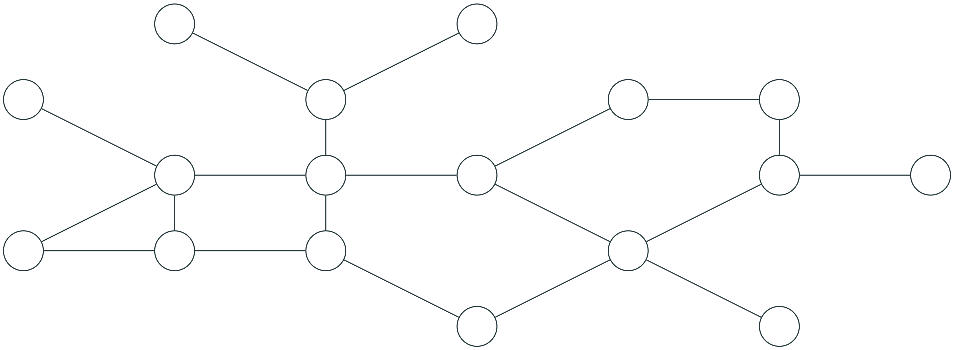
Distributed Computing
12 - Sleeping Model

Mikaël Rabie
Université Paris Cité, IRIF

The Awaken Complexity

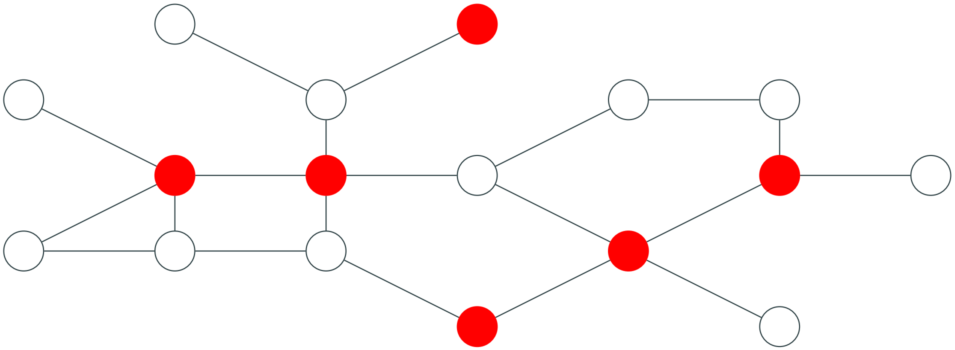
Distributed Sleeping Model

- LOCAL model
- At each round, a node decides if it is active or not
- A node communicates only with its active neighbors
- Complexity : maximal number of awaken rounds for a single node



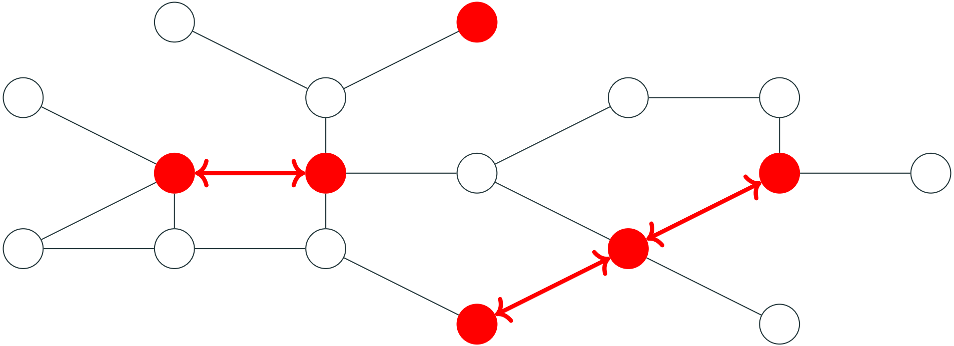
Distributed Sleeping Model

- LOCAL model
- At each round, a node decides if it is active or not
- A node communicates only with its active neighbors
- Complexity : maximal number of awaken rounds for a single node

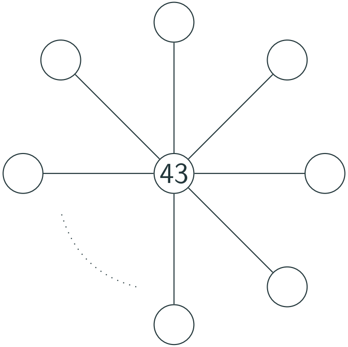


Distributed Sleeping Model

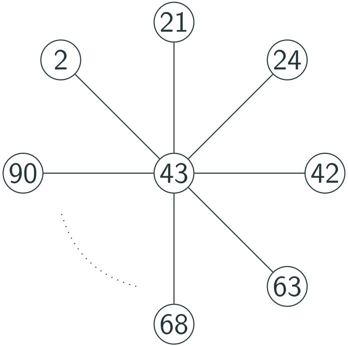
- LOCAL model
- At each round, a node decides if it is active or not
- A node communicates only with its active neighbors
- Complexity : maximal number of awaken rounds for a single node



$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds

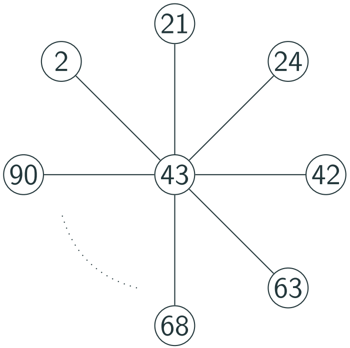


$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



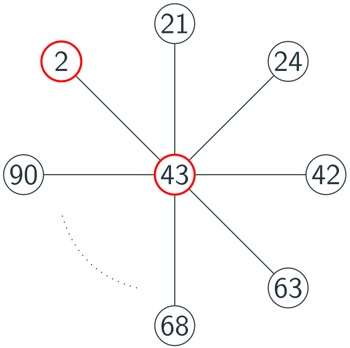
- Round 0 : Learn the identifiers of my neighbors

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



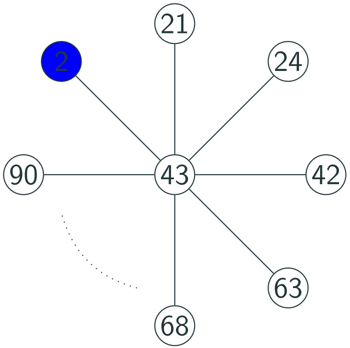
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



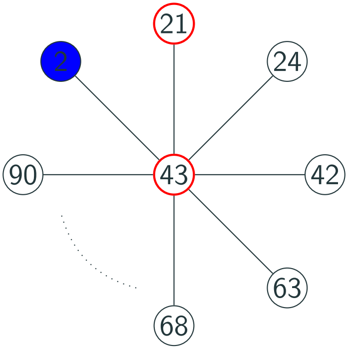
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 2 : Node 2 chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



