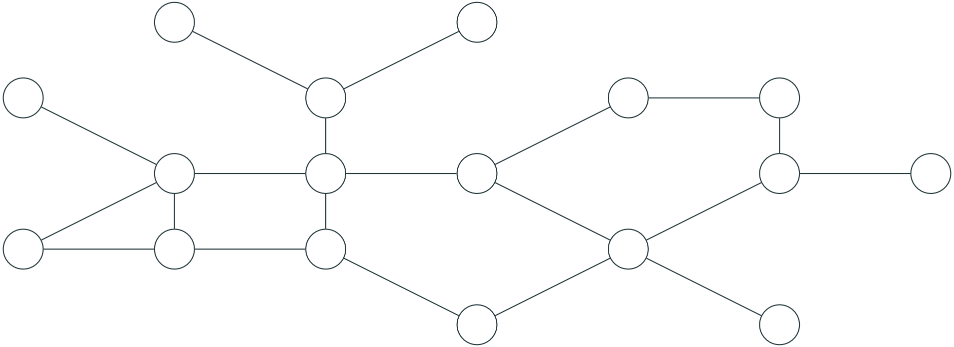


Figure from https ://jukkasuomela.fi/landscape-of-locality/

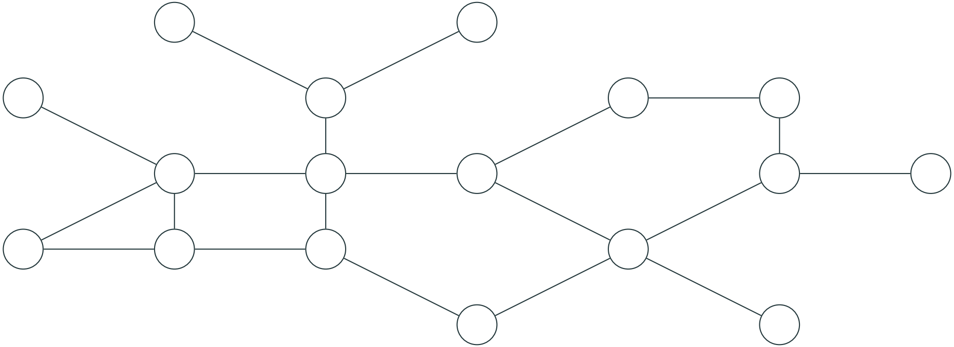
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes



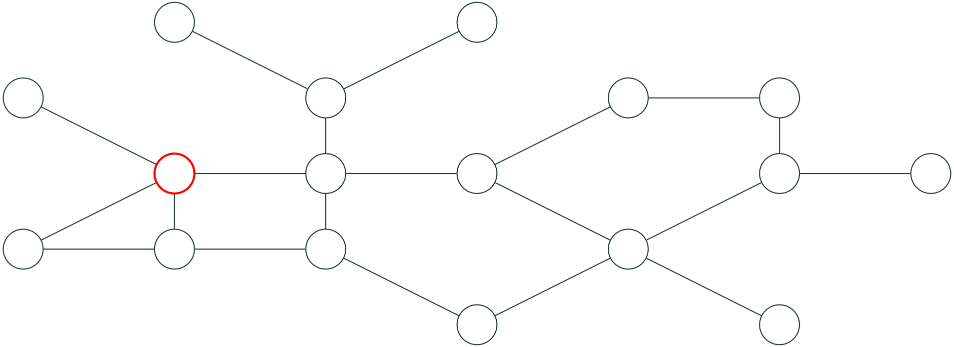
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



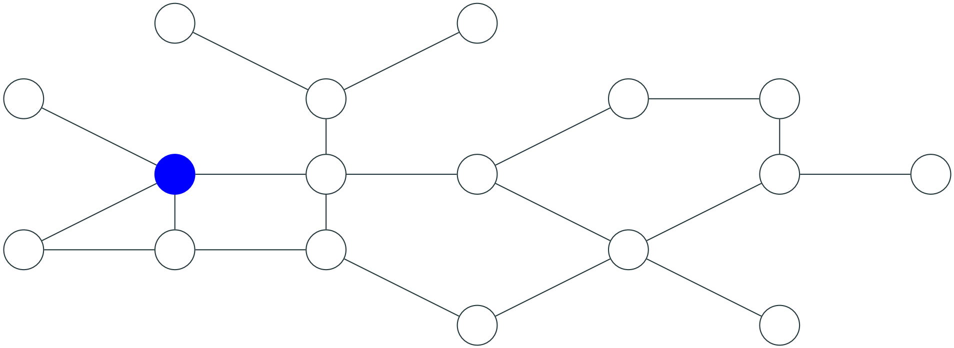
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



