Making Self-Stabilizing Algorithms for any Locally Quasi-Greedy Problem

Johanne Cohen, Laurence Pilard, Mikaël Rabie, **Jonas Sénizergues**

6 mars 2025
Séminaire

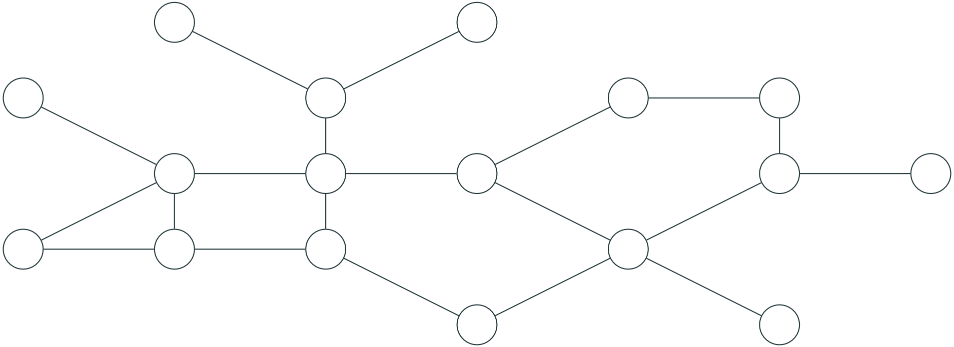
Introduction

Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

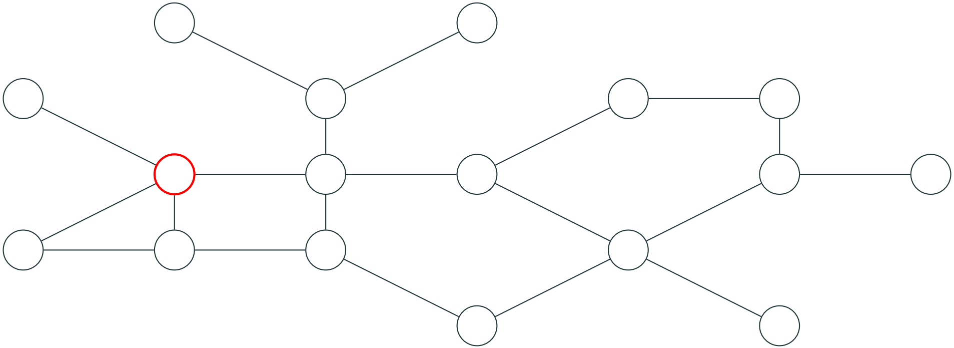


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

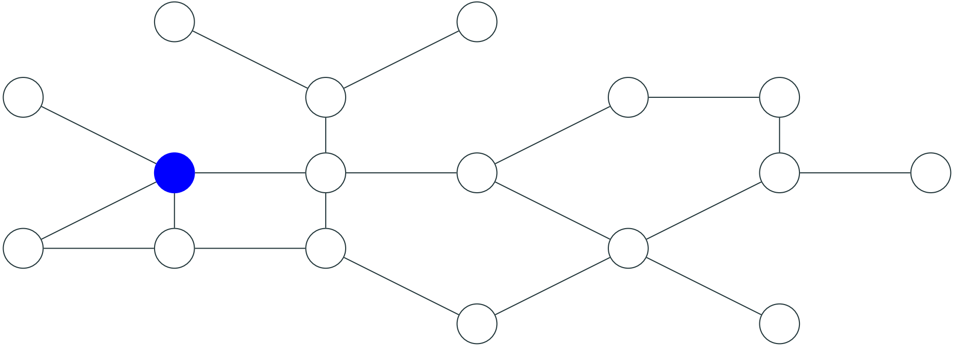


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

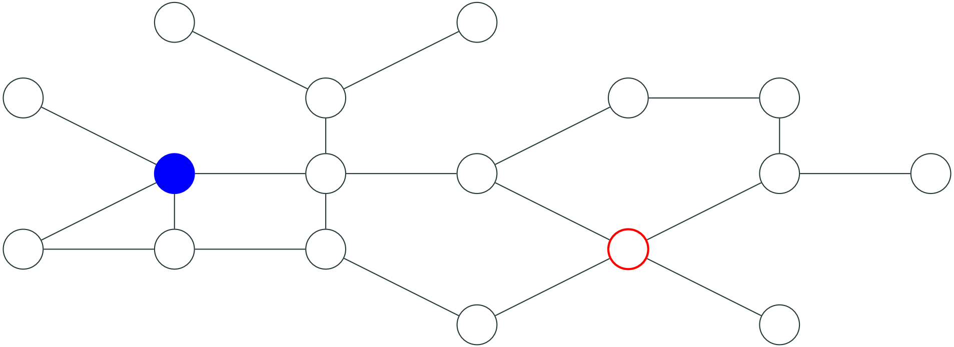


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

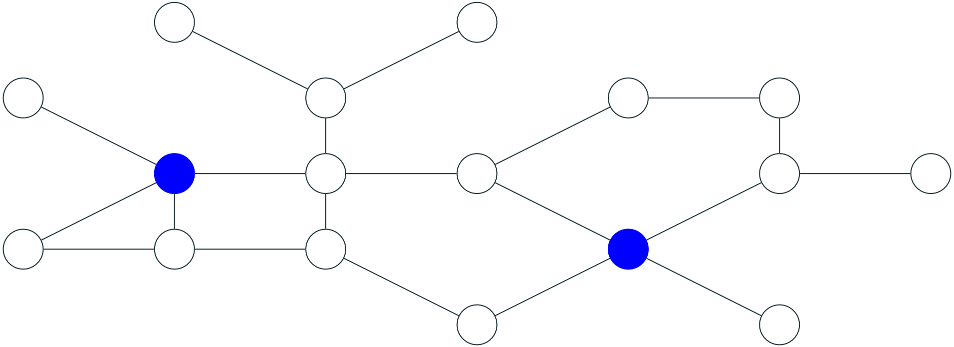


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

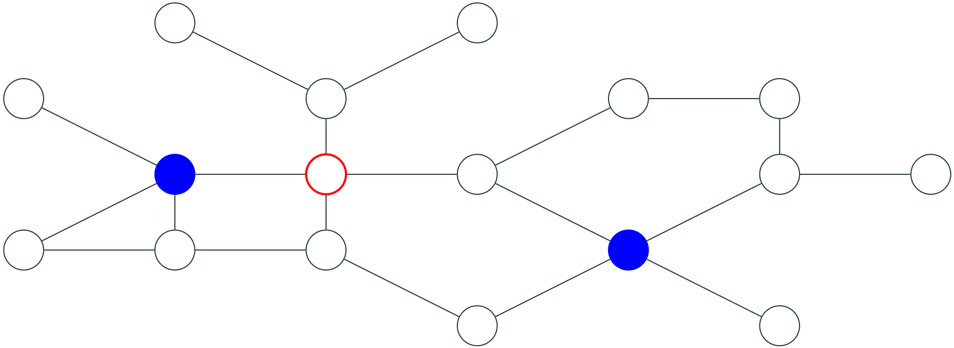


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

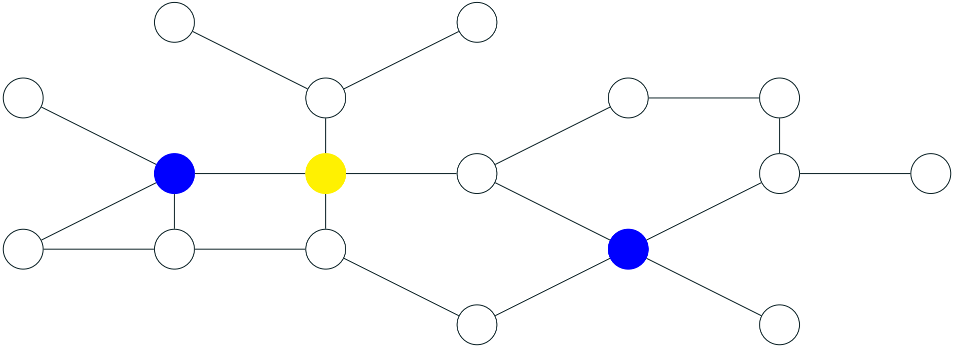
- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)



Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

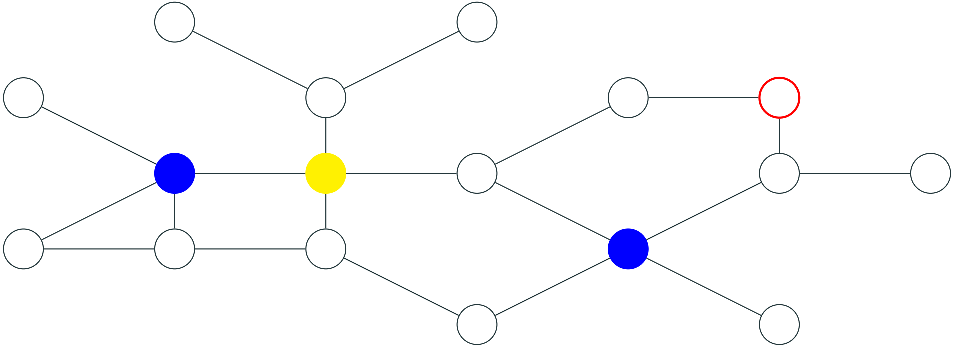


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

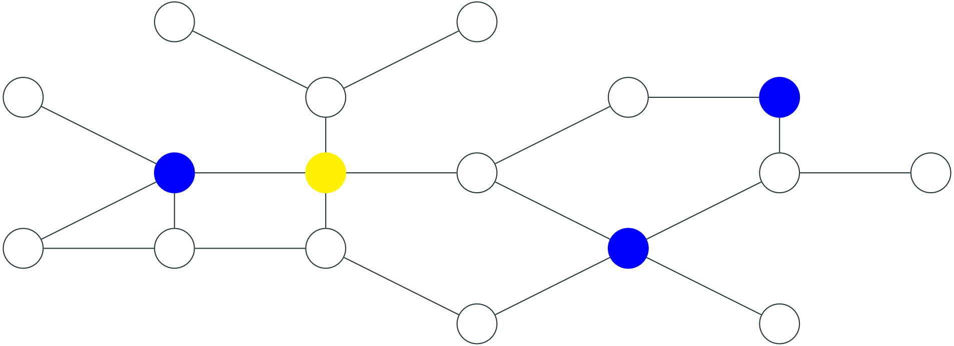


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

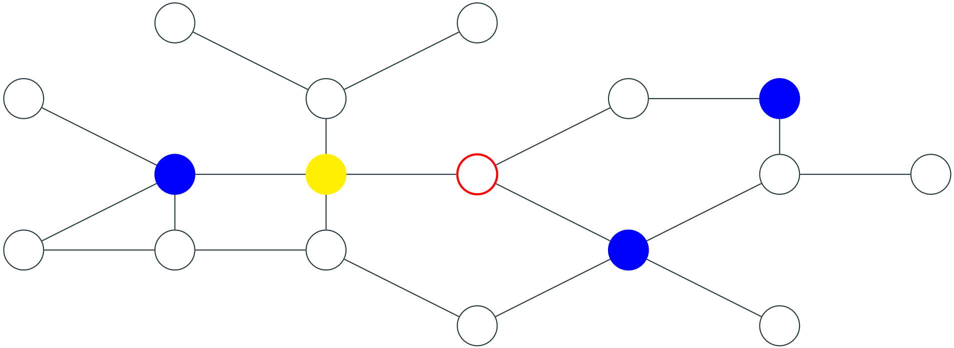


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

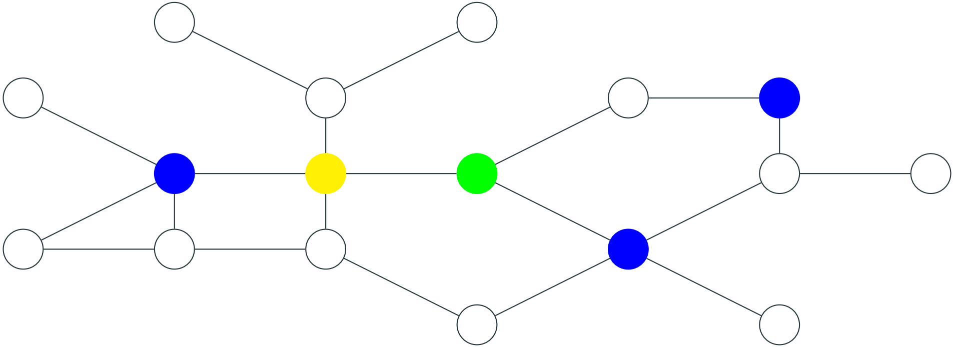
- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)



Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

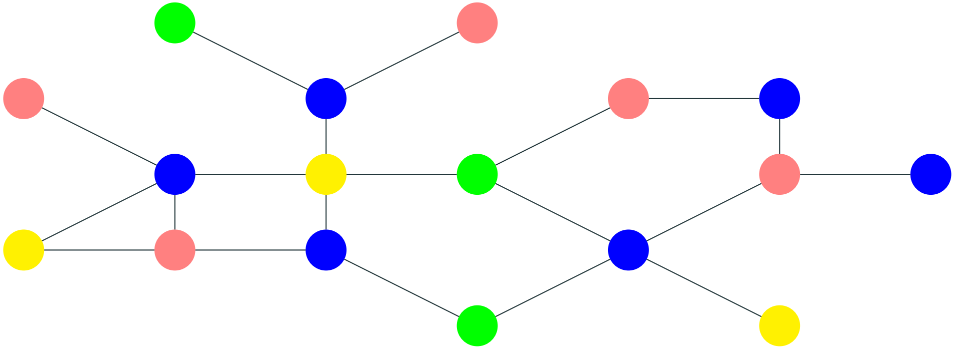


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)

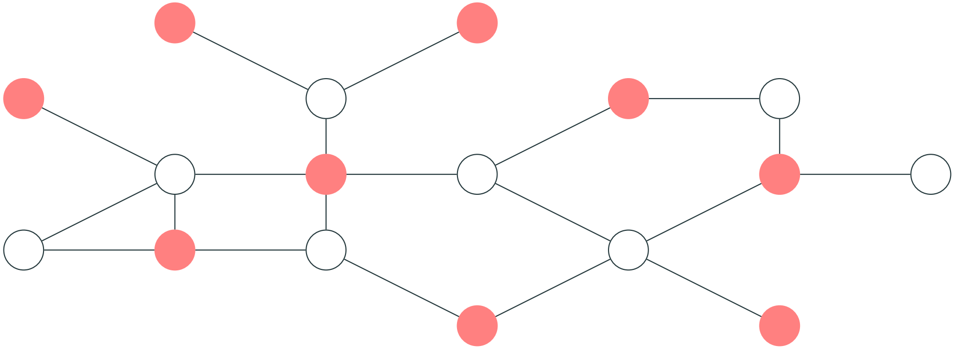


Locally Greedy Problems

Locally Greedy :

A solution can be completed considering nodes sequentially

- $\Delta + 1$-coloring
- Maximal Independent Set (MIS)



Mendable Problems

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

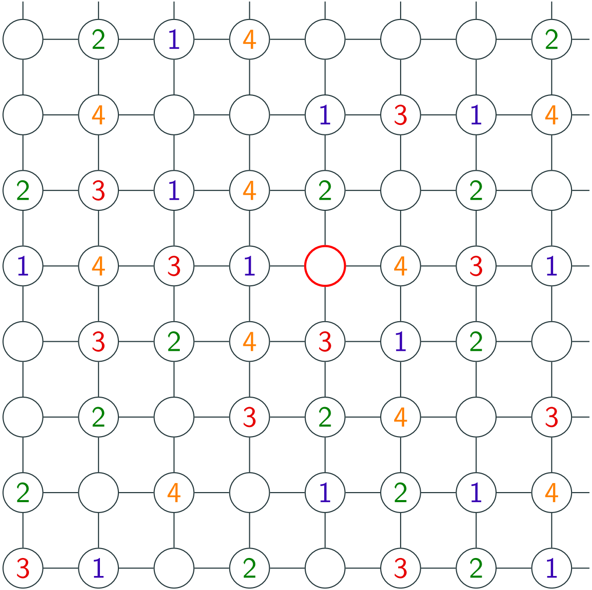
- $\mathcal{O}$ is the Output Set,
- $\forall u \in V : \Gamma^*(u) \neq \perp \Rightarrow$ we can complete the labels of the neighbors of u.

Balliu *et. al*, **Local Mending, 2022**

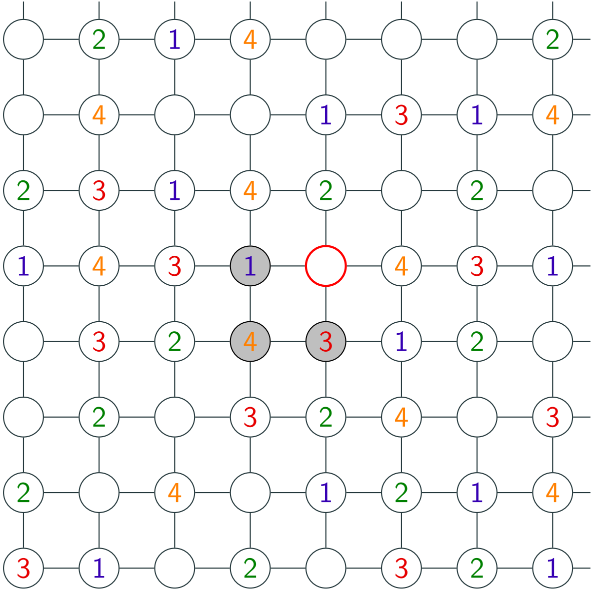
A problem is T-**Mendable** if, from any partial solution Γ^* and any $v \in V$ such that $\Gamma^*(v) = \perp$, there exists Γ' :