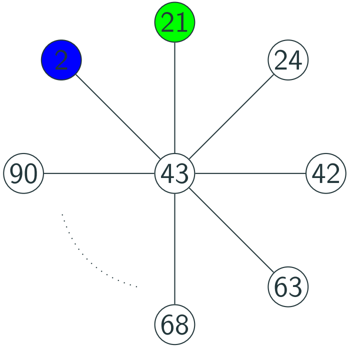
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 2 : Node 2 chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



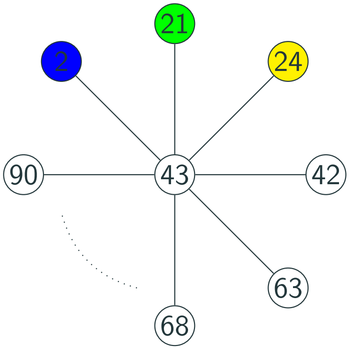
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 21 : Node 21 chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



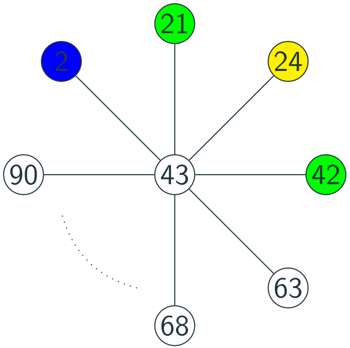
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 21 : Node 21 chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



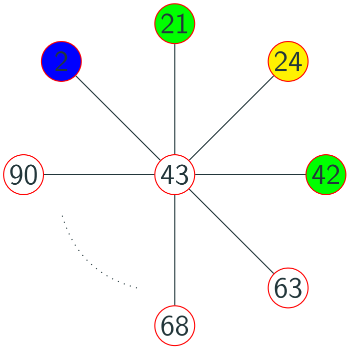
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 24 : Node 24 chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



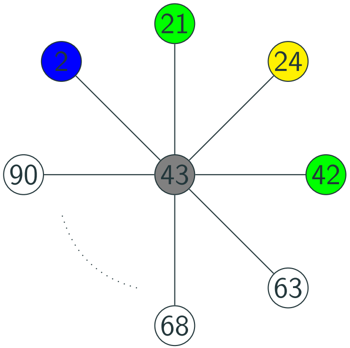
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 42 : Node 42 chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



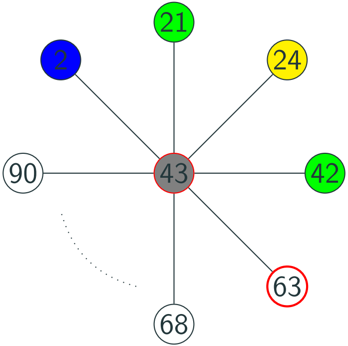
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 43 : u learns colors of nodes $2, 21, 24, 42$ and chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



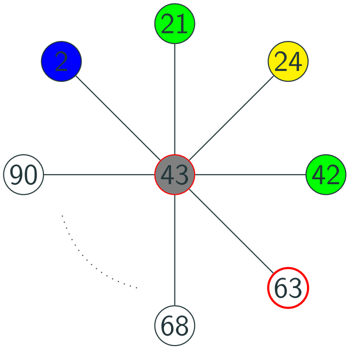
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 43 : u learns colors of nodes $2, 21, 24, 42$ and chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 63 : Node 63 learns color of u and chooses its color

$\Delta + 1$-Coloring in $O(\Delta)$ awaken rounds



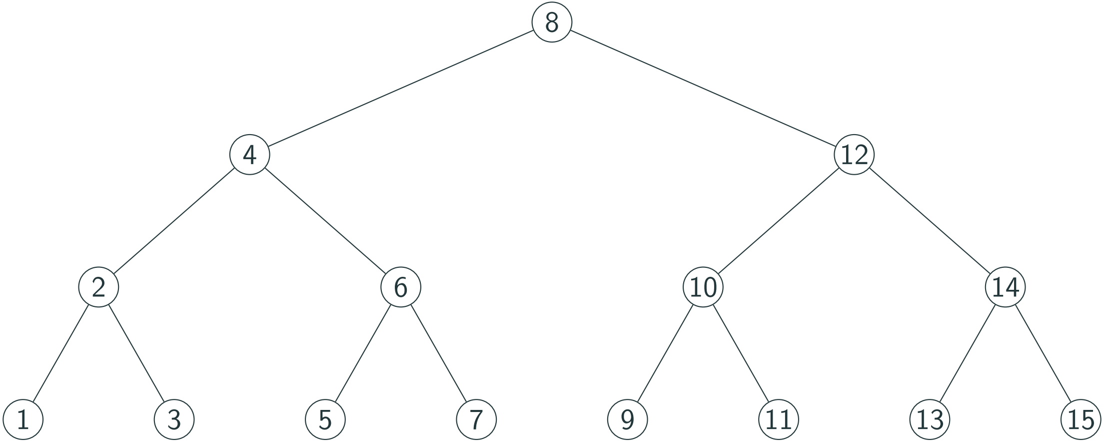
- Round 0 : Learn the identifiers of my neighbors
- For each $i \in N(u)_{\leq 1}$, round i : Wake up
- Round 63 : Node 63 learns color of u and chooses its color

Drawback : The round complexity is $O(M)$, M being the maximal identifier.

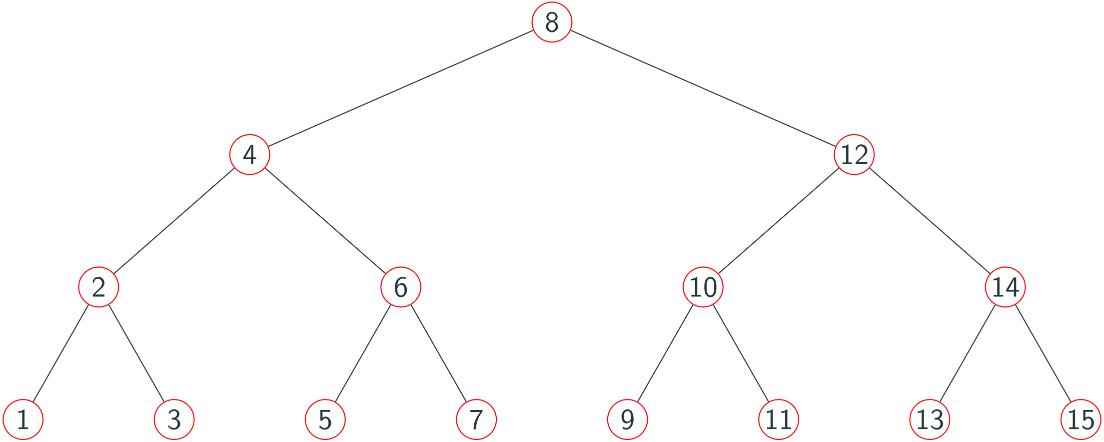
Reduce K colors in $\log K$ awake rounds

Barenboim and Maimon (2021)
Given a K-coloring of the graph, we can compute a $(\Delta + 1)$-coloring in $O(\log K)$ awaken rounds and $O(K)$ rounds in the Sleeping LOCAL model.