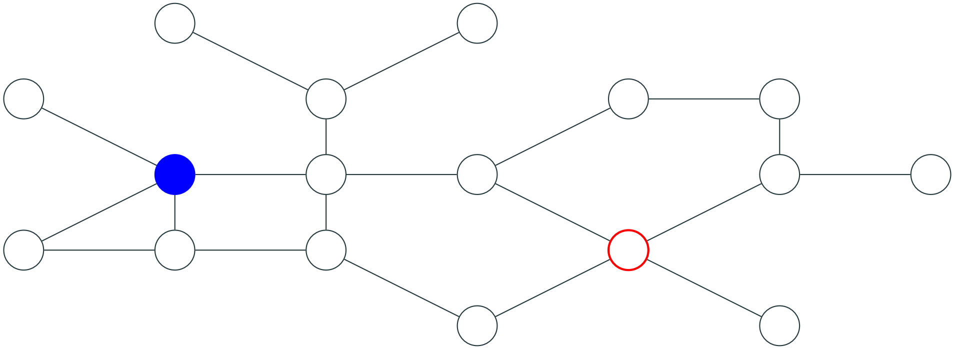
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



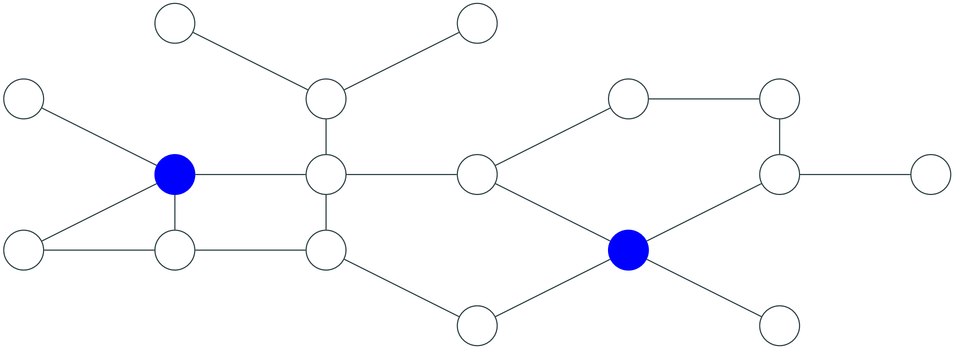
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



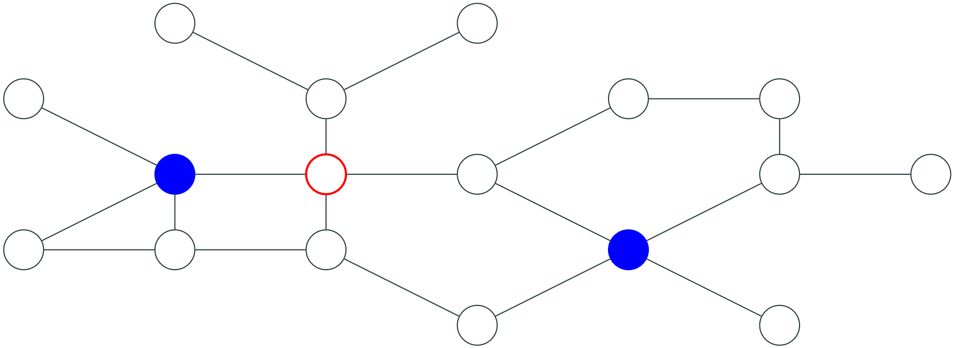
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