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

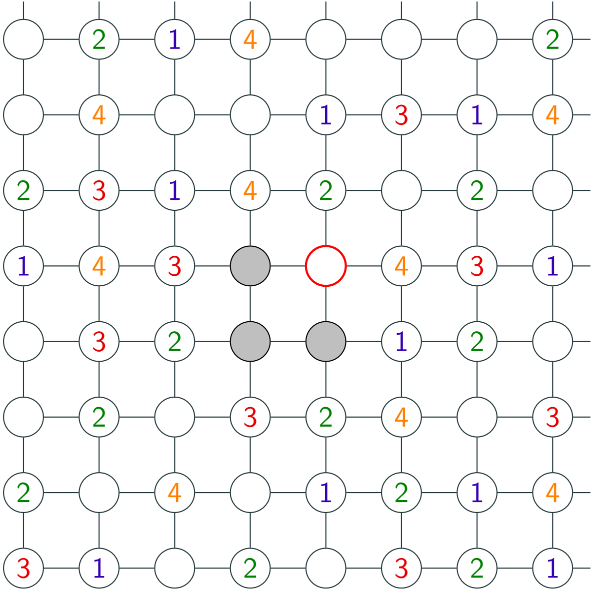
Example : 4-coloring the Grid



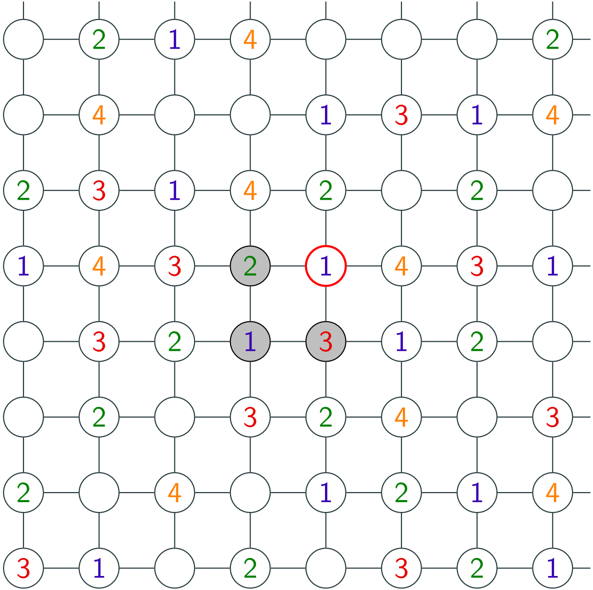
Example : 4-coloring the Grid



Example : 4-coloring the Grid



Example : 4-coloring the Grid

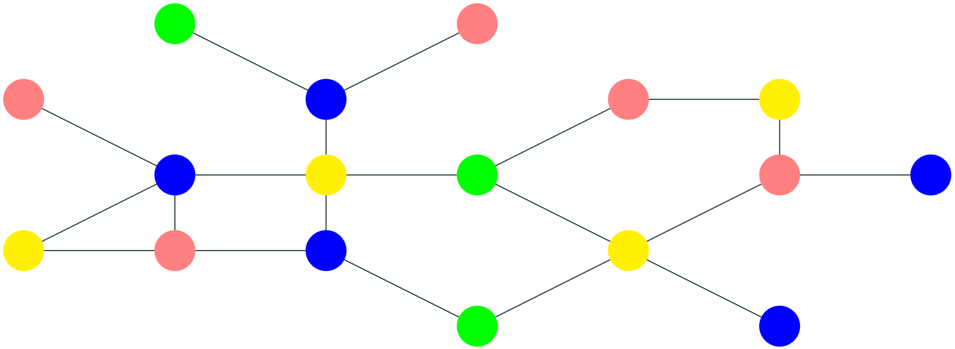


Self-Stabilization

Self-Stability : Possibility to reach a solution from any configuration.

Situations where self-stabilizing algorithms are needed :

- Failures
- *Slow* dynamic network

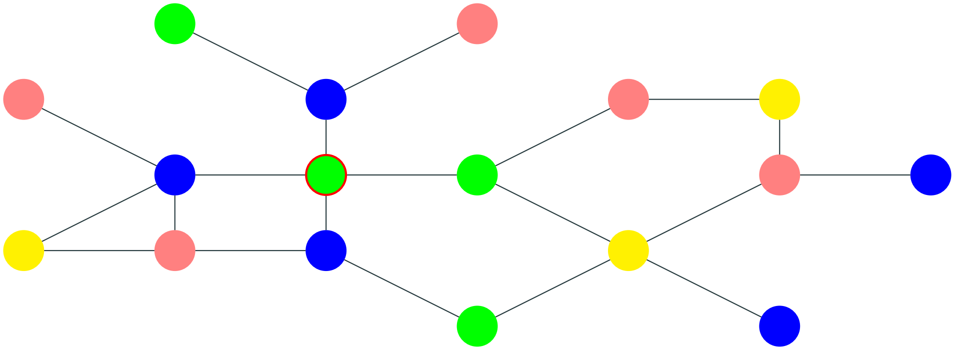


Self-Stabilization

Self-Stability : Possibility to reach a solution from any configuration.

Situations where self-stabilizing algorithms are needed :

- Failures
- *Slow* dynamic network

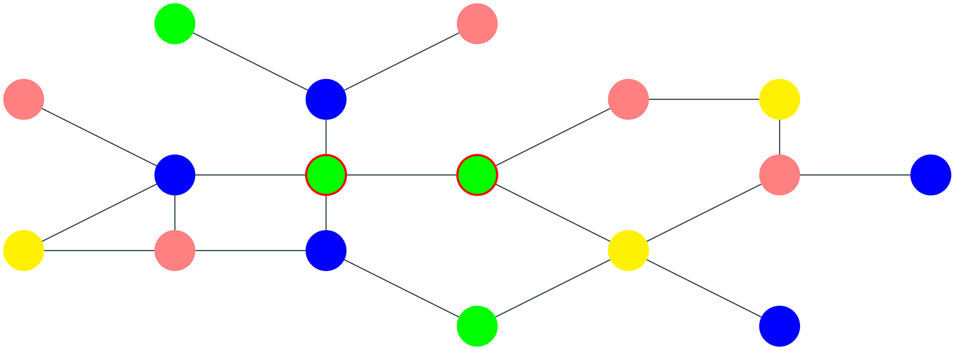


Self-Stabilization

Self-Stability : Possibility to reach a solution from any configuration.

Situations where self-stabilizing algorithms are needed :

- Failures
- *Slow* dynamic network

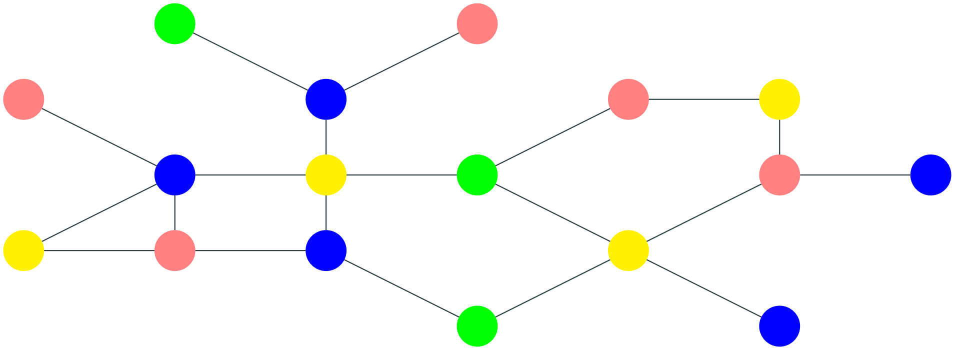


Self-Stabilization

Self-Stability : Possibility to reach a solution from any configuration.

Situations where self-stabilizing algorithms are needed :

- Failures
- *Slow* dynamic network

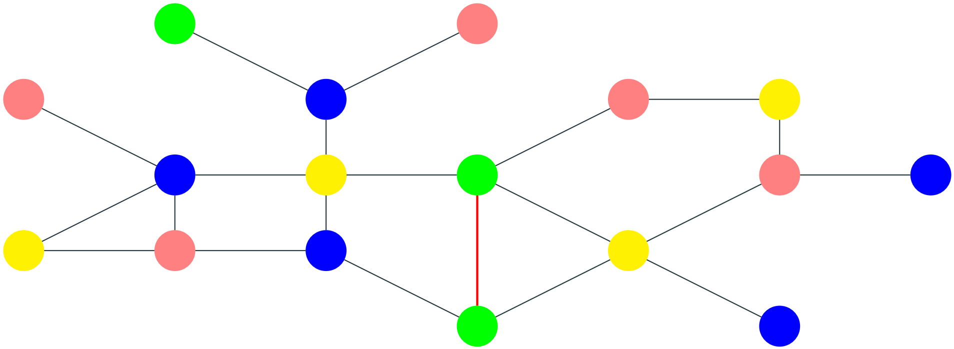


Self-Stabilization

Self-Stability : Possibility to reach a solution from any configuration.

Situations where self-stabilizing algorithms are needed :

- Failures
- *Slow* dynamic network

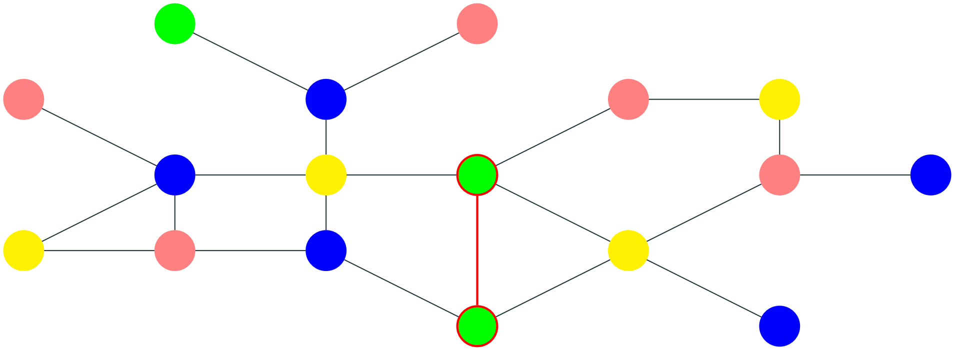


Self-Stabilization

Self-Stability : Possibility to reach a solution from any configuration.

Situations where self-stabilizing algorithms are needed :

- Failures
- *Slow* dynamic network



Computational Setting - State Model

Each node :

- Limited memory - $f(\Delta)$, no identifiers.
- When activated : has access to the memory of its neighbors.

Scheduler :

- Subset of agents activated at each round.
- Gouda Fair : If $\mathcal{C}$ appears infinitely often and $\mathcal{C} \to \mathcal{C}'$

  $\Rightarrow \mathcal{C}$' appears infinitely often.

  (Similar to probabilistic scheduler without time consideration.)

Goal :

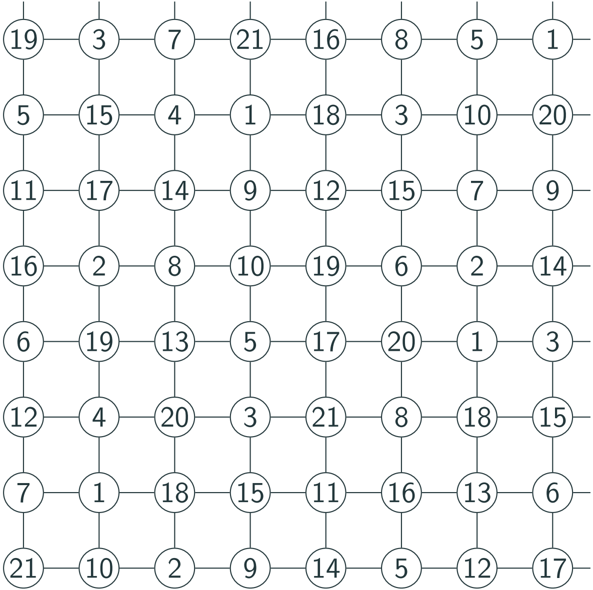
- Define the set of *Stable configurations* and their output.
- From any stable configuration, only stable configurations can be reached.
- From any configuration, a stable configuration is reachable.