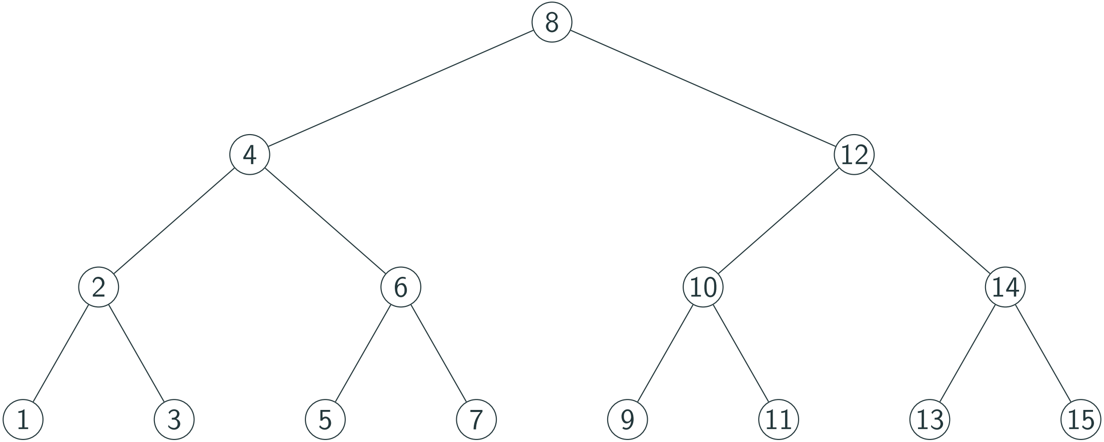
Reduce K colors in $\log K$ awake rounds



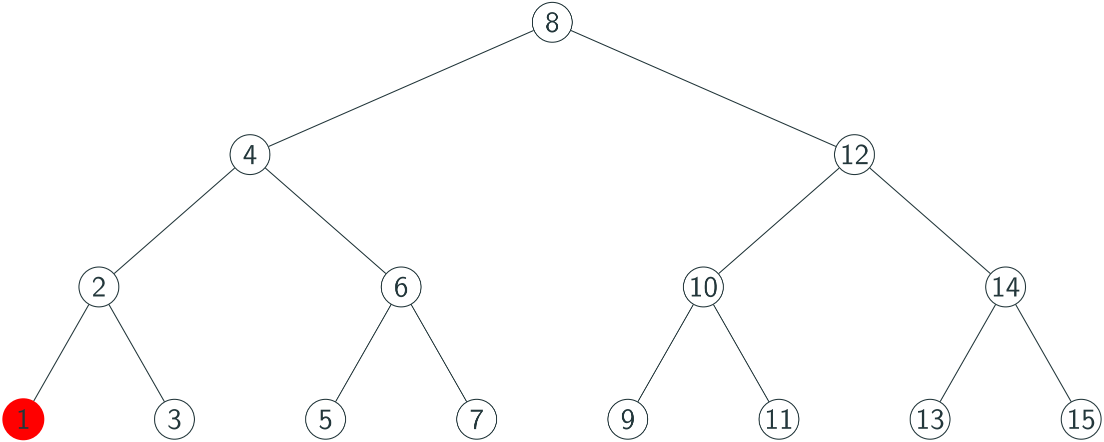
Reduce K colors in $\log K$ awake rounds



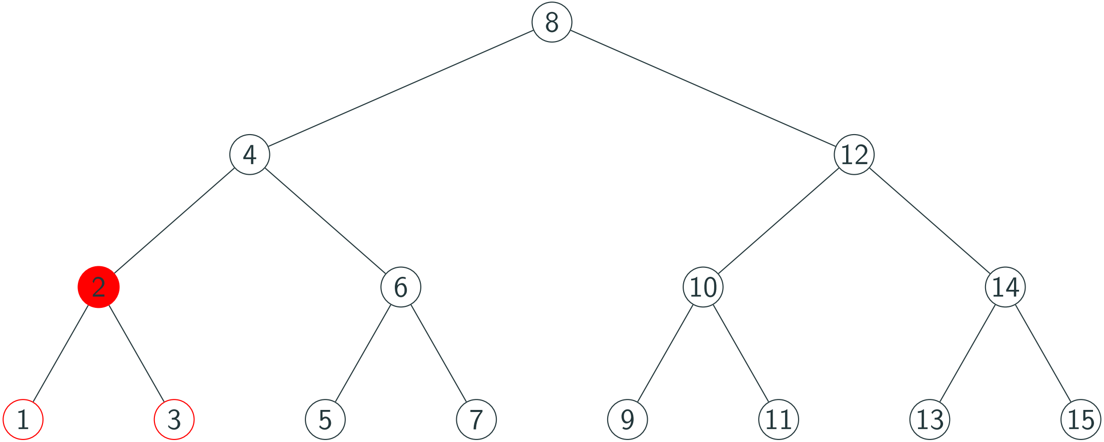
Reduce K colors in $\log K$ awake rounds



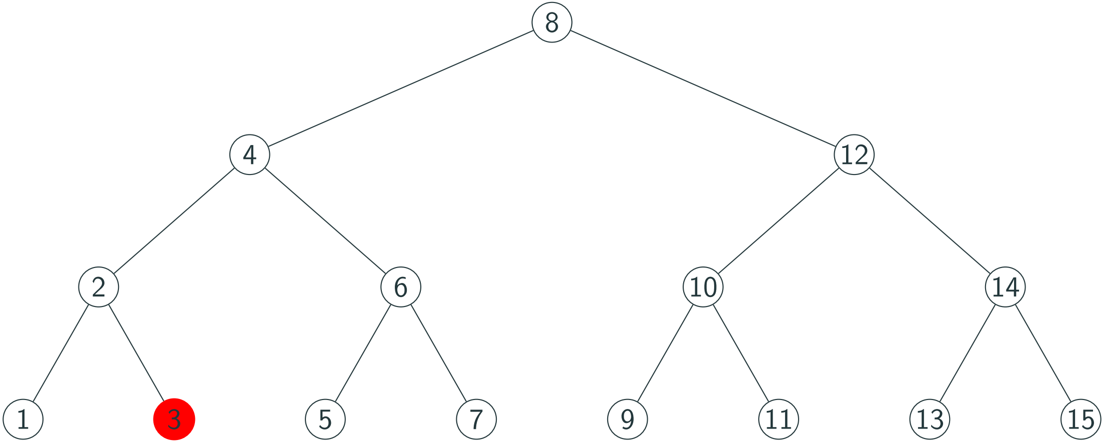
Reduce K colors in $\log K$ awake rounds