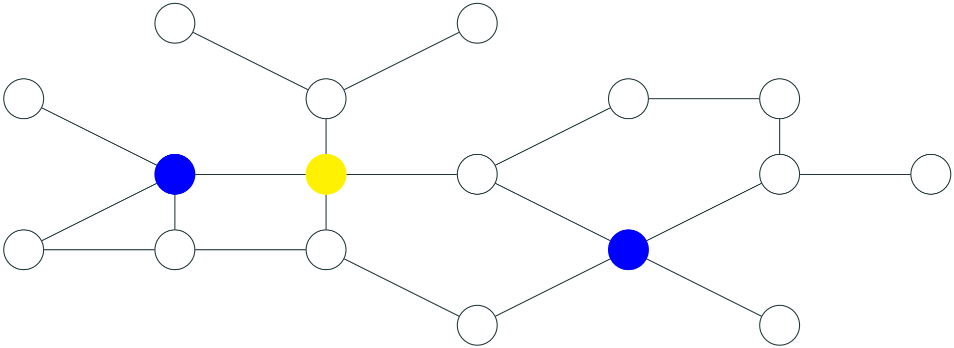
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



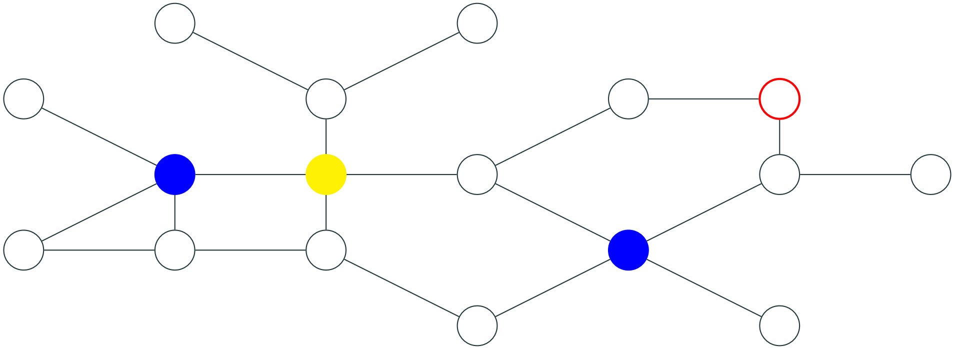
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



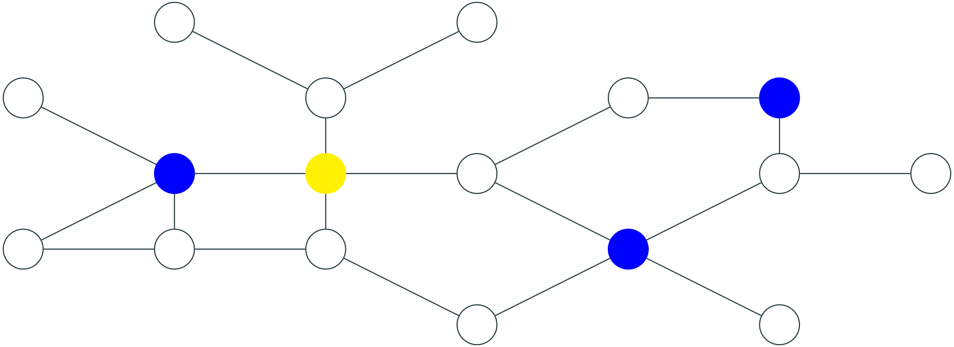
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



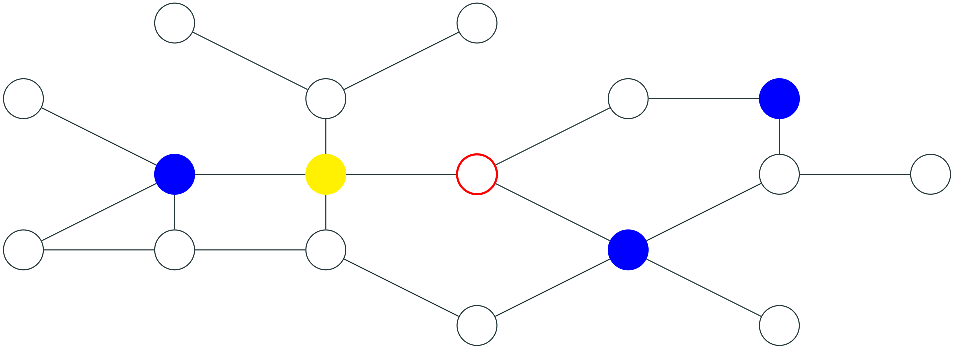
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