From Distance-k Coloring to LOCAL

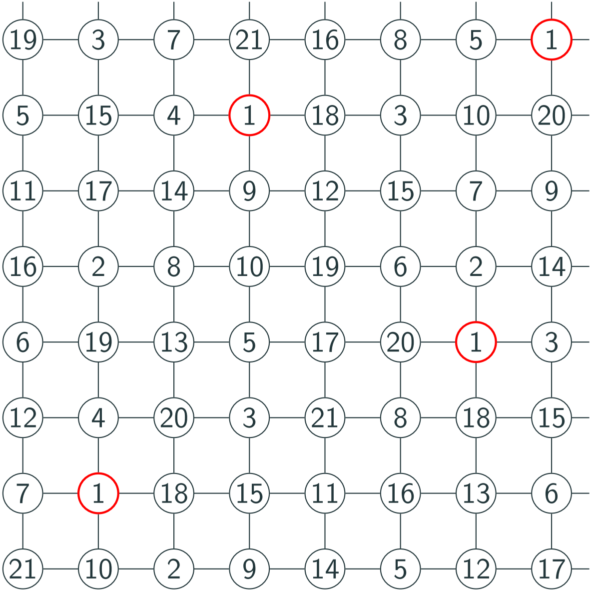
From Distance-K Coloring to Greedy Problems

$\mathcal{C} : V \to [c]$ is a **Distance-**K c-**Coloring** if
For all $uv \in V^2$, $dist(u, v) \leq K \Rightarrow \mathcal{C}(u) \neq \mathcal{C}(v)$.

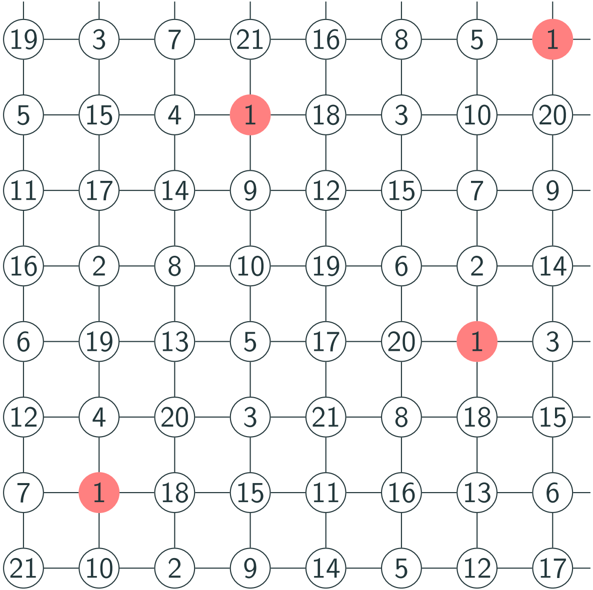
From Distance-K Coloring to Greedy Problems



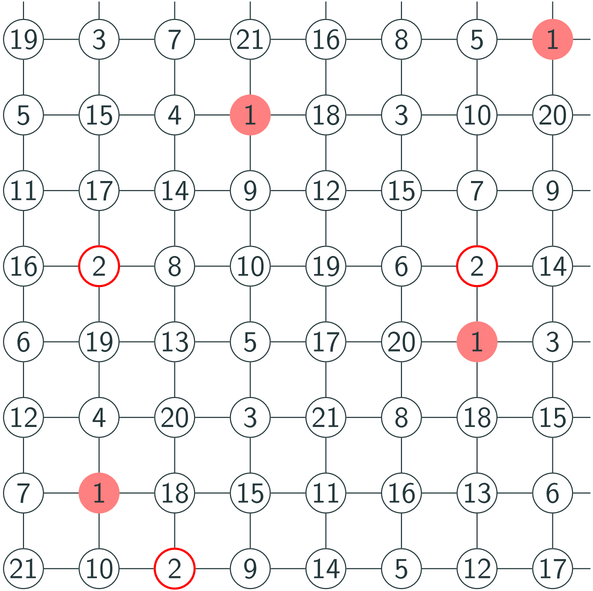
From Distance-K Coloring to Greedy Problems



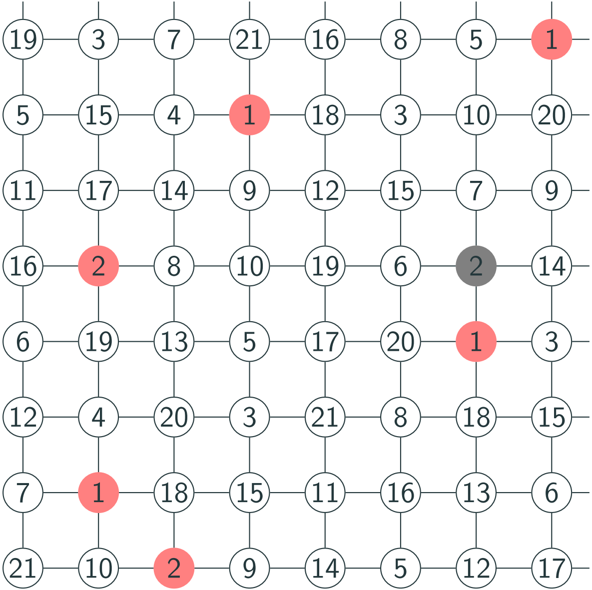
From Distance-K Coloring to Greedy Problems



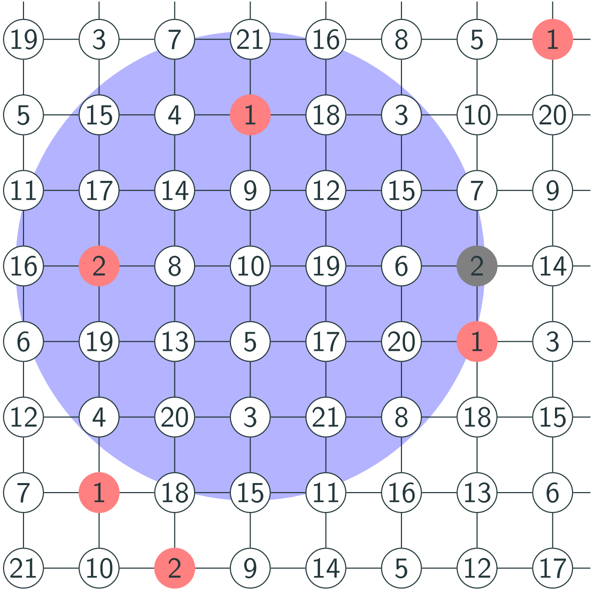
From Distance-K Coloring to Greedy Problems



From Distance-K Coloring to Greedy Problems



From Distance-K Coloring to Greedy Problems



Local Simulation

Local Model :

- Unique identifiers.
- Synchronous rounds.
- k-rounds is equivalent to knowing neighborhood at distance k.

Local Simulation :

- From a distance-$2K$ coloring, a node can compute its radius-K neighborhood :
  - At first, each node knows its radius-0 neighborhood.
  - With the radius-0 neighborhood of your neighbors, you know your radius-1.
  - With the radius-i neighborhood of your neighbors, you know your radius-$i+1$.
- Having unique colors/identifiers in radius-i removes any ambiguity.
- Each node can simulate a Local algorithm that runs on at most K rounds.

Solving Mendable Problems

Balliu ***et. al*****, Local Mending, 2022**

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

How to simulate the $\textsc{Local}$ algorithm :

- Compute the distance-$2T+1$ coloring.
- Compute a distance-$O(T\Delta^{2T})$ coloring for making fake "unique identifiers".
- For each node, compute the radius-$O(T\Delta^{2T})$ neighborhood.
- Give the output of the $\textsc{Local}$ algorithm on this neighborhood.

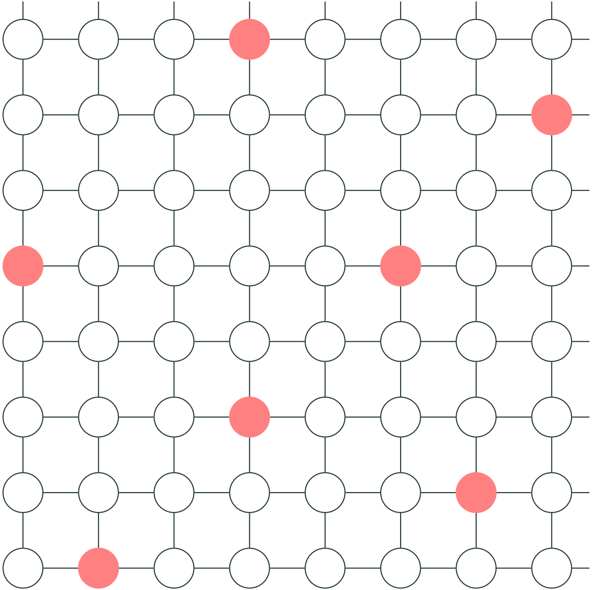
Computing a $(k, k-1)$-Ruling Set

$(k, k-1)$-Ruling Set

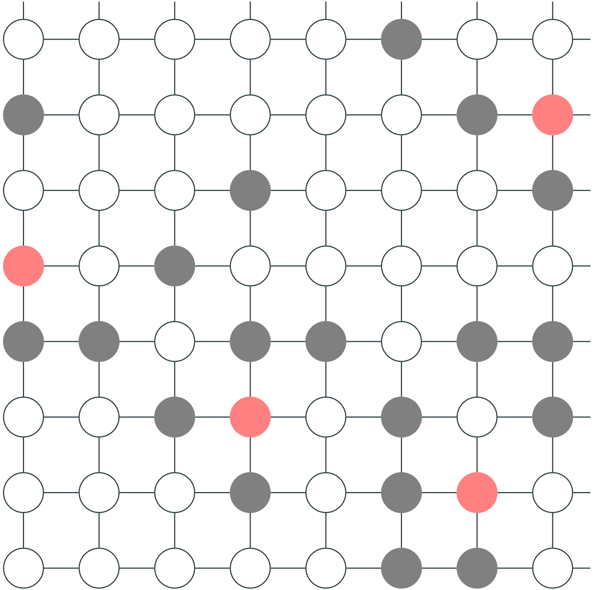
(a, b)-Ruling Set S :

- $\forall u, v \in S^2$, $dist(u, v) \geq a$.
- $\forall u \notin S$, $dist(u, S) \leq b$.
- $(2, 1)$-Ruling Set is MIS.
- $(k, k-1)$-Ruling Set is Distance-k MIS.
- A distance-k coloring is a partition into $(k+1, k)$-ruling sets :
  Each set corresponds to one of the colors.