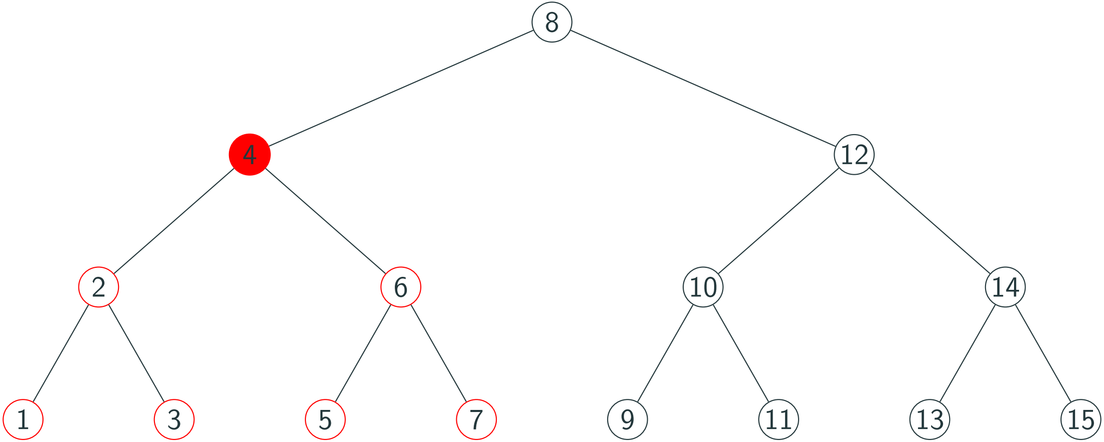
Reduce K colors in $\log K$ awake rounds



Reduce K colors in $\log K$ awake rounds



Reduce K colors in $\log K$ awake rounds



The $\log n$ Complexity

Full Knowledge of the Graph

Barenboim and Maimon (2021)
Any graph problem can be solved in $O(\log n)$ awaken rounds in the Sleeping LOCAL model.

This algorithm takes $O(\operatorname{poly} M)$ rounds.

Full Knowledge of the Graph

Barenboim and Maimon (2021)
Any graph problem can be solved in $O(\log n)$ awaken rounds in the Sleeping LOCAL model.

This algorithm takes $O(\text{poly}\, M)$ rounds.

Distributed Layered Tree (DLT) - Oriented Spanning Tree such as :

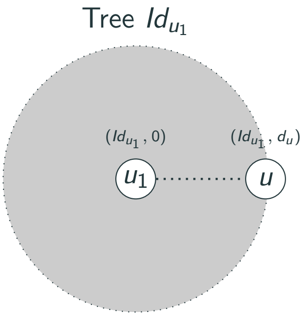
- Each vertex has a label
- The label of a vertex is bigger than its parent's
- Each vertex knows the label of its neighbours in the tree

Constant Coordination
Broadcast and Convergecast can be done in $O(1)$ rounds in a DLT.

Building a DLT

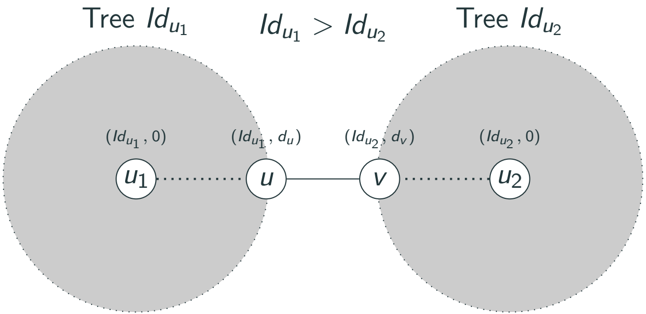
Barenboim and Maimon (2021)
A DLT can be built in $O(\log n)$ awaken rounds in the Sleeping LOCAL model.

Building a DLT



- Labels are of the form (a, b), ordered lexicographically.
- At the beginning, all nodes have label $(Id(u), 0)$.
- At the beginning of each expand step, all nodes of a subtree T are of the form $(L(T), b)$.

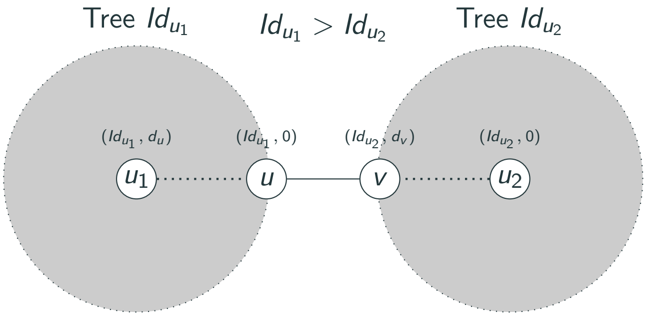
Building a DLT



- Repeat $\log n$ times :

1. Select a neighbour Tree T' with smaller label ($Id_{u_1} > Id_{u2}$).

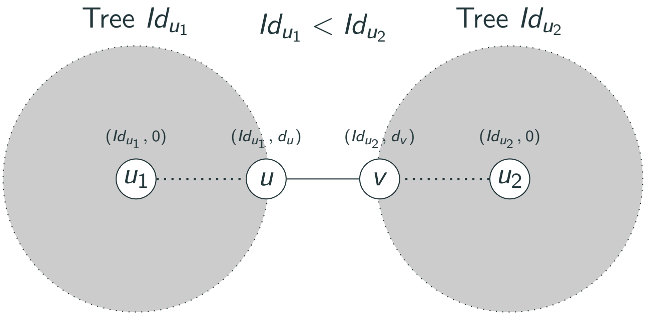
Building a DLT



- Repeat $\log n$ times :

2. Merge T and T', using an edge (u, v).

Building a DLT



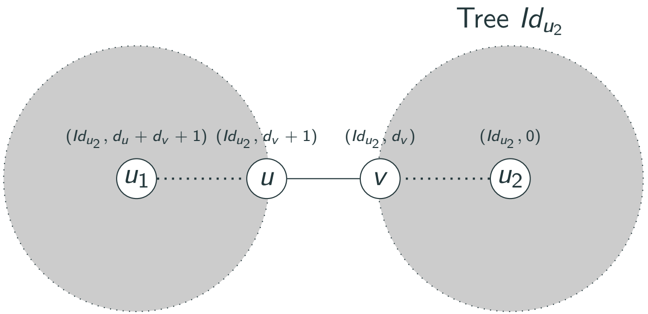
- Repeat $\log n$ times :

3. If T could not choose a neighbour and was not selected

   T chooses a tree T' to join using an edge (u, v).