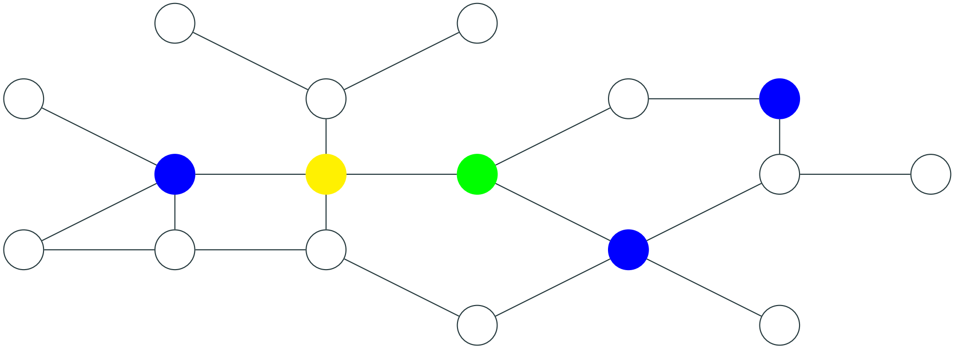
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity : maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$



The Complexities of Sinkless Orientation

Brandt *et. al* (2016), Mohsen Ghaffari, Hsin-Hao Su (2017)
Sinkless Orientation has complexity in the **deterministic** LOCAL model.

The Complexities of Sinkless Orientation

Brandt *et. al* (2016), Mohsen Ghaffari, Hsin-Hao Su (2017)
Sinkless Orientation has complexity $\Theta(\log n)$ in the **deterministic** LOCAL model.

The Complexities of Sinkless Orientation

Brandt *et. al* (2016), Mohsen Ghaffari, Hsin-Hao Su (2017)
Sinkless Orientation has complexity $\Theta(\log n)$ in the **deterministic** LOCAL model.

Idea of the algorithm :

- At distance at most $\log n$, we have :
  1. a node of degree ≤ 2
  2. a cycle of length $\leq 2 \log n$
- Case (1) : orient your edge towards the closest one
- Case (2) : pick the smallest one (in size, and then according to identifiers)
  Orient this cycle from smallest identifier to smaller neighbor
- Otherwise, orient your edge towards the closest cycle
- All nodes are now satisfied, orient other edges arbitrarily

The Complexities of Sinkless Orientation

Brandt *et. al* (2016), Mohsen Ghaffari, Hsin-Hao Su (2017)
Sinkless Orientation has complexity $\Theta(\log n)$ in the **deterministic** LOCAL model.

The Complexities of Sinkless Orientation

Brandt *et. al* **(2016), Mohsen Ghaffari, Hsin-Hao Su (2017)**
Sinkless Orientation has complexity $\Theta(\log n)$ in the **deterministic** LOCAL model.

Brandt *et. al* **(2016), Mohsen Ghaffari, Hsin-Hao Su (2017)**
Sinkless Orientation has complexity $\Theta(\log_\Delta \log n)$ in the **randomized** LOCAL model.

Balliu *et. al* **(2023)**
Sinkless Orientation has complexity $O(\log \log n)$ in the **deterministic** SLOCAL model.

Randomized Sinkless Orientation

General idea :

- Compute a random solution
- Remove outputs that do not work
- Shattering of the graph (i.e. small connected components of unsatisfied nodes)
- Run deteministic algorithm on small components

Randomized Sinkless Orientation

We assume a Δ-regular graph with $\Delta > 500$

1. Each edge, with probability $1/4$, decides to orient itself (or not)
2. Identify bad nodes :
   - Type I : More than $\Delta/2$ incident oriented edges
   - Type II : Not type I and neighbors of a Type I node
   - Type III : Not type I or II and no outgoing oriented edge
3. Unorient edges incident to Type I nodes

Randomized Sinkless Orientation

We assume a Δ-regular graph with $\Delta > 500$

1. Each edge, with probability $1/4$, decides to orient itself (or not)
2. Identify bad nodes :
   - Type I : More than $\Delta/2$ incident oriented edges
   - Type II : Not type I and neighbors of a Type I node
   - Type III : Not type I or II and no outgoing oriented edge
3. Unorient edges incident to Type I nodes