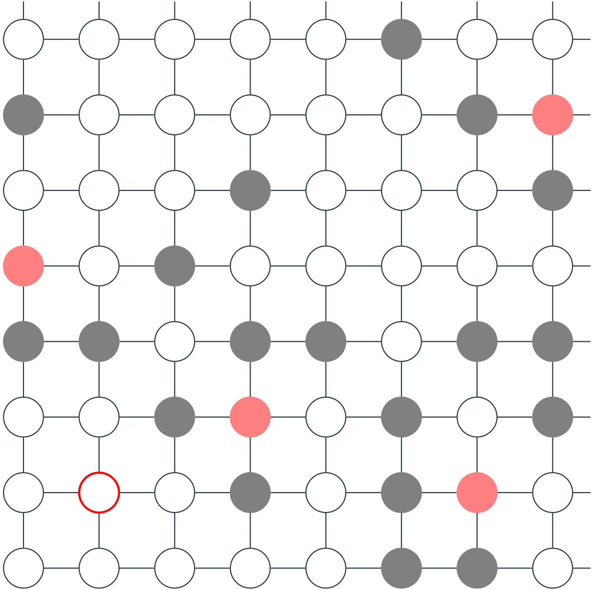
$(k, k-1)$-Ruling Set



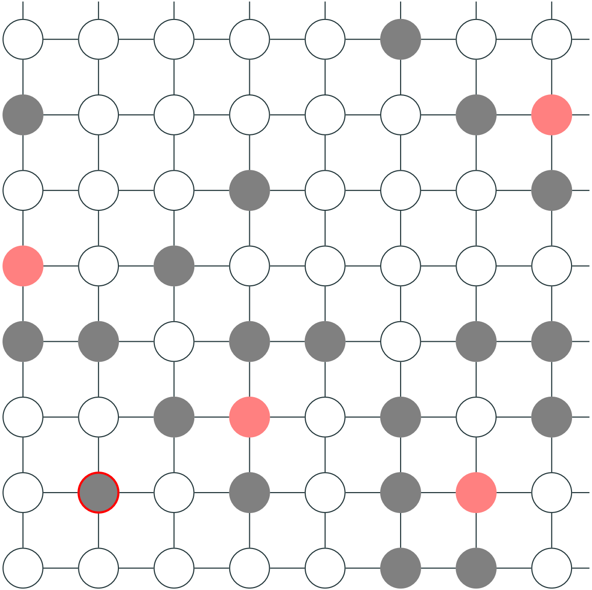
$(k, k-1)$-Ruling Set



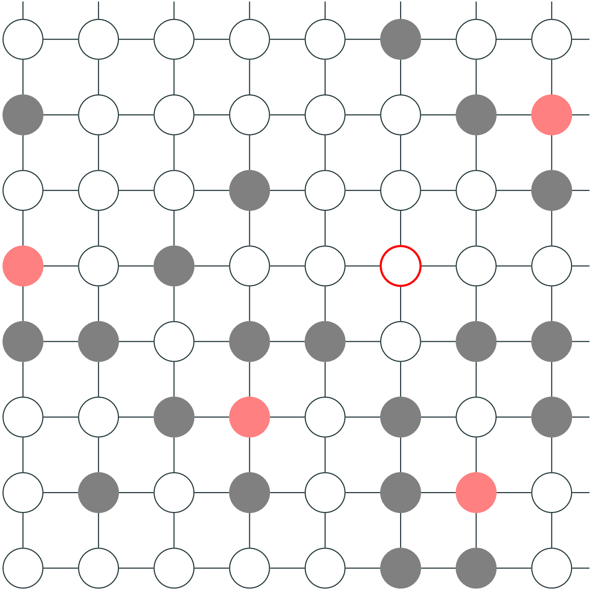
$(k, k-1)$-Ruling Set



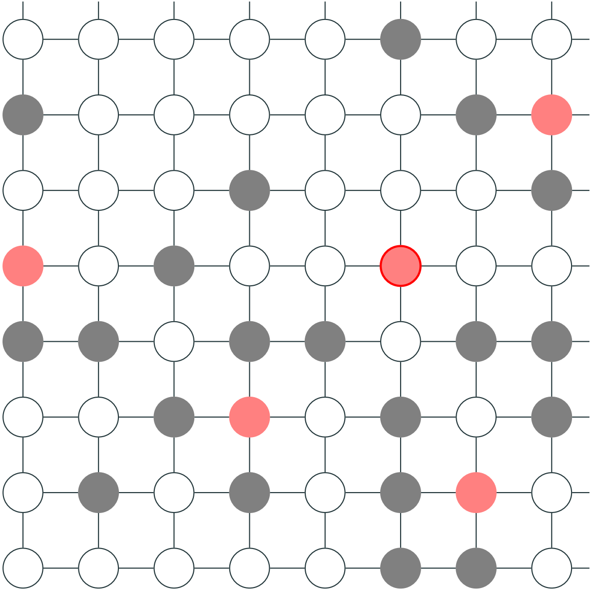
$(k, k-1)$-Ruling Set



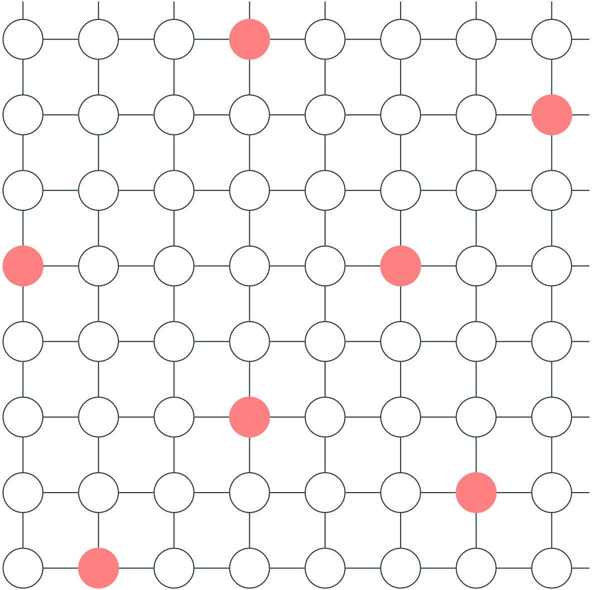
$(k, k-1)$-Ruling Set



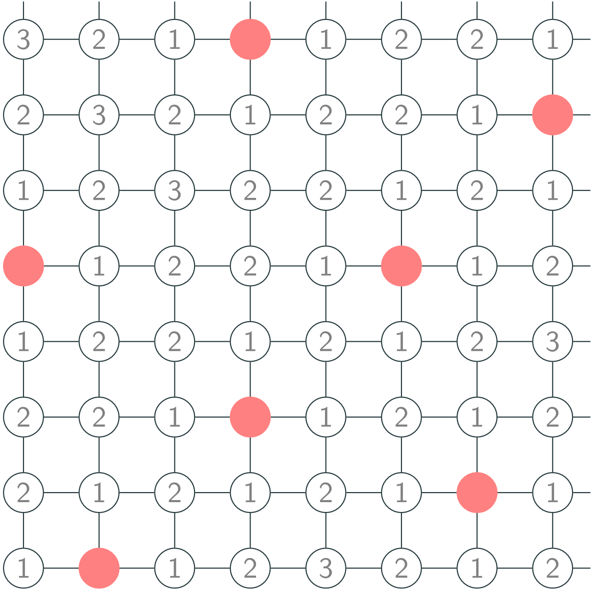
$(k, k-1)$-Ruling Set



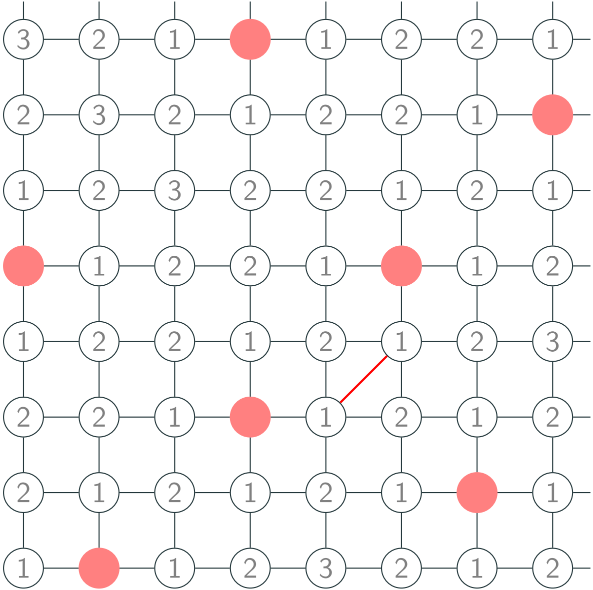
$(k, k-1)$-Ruling Set



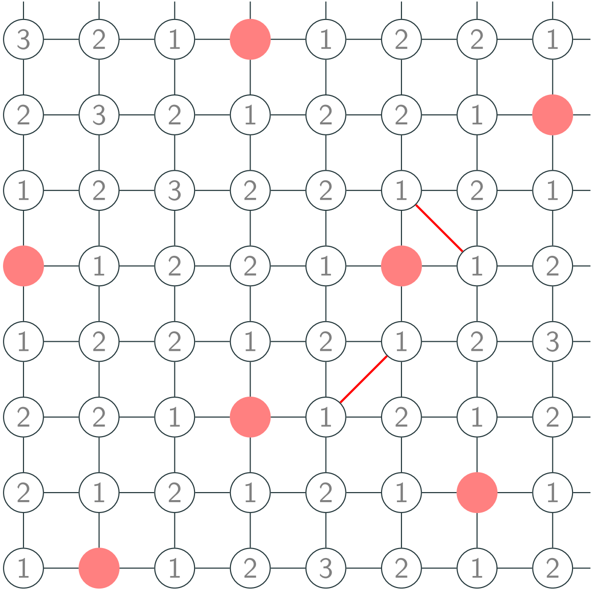
$(k, k-1)$-Ruling Set



$(k, k-1)$-Ruling Set



$(k, k-1)$-Ruling Set



Idea of the Algorithm

Each node u has 3 elements :

- $\mathrm{d}_u \in [\![0, k-1]\!]$
  - Each node keeps their distance to the ruling set candidates.
  - $d_u = 0$ means that u is candidate (called Leader).
- $\mathrm{err}_u \in \{0, 1\}$
  - Nodes keep in their memory if an error is detected.
  - Errors are transmitted to nodes with smaller d_u.
  - Leaders with an error reset $d_u = k - 1$.
  - When $d_u = k - 1$, the error is removed.
- For every $i \in [\![1, \left\lfloor \frac{k}{2} \right\rfloor - 1]\!]$: $\mathrm{c}_{i,u} \in \mathbb{Z}/4\mathbb{Z}$ and $\mathrm{b}_{i,u} \in \{\uparrow, \downarrow\}$
  - Clock system to synchronize to the closest leader.
  - One clock for each distance from d_u to $\left\lfloor \frac{k}{2} \right\rfloor$.

Keeping your Distances

Each node u stores a distance integer $d_u \in [0, k-1]$:

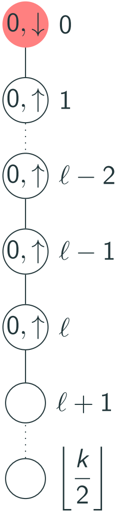
- If $d_u = 0$, u is a leader (i.e. une the Ruling Set)
- If $d_u > 0$, we need that $\min\{d_v : v \in N(u)\} = d_u - 1$

  If it is not the case, $d_u = \min\{\min\{d_v : v \in N(u)\} + 1, k-1\}$
- If $\forall v \in N(u)$, $d_v = k-1$, $d_u = 0$

  u becomes a new leader.

  (Scheduler ensures that several nodes in that situation will not be an issue.)
- If there is $v \in N(u)$ with $|d_u - d_v| \geq 2$, $err_u = 1$.
- If there are two leaders v_1 and v_2 in your closed neighborhood, $d_{v_1} = d_{v_2} = k-1$.

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}$; $b_{\ell,u} := \downarrow$

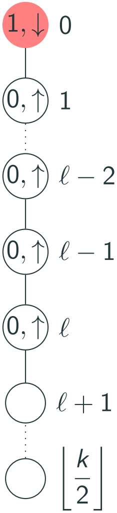
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow$; $c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}$; $b_{\ell,u} := \downarrow$

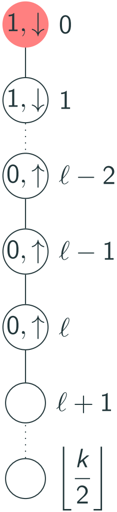
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow$; $c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}\,;\, b_{\ell,u} := \downarrow$