   This forms a star of trees around $T' \Rightarrow O(1)$ merge rounds.

Building a DLT



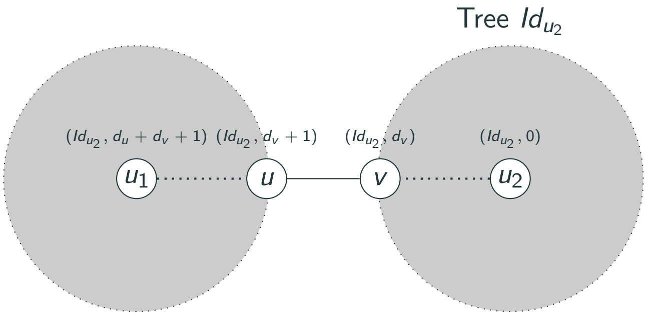
- Repeat $\log n$ times :

3. If T could not choose a neighbour and was not selected

   T chooses a tree T' to join using an edge (u, v).

   This forms a star of trees around $T' \Rightarrow O(1)$ merge rounds.

Building a DLT



- Repeat $\log n$ times :

4. All nodes learn their new neighbours in the tree.
5. Convergecast to gather the new structure of the component C to the root r.
6. Broadcast a new labelling $(L(r), dist(r))$.

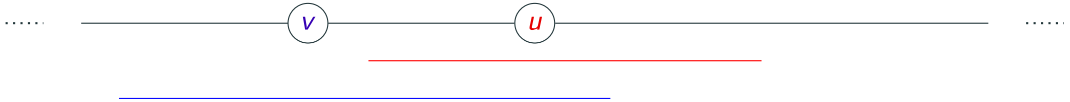
Sleeping Lower Bound

Augustine ***et. al*** **(2022)**

Any algorithm to solve 2-coloring with probability exceeding $1/8$ on a ring network requires $\Omega(\log n)$ awake time.

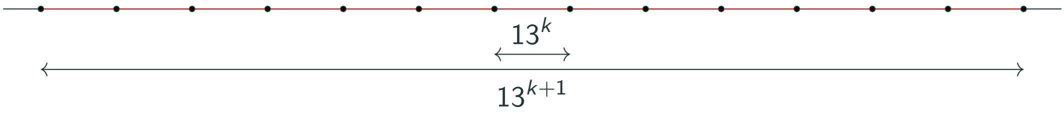
Sleeping Lower Bound

Augustine *et. al* **(2022)**
Any algorithm to solve 2-coloring with probability exceeding $1/8$ on a ring network requires $\Omega(\log n)$ awake time.



- After k rounds, a node knows about some segment that includes itself
- No node v on the left of u in the path can know more than u on its right

Sleeping Lower Bound



By induction : For any k, for any segment I of 13^k nodes, there exists, with probability $\mathcal{P} > 1/2$, a node $u \in I$ who knows less than I after k rounds.

Sleeping Lower Bound

By induction : For any k, for any segment I of 13^k nodes, there exists, with probability $\mathcal{P} > 1/2$, a node $u \in I$ who knows less than I after k rounds.

- Probability that it is true on 5 of the 13 subsegments is at least $5/6$
- Probability that B, C or D wakes up before A and E is at least $1/2$

$(\Delta + 1)$-Coloring

Trade-Off

Find the possible trade-off between awaken and usual rounds to resolve a problem.

$(\Delta + 1)$-coloring of paths :

Awaken rounds	Rounds

Trade-Off

Find the possible trade-off between awaken and usual rounds to resolve a problem.

$(\Delta + 1)$-coloring of paths :

Awaken rounds	Rounds
3	$O(M)$

Trade-Off

Find the possible trade-off between awaken and usual rounds to resolve a problem.

$(\Delta + 1)$-coloring of paths :

Awaken rounds	Rounds
3	$O(M)$
$O(\log^* n)$	$O(\log^* n)$

Trade-Off

Find the possible trade-off between awaken and usual rounds to resolve a problem.

$(\Delta + 1)$-coloring of paths :

Awaken rounds	Rounds
3	$O(M)$
$O(\log^* n)$	$O(\log^* n)$
$3 + k$	$O(\log^{(2k)} M)$

Trade-Off

Find the possible trade-off between awaken and usual rounds to resolve a problem.

$(\Delta + 1)$-coloring :

Awaken rounds	Rounds
$O(\Delta)$	$O(M)$
$O(\log M)$	$O(M)$
$O(\log^* n + \log \Delta)$	$O(\log^* n + \text{poly}\, \Delta)$

Trade-Off

Find the possible trade-off between awaken and usual rounds to resolve a problem.

$(\Delta + 1)$-coloring :

Awaken rounds	Rounds
$O(\Delta)$	$O(M)$
$O(\log M)$	$O(M)$
$O(\log^* n + \log \Delta)$	$O(\log^* n + \text{poly}\, \Delta)$

Linial (1992)
There exists an algorithm that solves $O(\Delta^2)$-coloring with round-complexity $O(\log^* n)$.

Trade-Off

Find the possible trade-off between awaken and usual rounds to resolve a problem.

$(\Delta + 1)$-coloring :

Awaken rounds	Rounds
$O(\Delta)$	$O(M)$
$O(\log M)$	$O(M)$
$O(\log^* n + \log \Delta)$	$O(\log^* n + \text{poly}\, \Delta)$

Balliu, Fraigniaud, Olivetti, R.

There exists an algorithm that solves $(\Delta + 1)$-coloring with $O(\sqrt{\log n} \cdot \log^* n)$ awake-complexity and round-complexity $\text{poly}(M)$.

Label Clustering

Uniquely-labeled BFS-clustering :

- Two functions (ℓ, δ) assigning a pair $(\ell(v), \delta(v)) \in \mathbb{N} \times \mathbb{N}$ to each node $v \in V$
- $\forall i > 0$, $V_i = \{v \in V \mid \ell(v) = i\}$ and $G_i = G_{V_i}$. If G_i is non-empty :
  - G_i is connected
  - There is a unique node u of G_i with $\delta(u) = 0$
  - $\forall v \in V_i$, $\delta(v)$ is the distance from u to v in G_i.
- Each G_i is a cluster.