Observations :

- Good nodes have an outgoing edge
- (Admitted) WHP, each connected component of bad nodes have diameter $\operatorname{poly}\log n$

Randomized Sinkless Orientation

We assume a Δ-regular graph with $\Delta > 500$

1. Each edge, with probability $1/4$, decides to orient itself (or not)
2. Identify bad nodes :
   - Type I : More than $\Delta/2$ incident oriented edges
   - Type II : Not type I and neighbors of a Type I node
   - Type III : Not type I or II and no outgoing oriented edge
3. Unorient edges incident to Type I nodes
4. Use the deterministic algorithm on the bad nodes in $O(\log \log n)$

Observations :

- Good nodes have an outgoing edge
- (Admitted) WHP, each connected component of bad nodes have diameter $\operatorname{poly}\log n$

Composition in the SLOCAL model

Balliu *et. al* **(2023)**
Let $\mathcal{A}$ and $\mathcal{B}$ be SLOCAL algorithms with respective localities $T_{\mathcal{A}}$ and $T_{\mathcal{B}}$, and let $\mathcal{B}$ depends on the output of $\mathcal{A}$. Then there exists an algorithm $\mathcal{C}$ with locality $T_{\mathcal{A}} + 2T_{\mathcal{B}}$ that solves the same problem as $T_{\mathcal{B}}$ without dependency on the output of $T_{\mathcal{A}}$.

Composition in the SLOCAL model

Balliu *et. al* **(2023)**
Let $\mathcal{A}$ and $\mathcal{B}$ be SLOCAL algorithms with respective localities $T_{\mathcal{A}}$ and $T_{\mathcal{B}}$, and let $\mathcal{B}$ depends on the output of $\mathcal{A}$. Then there exists an algorithm $\mathcal{C}$ with locality $T_{\mathcal{A}} + 2T_{\mathcal{B}}$ that solves the same problem as $T_{\mathcal{B}}$ without dependency on the output of $T_{\mathcal{A}}$.

Proof :

- To compute $\mathcal{B}(v)$, we need output of $\mathcal{A}$ on $N_{T_{\mathcal{A}}}(v)$
- $\mathcal{C}$ stores the outputs of $\mathcal{A}$ pre-computed by each u for their neighborhood on u
- $\mathcal{C}$ needs to look at distance $T_{\mathcal{B}}$ of $N_{T_{\mathcal{B}}}(v)$ to check for pre-computed outputs

SLOCAL Sinkless Orientation of High Degree Nodes

Algorithm that processes edges sequentially :

1. u is satisfied if it has an outgoing edge or has degree $\leq \log n + 1$
2. For $uv \in E$:
   - If u or v is satisfied, orient edge to the satisfied node
   - Otherwise, orient towards the node with the fewest processed edges
     This edge gets marked
3. At the end, all nodes are satisfied

SLOCAL Sinkless Orientation of High Degree Nodes

Algorithm that processes edges sequentially :

1. u is satisfied if it has an outgoing edge or has degree $\leq \log n + 1$
2. For $uv \in E$:
   - If u or v is satisfied, orient edge to the satisfied node
   - Otherwise, orient towards the node with the fewest processed edges
     This edge gets marked
3. At the end, all nodes are satisfied

Observations :

- At each step, we consider the connected components using marked edges
- An unsatisfied node with b marked edges towards itself is in a component of size $\geq 2^b$

SLOCAL Sinkless Orientation of High Degree Nodes

2. For $uv \in E$:
    - Orient towards the node with the fewest processed edges
      This edge gets marked
3. At the end, all nodes are satisfied

Observations :

- At each step, we consider the connected components using marked edges
- An unsatisfied node with b marked edges towards itself is in a component of size $\geq 2^b$

Proof : By induction on b. For $b = 0$, it is true.