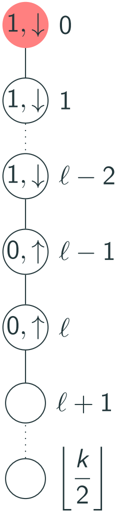
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow\,;\, c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}\,;\ b_{\ell,u} := \downarrow$

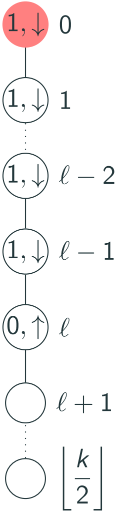
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow\,;\ c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} =\downarrow)$
then $c_{\ell,u} := c_{\ell,v}$; $b_{\ell,u} :=\downarrow$

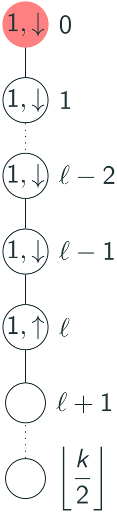
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} =\uparrow)$
then $b_{\ell,u} :=\uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} =\downarrow)$
then $b_{d_u,u} :=\uparrow$; $c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}\,;\, b_{\ell,u} := \downarrow$

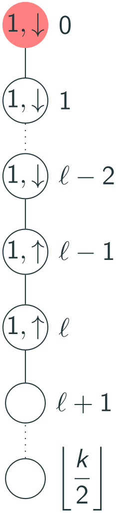
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow\,;\, c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}$; $b_{\ell,u} := \downarrow$

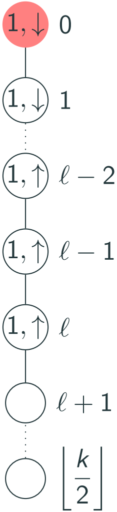
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow$; $c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} =\downarrow)$
then $c_{\ell,u} := c_{\ell,v}$; $b_{\ell,u} :=\downarrow$

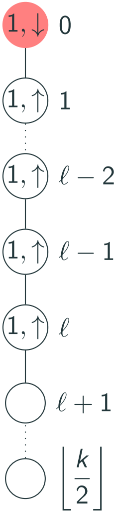
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} =\uparrow)$
then $b_{\ell,u} :=\uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} =\downarrow)$
then $b_{d_u,u} :=\uparrow$; $c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}$; $b_{\ell,u} := \downarrow$

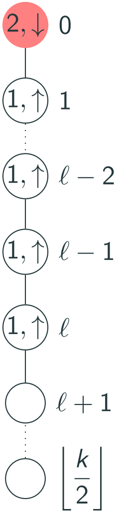
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow$; $c_{d_u,u} := c_{\ell,v}$

Synchronization Waves



Incr Leader : :

if $(d_u = 0) \wedge (\forall v \in N(u), d_v = 1 \wedge c_{\ell,u} - c_{\ell,v} = 0)$
then $c_{\ell,u} := c_{\ell,u} + 1$

Sync 1+ down : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u - 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $c_{\ell,u} := c_{\ell,v}\, ;\, b_{\ell,u} := \downarrow$

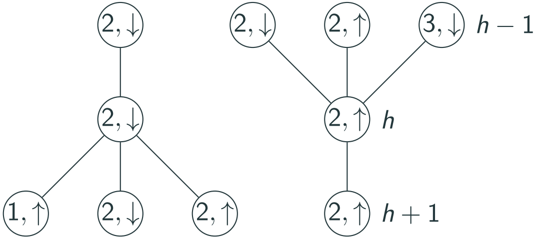
Sync 1+ up : :

if $0 < d_u < \ell \wedge \forall v \in N(u), d_v = d_u + 1 \Rightarrow (c_{\ell,u} = c_{\ell,v} \wedge b_{\ell,v} = \uparrow)$
then $b_{\ell,u} := \uparrow$

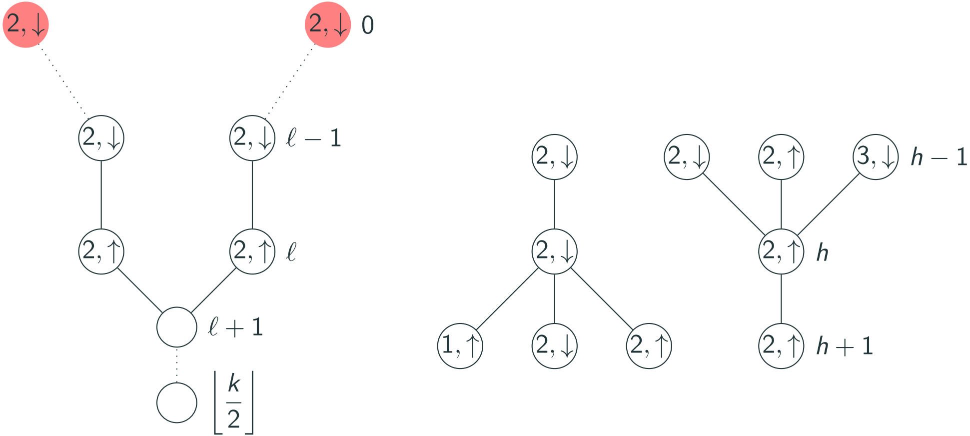
Sync end-of-chain : :

if $d_u = \ell \wedge \forall v \in N(u), d_v = \ell - 1 \Rightarrow (c_{d_u,u} = c_{d_u,v} - 1 \wedge b_{\ell,v} = \downarrow)$
then $b_{d_u,u} := \uparrow\, ;\, c_{d_u,u} := c_{\ell,v}$