Virtual Graph $H = (V_H, E_H)$:

- $V_H = \{\text{clusters } C \text{ induced by labels}\}$
- $CC' \in E_H \Leftrightarrow \exists u \in C$ and $v \in C'$ such that $uv \in E$.

Label Clustering

Colored BFS-clustering :

- Two functions (γ, δ) assigning a pair $(\gamma(v), \delta(v)) \in \mathbb{N} \times \mathbb{N}$ to each node $v \in V$
- $\forall i > 0$, $V_i = \{v \in V \mid \gamma(v) = i\}$ and $G_i = G_{V_i}$. If G_i is non-empty :
  - For any connected component C_i of G_i
  - There is a unique node u of C_i with $\delta(u) = 0$
  - $\forall v \in C_i$, $\delta(v)$ is the distance from u to v in C_i.
- Each C_i is a cluster.

Virtual Graph $H = (V_H, E_H)$:

- $V_H = \{\text{clusters } C \text{ induced by labels}\}$
- $CC' \in E_H \Leftrightarrow \exists u \in C$ and $v \in C'$ such that $uv \in E$.

From Clusters to $(\Delta + 1)$-coloring

$(\Delta + 1)$-coloring

- (γ, δ) : a colored BFS-clustering of G
- Assume each node v of G knows $\gamma(v)$ and $\delta(v)$
- $c = \max_{v \in V} \gamma(v)$

$(\Delta + 1)$-coloring can be solved by a distributed algorithm with awake complexity $O(\log c)$, and round complexity $O(c \cdot n)$.

From Clusters to $(\Delta + 1)$-coloring

$(\Delta + 1)$-coloring

- (γ, δ) : a colored BFS-clustering of G
- Assume each node v of G knows $\gamma(v)$ and $\delta(v)$
- $c = \max_{v \in V} \gamma(v)$

$(\Delta + 1)$-coloring can be solved by a distributed algorithm with awake complexity $O(\log c)$, and round complexity $O(c \cdot n)$.

Idea :

- For each $i \leq c$:
  - If cluster has color in the Binary subtree of i :
    - Wake up, gather colors from awaken neighboring clusters
    - If cluster color is i, compute and broadcast your output colors
    - Gather output colors from neighbor clusters of color i

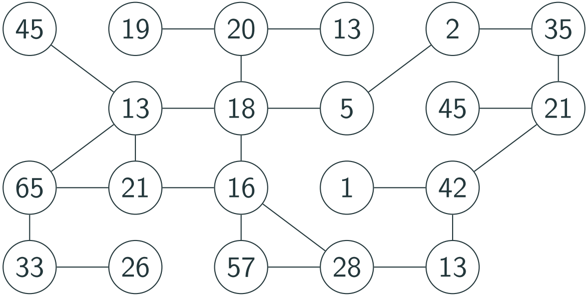
Clustering a Graph

One-round Cluster Reduction

$\exists a > 0 : \forall b > 0, \exists$ algorithm $\mathcal{A}$, with awake complexity $O(\log^* n)$ and round complexity $O(n^4)$ which computes a colored BFS-clustering (γ, δ) such that :

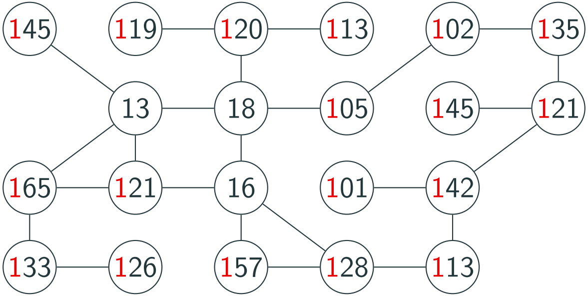
- (γ, δ) restricted to $\{v \in V \mid \gamma(v) \in \{1, \ldots, a \cdot b^2\}\}$ is a colored BFS-clustering
- $\forall v \in V$ with $\gamma(v) \in \{1, \ldots, a \cdot b^2\}$, $\delta(v) = 0$ (i.e., v is alone in its cluster) ;
- (γ, δ) restricted to $\{v \in V \mid \gamma(v) > a \cdot b^2\}$ is a uniquely-labeled BFS-clustering with at most n/b clusters.

Clustering a Graph



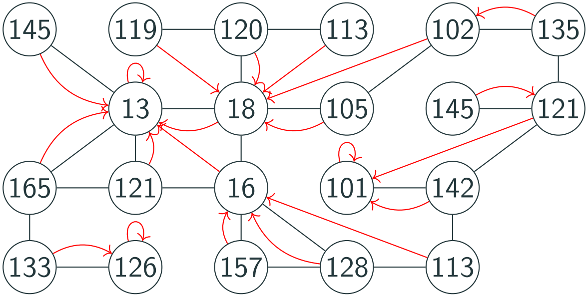
1. Compute a distance-2 k-coloring in $O(\log^* n)$ rounds ($k \in O(n^4)$)

Clustering a Graph

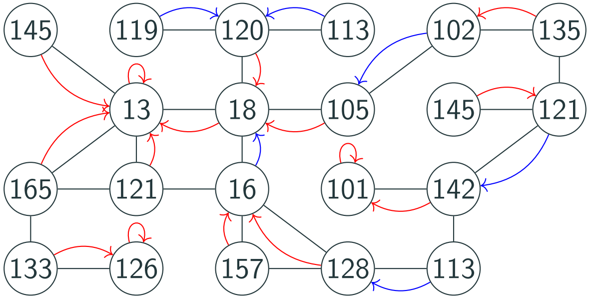


1. Compute a distance-2 k-coloring in $O(\log^* n)$ rounds ($k \in O(n^4)$)
2. Add k to the colors of nodes of degree $\leq b$

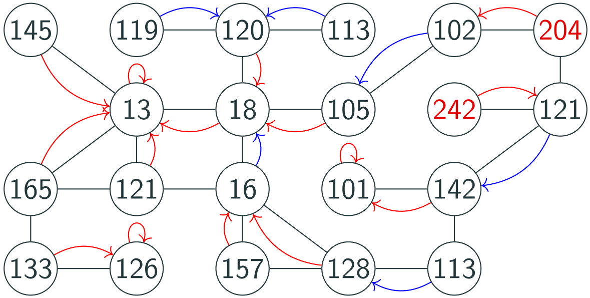
Clustering a Graph



1. Compute a distance-2 k-coloring in $O(\log^* n)$ rounds ($k \in O(n^4)$)
2. Add k to the colors of nodes of degree $\leq b$
3. Choose as parent :
   - yourself if your color is the smallest at distance 2
   - your smaller neighbor if one of them has smaller color than yours
   - your distance-2 smaller neighbor otherwise