When edge $v \to u$ is added, indegree of u is $b + 1$ and indegree of v is $\geq b$

SLOCAL Sinkless Orientation of High Degree Nodes

2. For $uv \in E$:
   - Orient towards the node with the fewest processed edges
     This edge gets marked
3. At the end, all nodes are satisfied

Observations :

- At each step, we consider the connected components using marked edges
- An unsatisfied node with b marked edges towards itself is in a component of size $\geq 2^b$

Proof : By induction on b. For $b = 0$, it is true.
When edge $v \to u$ is added, indegree of u is $b + 1$ and indegree of v is $\geq b$
If u and v where in the same component, there is a cycle of marked edges

SLOCAL Sinkless Orientation of High Degree Nodes

2. For $uv \in E$:
   - Orient towards the node with the fewest processed edges
     This edge gets marked
3. At the end, all nodes are satisfied

Observations :

- At each step, we consider the connected components using marked edges
- An unsatisfied node with b marked edges towards itself is in a component of size $\geq 2^b$

Proof : By induction on b. For $b = 0$, it is true.
When edge $v \to u$ is added, indegree of u is $b + 1$ and indegree of v is $\geq b$
If u and v where in the same component, there is a cycle of marked edges
In that cycle, u has two ingoing edges
$\Rightarrow$ The cycle has a node with two ongoing edges
$\Rightarrow$ One of the edges of the cycle could not have been marked

SLOCAL Sinkless Orientation

Three steps algorithm :

1. Compute a MIS I of G^{2T+1} with $T = \log(\log n + 1)$
   - Partition into clusters : each node u selects closest element $v \in I$ and join C_v
   - uv inter-cluster edge if u and v in different clusters
   - uv intra-cluster edge if u and v in same cluster
   - G_C cluster graph, with $C_u C_v \in E_C$ if inter-cluster edge between C_u and C_v
   - One round in G_C is $O(T)$ rounds in G

SLOCAL Sinkless Orientation

Three steps algorithm :

1. Compute a MIS I of G^{2T+1} with $T = \log(\log n + 1)$
   - Partition into clusters : each node u selects closest element $v \in I$ and join C_v
   - uv inter-cluster edge if u and v in different clusters
   - uv intra-cluster edge if u and v in same cluster
   - G_C cluster graph, with $C_u C_v \in E_C$ if inter-cluster edge between C_u and C_v
   - One round in G_C is $O(T)$ rounds in G
2. Compute SLOCAL Sinkless Orientation of High Degree Nodes on G_C

SLOCAL Sinkless Orientation

Three steps algorithm :

1. Compute a MIS I of G^{2T+1} with $T = \log(\log n + 1)$
   - Partition into clusters : each node u selects closest element $v \in I$ and join C_v
   - uv inter-cluster edge if u and v in different clusters
   - uv intra-cluster edge if u and v in same cluster
   - G_C cluster graph, with $C_u C_v \in E_C$ if inter-cluster edge between C_u and C_v
   - One round in G_C is $O(T)$ rounds in G
2. Compute SLOCAL Sinkless Orientation of High Degree Nodes on G_C
   - High degree clusters (at least $\log n + 1$ neighbors) has an outgoing edge
   - Low degree clusters have a node of degree ≤ 2 or a cycle

SLOCAL Sinkless Orientation

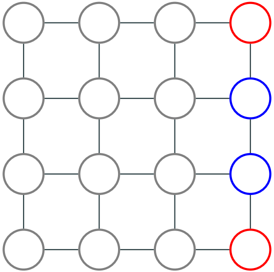
Three steps algorithm :

1. Compute a MIS I of G^{2T+1} with $T = \log(\log n + 1)$
   - Partition into clusters : each node u selects closest element $v \in I$ and join C_v
   - uv inter-cluster edge if u and v in different clusters
   - uv intra-cluster edge if u and v in same cluster
   - G_C cluster graph, with $C_u C_v \in E_C$ if inter-cluster edge between C_u and C_v
   - One round in G_C is $O(T)$ rounds in G
2. Compute SLOCAL Sinkless Orientation of High Degree Nodes on G_C
   - High degree clusters (at least $\log n + 1$ neighbors) has an outgoing edge
   - Low degree clusters have a node of degree ≤ 2 or a cycle
3. Depending on the cluster :
   - High degree clusters compute a spanning tree toward a non-sink node of the cluster.
   - Low degree clusters orient toward low degree node or cycle

Shared Randomness Separation

Carrying input problem :