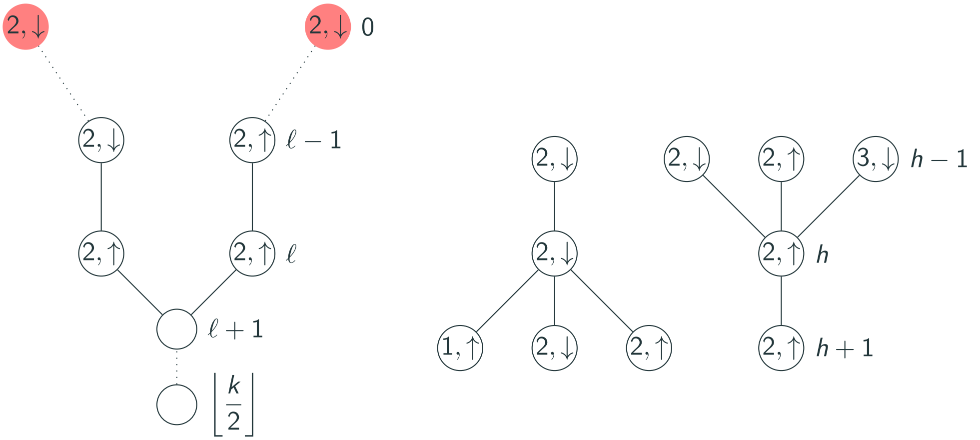
Detecting 2 Too Close Leaders



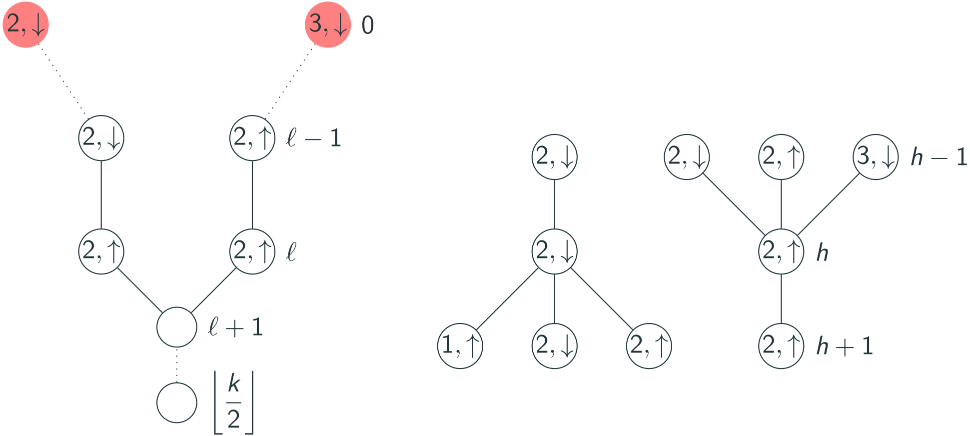
Detecting 2 Too Close Leaders



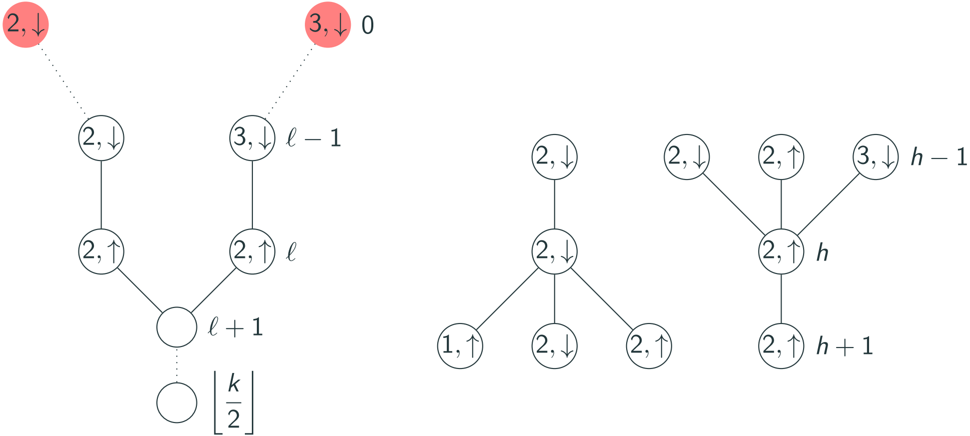
Detecting 2 Too Close Leaders



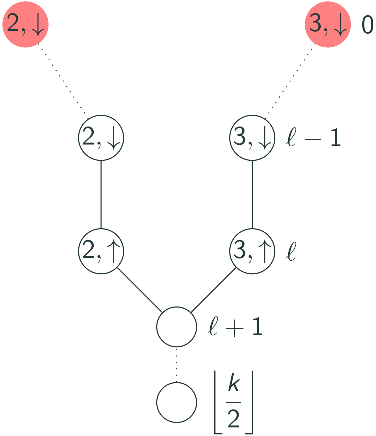
Detecting 2 Too Close Leaders

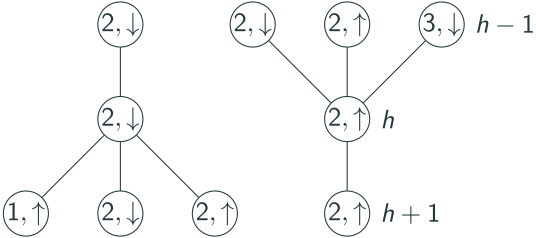


Detecting 2 Too Close Leaders

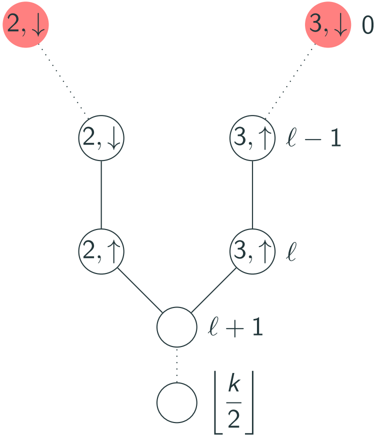


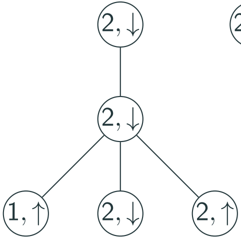
Detecting 2 Too Close Leaders

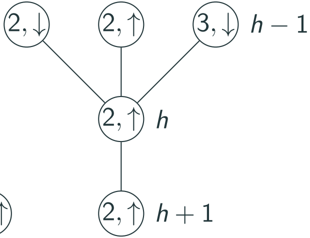




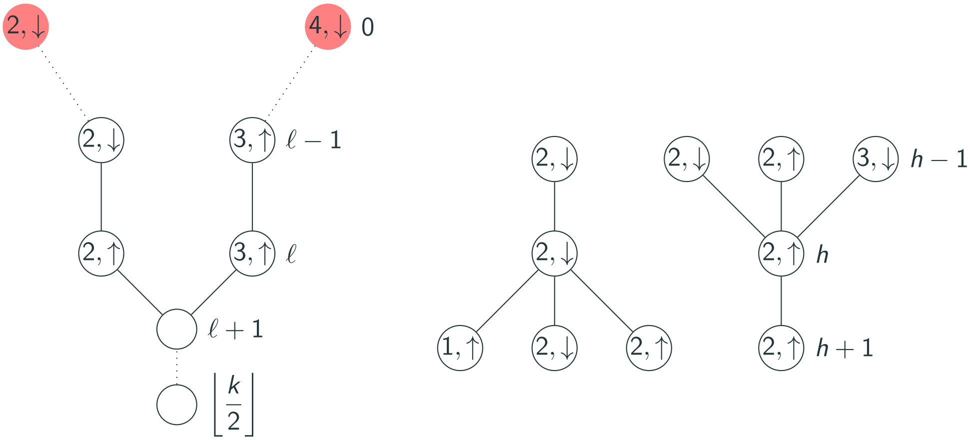
Detecting 2 Too Close Leaders



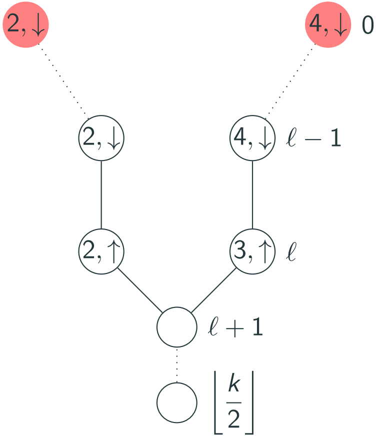


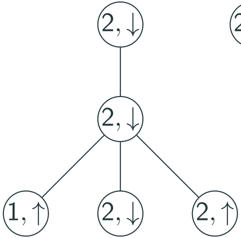


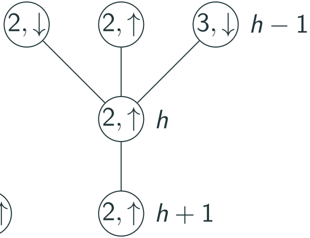
Detecting 2 Too Close Leaders



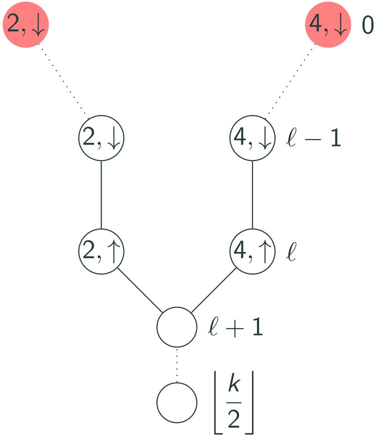
Detecting 2 Too Close Leaders

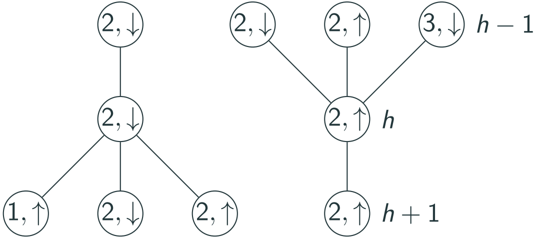




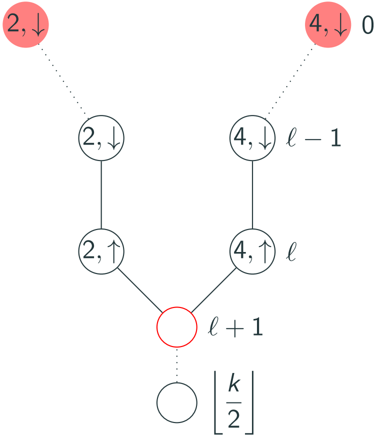


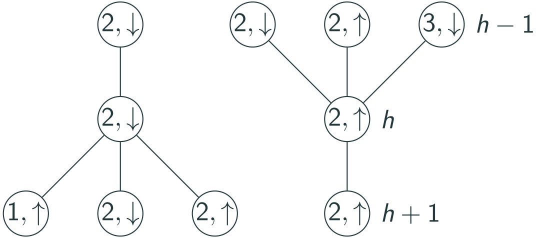
Detecting 2 Too Close Leaders



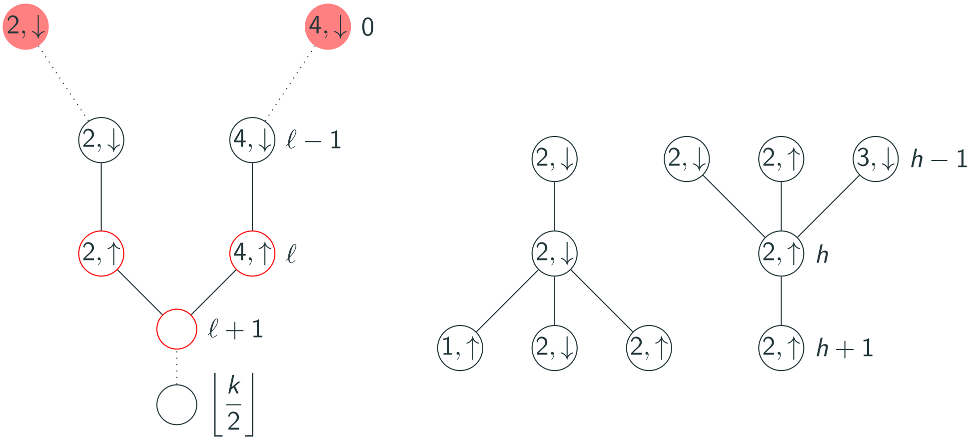


Detecting 2 Too Close Leaders

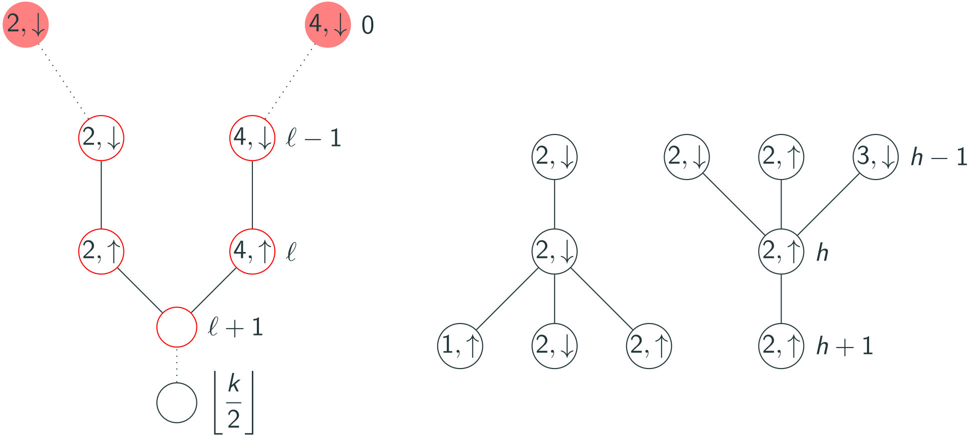




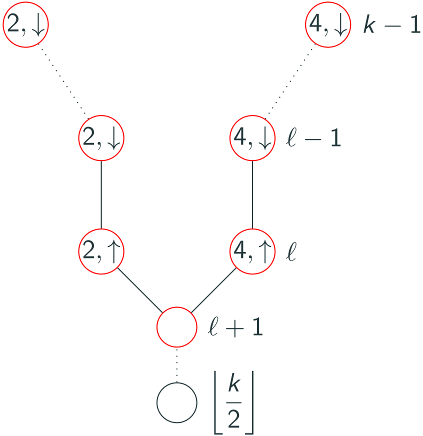
Detecting 2 Too Close Leaders

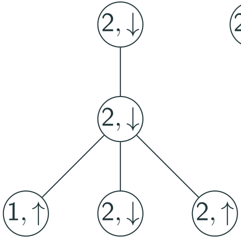


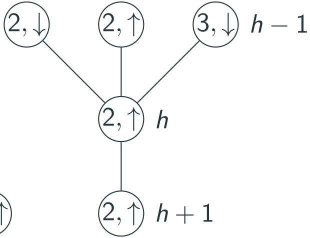
Detecting 2 Too Close Leaders



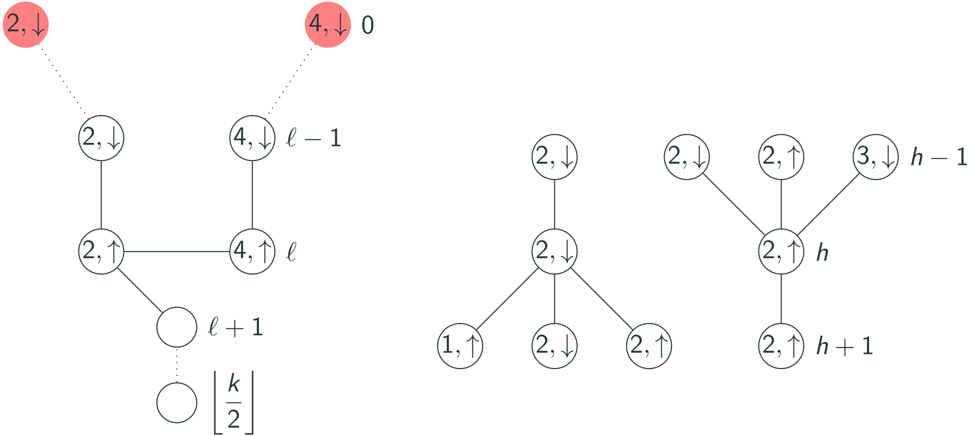
Detecting 2 Too Close Leaders



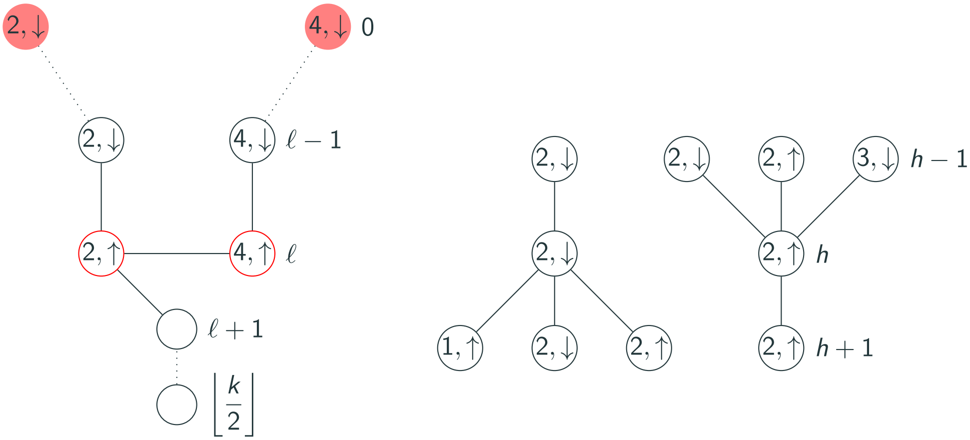




Detecting 2 Too Close Leaders



Detecting 2 Too Close Leaders



Stable Configurations

A leader u is **Locally Stable** if :

- $N\left(u, \frac{k}{2}\right)$ is synchronized with the right distances.