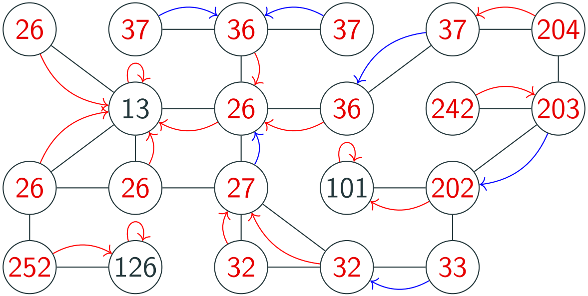


3. Choose as parent :
   - yourself if your color is the smallest at distance 2
   - your smaller neighbor if one of them has smaller color than yours
   - your distance-2 smaller neighbor otherwise
4. In the third case, choose a neighbor with same parent (blue case)



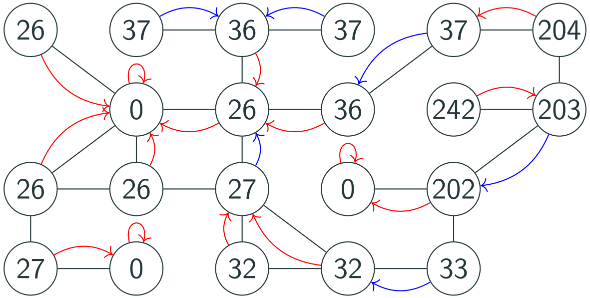
5. Compute a new color :
   - 2 times the color of your parent if you are not your parent
   - Add 1 if you are in the blue case
   - 0 if you are your own parent

Clustering a Graph



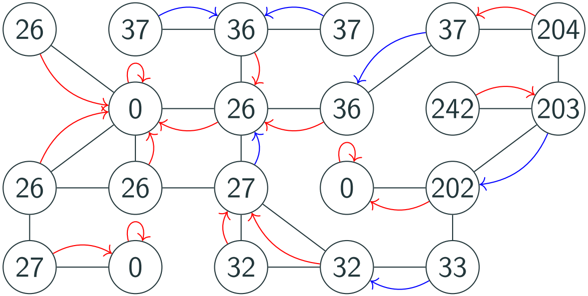
5. Compute a new color :
   - 2 times the color of your parent if you are not your parent
   - Add 1 if you are in the blue case
   - 0 if you are your own parent

Clustering a Graph



5. Compute a new color :
   - 2 times the color of your parent if you are not your parent
   - Add 1 if you are in the blue case
   - 0 if you are your own parent

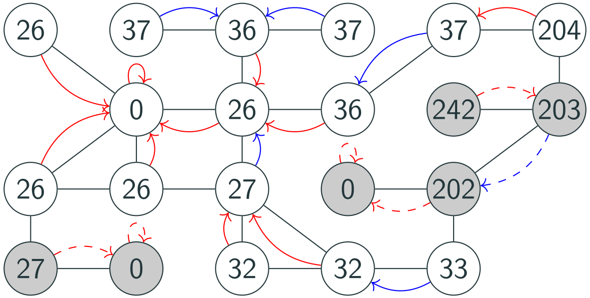
Clustering a Graph



5. Compute a new color :
   - 2 times the color of your parent if you are not your parent
   - Add 1 if you are in the blue case
   - 0 if you are your own parent

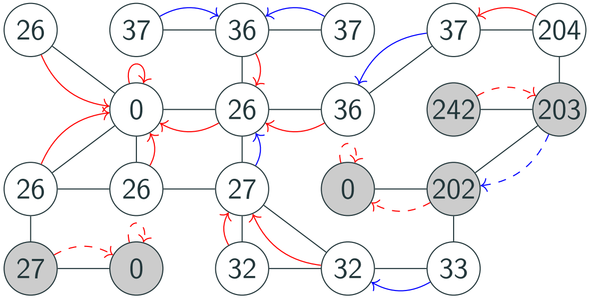
   You have built DLTs

Clustering a Graph



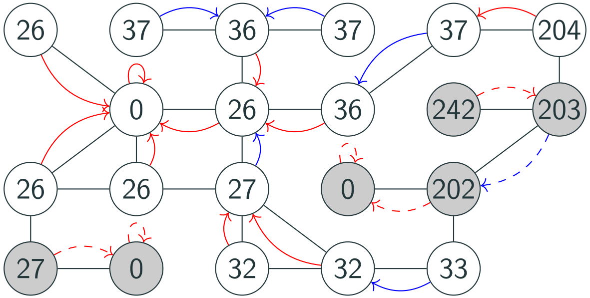
5. You have built DLTs
6. Perform a convergecast and broadcast
   Nodes compute their distance to their root
7. Keep only DLTs with roots of degree $> b$

Clustering a Graph



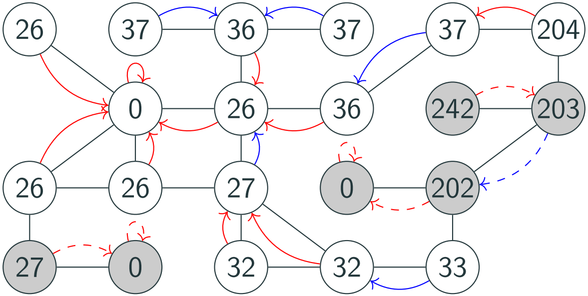
7. Keep only DLTs with roots of degree $> b$
8. U is the set of nodes not in a DLT :
   - They compute a $O(b^2)$-coloring of G_U in $O(\log^* n)$ rounds
   - They have colors from 1 to $a \cdot b^2$ (a is the constant of Linial's algorithm)
   - Nodes of U form clusters of size 1 with this new color

Clustering a Graph



8. U is the set of nodes not in a DLT :
   - They compute a $O(b^2)$-coloring of G_U in $O(\log^* n)$ rounds
   - They have colors from 1 to $a \cdot b^2$ (a is the constant of Linial's algorithm)
   - Nodes of U form clusters of size 1 with this new color
9. Other nodes set $\gamma(u) = \ell(v) + a \cdot b^2$ and $\delta(u) = dist(u, v)$, v being the root of u

Clustering a Graph



Observations : Subgraph induced by $\{v \in V \mid \gamma(v) > a \cdot b^2\}$ has at most n/b clusters

- Each root has more than b neighbors, all in its cluster (roots are distance-2 minimas)
- For each root, we can charge b nodes to itself
- Each node is charged at most once
- $\Rightarrow$ We have at most n/b roots of degree $> b$

Cluster Graph Simulation

LOCAL Simulation on Clusters

- H : the virtual graph induced by some uniquely-labeled BFS-clustering (ℓ, δ) of G.
- $\mathcal{A}$: a distributed algorithm running on H with α awake rounds and round complexity ϱ.
- $\forall u \in V_H$, input(u) is the input of u in H, and output(u) the output of $\mathcal{A}$ at vertex u.
- Assume each node v of G knows input$(\ell(v))$

It is possible to run $\mathcal{A}$ on G such that every node v of G computes output$(\ell(v))$ with awake complexity at most $7 \cdot \alpha$ and round complexity $O(\varrho \cdot n)$.