- Nodes form an oriented grid of size $\sqrt{n} \times \sqrt{n}$
- Nodes on the right have input 0 or 1
- Each node must copy the output on its right
- At least one of the right nodes must have input=output



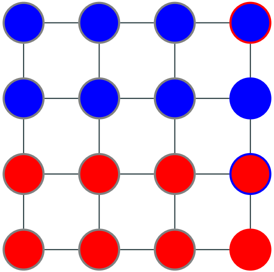
The complexities :

- Deterministic :
- Randomized :
- Shared Randomness :

Shared Randomness Separation

Carrying input problem :

- Nodes form an oriented grid of size $\sqrt{n} \times \sqrt{n}$
- Nodes on the right have input 0 or 1
- Each node must copy the output on its right
- At least one of the right nodes must have input=output



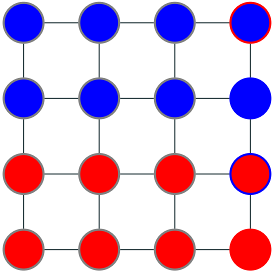
The complexities :

- Deterministic :
- Randomized :
- Shared Randomness :

Shared Randomness Separation

Carrying input problem :

- Nodes form an oriented grid of size $\sqrt{n} \times \sqrt{n}$
- Nodes on the right have input 0 or 1
- Each node must copy the output on its right
- At least one of the right nodes must have input=output



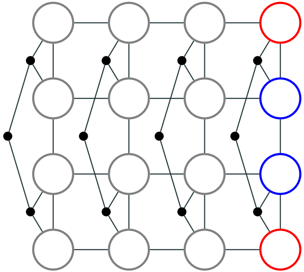
The complexities :

- Deterministic : $\Theta(\sqrt{n})$
- Randomized : $\Theta(\sqrt{n})$
- Shared Randomness : $\Theta(\sqrt{n})$

Shared Randomness Separation

Carrying input problem :

- Nodes form an oriented grid of size $\sqrt{n} \times \sqrt{n}$
- Nodes on the right have input 0 or 1
- Each node must copy the output on its right
- At least one of the right nodes must have input=output
- Binary tree on each column



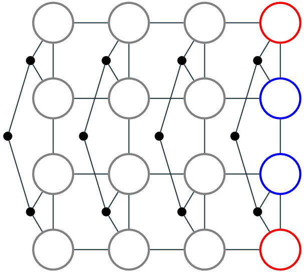
The complexities :

- Deterministic : $\Theta(\sqrt{n})$
- Randomized : $\Theta(\sqrt{n})$
- Shared Randomness : $\Theta(\sqrt{n})$

Shared Randomness Separation

Carrying input problem :

- Nodes form an oriented grid of size $\sqrt{n} \times \sqrt{n}$
- Nodes on the right have input 0 or 1
- Each node must copy the output on its right
- At least one of the right nodes must have input=output
- Binary tree on each column



The complexities :

- Deterministic : $\Theta(\sqrt{n})$
- Randomized : $\Theta(\sqrt{n})$
- Shared Randomness : $\Theta(\log n)$

Bibliography

- Balliu, Hirvonen, Melnyk, Olivetti, Rybicki, Suomela. **Local Mending**. In SIROCCO 22.
- Balliu, Ghaffari, Kuhn, Modanese, Olivetti, Rabie, Suomela, Uitto. **Shared Randomness Helps with Local Distributed Problems**. On arXiv.
- Balliu, Korhonen, Kuhn, Lievonen, Olivetti, Pai, Paz, Rybicki, Schmid, Studený, Suomela, Uitto. **Sinkless Orientation Made Simple**. In SOSA 23.
- Brandt, Fischer, Hirvonen, Keller, Lempiäinen, Rybicki, Suomela, Uitto. **A lower bound for the distributed Lovász local lemma**. In STOC 16.
- Guy Even, Moti Medina, Dana Ron. **Best of two local models : Centralized local and distributed local algorithms** In Inf. Comput. 262, 2018.
- Ghaffari, Kuhn, Maus. **On the complexity of local distributed graph problems**. In STOC 17.
- Ghaffari, Su. **Distributed Degree Splitting, Edge Coloring, and Orientations**. In SODA 17.
- Rosenbaum, Suomela. **Seeing far vs. seeing wide** In PODC 20.