- $N(u, k-1) \setminus N\left(u, \frac{k}{2}\right)$ have coherent distances.

A configuration is **Stable** if :

- The set S of leaders is a $(k, k-1)$-ruling set.
- $N\left(S, \frac{k}{2}\right)$ is synchronized with their leaders.
- Each node knows its exact distance to S.

From any configuration :

- If two leaders are too close, we can remove them.
- If a leader has no leader too close, we can make it locally stable.

Computing a Distance-K Coloring

$\mathcal{C} : V \to [c]$ is a **Distance-K c-Coloring** if
For all $uv \in V^2$, $dist(u, v) \leq K \Rightarrow \mathcal{C}(u) \neq \mathcal{C}(v)$

- Such a coloring can be created with a partition into $(K + 1, K)$-ruling sets :
  Each set corresponds to one of the colors.
- Our algorithm can be composed.

Theorem
There exists a distance$-K$ Δ^K-coloring self-stabilizing algorithm using $f(\Delta, K)$ states.

Conclusion

Open Questions

Theorem

Let Π be a problem with mending radius k. Π can be solved under the Gouda Daemon by a self-stabilizing algorithm, using $f(\Delta, \Pi)$ states.

Open Questions

Theorem
Let Π be a problem with mending radius k. Π can be solved under the Gouda Daemon by a self-stabilizing algorithm, using $f(\Delta, \Pi)$ states.

- With randomness, what should be the complexity ?
- Can we use other Daemons to compute a ruling set ?
- Can we be robust to byzantine agents ?

Open Questions

Theorem
Let Π be a problem with mending radius k. Π can be solved under the Gouda Daemon by a self-stabilizing algorithm, using $f(\Delta, \Pi)$ states.

- With randomness, what should be the complexity ?
- Can we use other Daemons to compute a ruling set ?
- Can we be robust to byzantine agents ?