Cluster Graph Simulation

LOCAL Simulation on Clusters

- H : the virtual graph induced by some uniquely-labeled BFS-clustering (ℓ, δ) of G.
- $\mathcal{A}$: a distributed algorithm running on H with α awake rounds and round complexity ϱ.
- $\forall u \in V_H$, input(u) is the input of u in H, and output(u) the output of $\mathcal{A}$ at vertex u.
- Assume each node v of G knows input$(\ell(v))$

It is possible to run $\mathcal{A}$ on G such that every node v of G computes output$(\ell(v))$ with awake complexity at most $7 \cdot \alpha$ and round complexity $O(\varrho \cdot n)$.

Idea :

- The root of each cluster (i.e. with $\delta = 0$) simulates $\mathcal{A}$
- Each time a node awakes in H, the cluster performs some broadcasts and convergecasts
- In each round, 7 awaken activations are needed, and $\forall u \in V, \delta(u) \leq n$

Building Incrementally the Colored Clustering

Input : $G = (V, E)$

1. $H_0 : \ell_0(v) = \mathrm{ID}(v)$ and $\delta_0(v) = 0$
2. While $H_{i-1} \neq \emptyset$

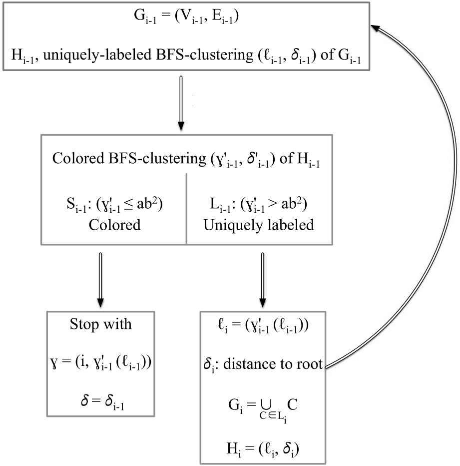
   Use clustering algorithm on H_{i-1}

   to compute H_i

Input : $G = (V, E)$

1. $H_0 : \ell_0(v) = \mathrm{ID}(v)$ and $\delta_0(v) = 0$
2. While $H_{i-1} \neq \emptyset$

   Use clustering algorithm on H_{i-1} to compute H_i



Building Incrementally the Colored Clustering

Input : $G = (V, E)$

1. $H_0 : \ell_0(v) = \mathrm{ID}(v)$ and $\delta_0(v) = 0$
2. While $H_{i-1} \neq \emptyset$

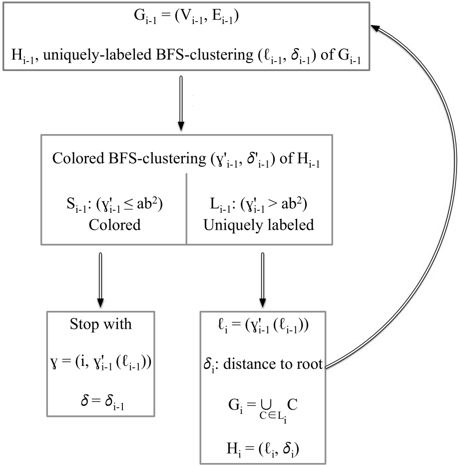
   Use clustering algorithm on H_{i-1}

   to compute H_i

After each iteration, we go from N to (up to) N/b vertices

$\Rightarrow$ k such that $b^k > n$ is enough

$\Rightarrow$ Clusters use kab^2 colors



Putting Things Together

We have :

- Two parameters b and k with $b^k \geq n$
- We compute a (γ, δ) colored BFS-clustering in $O(k \log^* n)$ rounds
- γ has maximal value kab^2 (a is a constant)
- From the BFS-clustering, we get a $(\Delta + 1)$-coloring in $O(\log(kb^2))$

Question : What is the optimal choice for k and b ?

Putting Things Together

We have :

- Two parameters b and k with $b^k \geq n$
- We compute a (γ, δ) colored BFS-clustering in $O(k \log^* n)$ rounds
- γ has maximal value kab^2 (a is a constant)
- From the BFS-clustering, we get a $(\Delta + 1)$-coloring in $O(\log(kb^2))$

Question : What is the optimal choice for k and b ?

- $b^k \geq n \Rightarrow k \geq \dfrac{\log n}{\log b}$
- We need $k \approx \log(kab^2) \approx \log b$

Putting Things Together

We have :

- Two parameters b and k with $b^k \geq n$
- We compute a (γ, δ) colored BFS-clustering in $O(k \log^* n)$ rounds
- γ has maximal value kab^2 (a is a constant)
- From the BFS-clustering, we get a $(\Delta + 1)$-coloring in $O(\log(kb^2))$

Question : What is the optimal choice for k and b ?

- $b^k \geq n \Rightarrow k \geq \dfrac{\log n}{\log b}$
- We need $k \approx \log(kab^2) \approx \log b$
- **Conclusion** : $k = 2\sqrt{\log n}$ and $b = 2^{\sqrt{\log n}}$ work

We get a $(Delta + 1)$-coloring algorithm with $O(\sqrt{\log n} \log^* n)$ awake complexity

Bibliography

- John Augustine, William K. Moses Jr., Gopal Pandurangan. **Awake complexity of distributed minimum spanning tree**. In SIROCCO 2024.
- Alkida Balliu, Pierre Fraigniaud, Dennis Olivetti, Mikaël Rabie. **Solving Sequential Greedy Problems Distributedly with Sub-Logarithmic Energy Cost**. In arXiv, 2024.
- Leonid Barenboim, Tzalik Maimon. **Deterministic Logarithmic Completeness in the Distributed Sleeping Model**. In DISC 2021.
- Nathan Linial. **Locality in distributed graph algorithms**. In SIAM J. Comput., 1992.