# Proving bounds on locality of distributed computing

*Juho Hirvonen*
IRIF, CNRS, and Université Paris Diderot

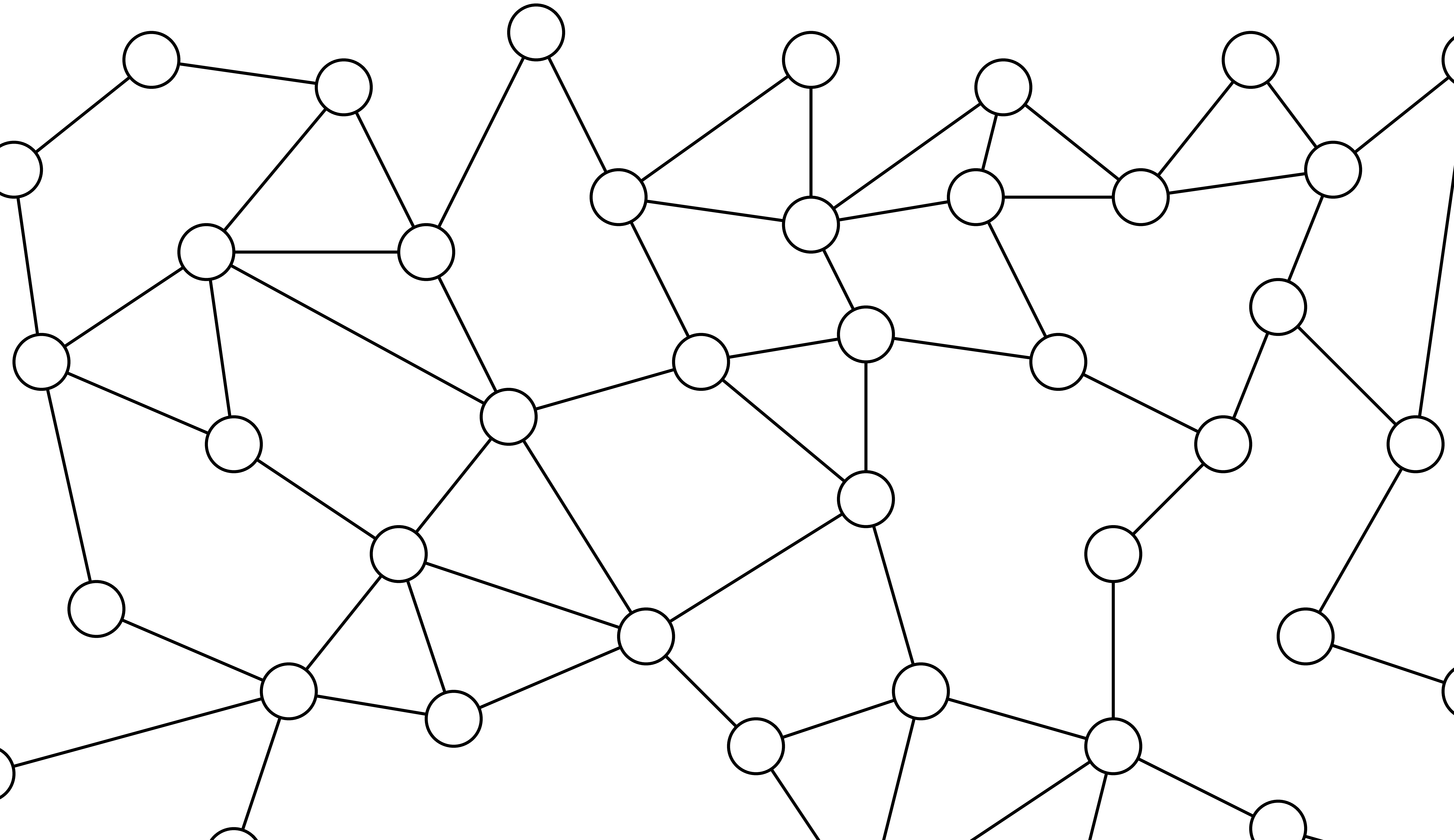
CoA, Lyon, 28 November 2017

# Talk outline

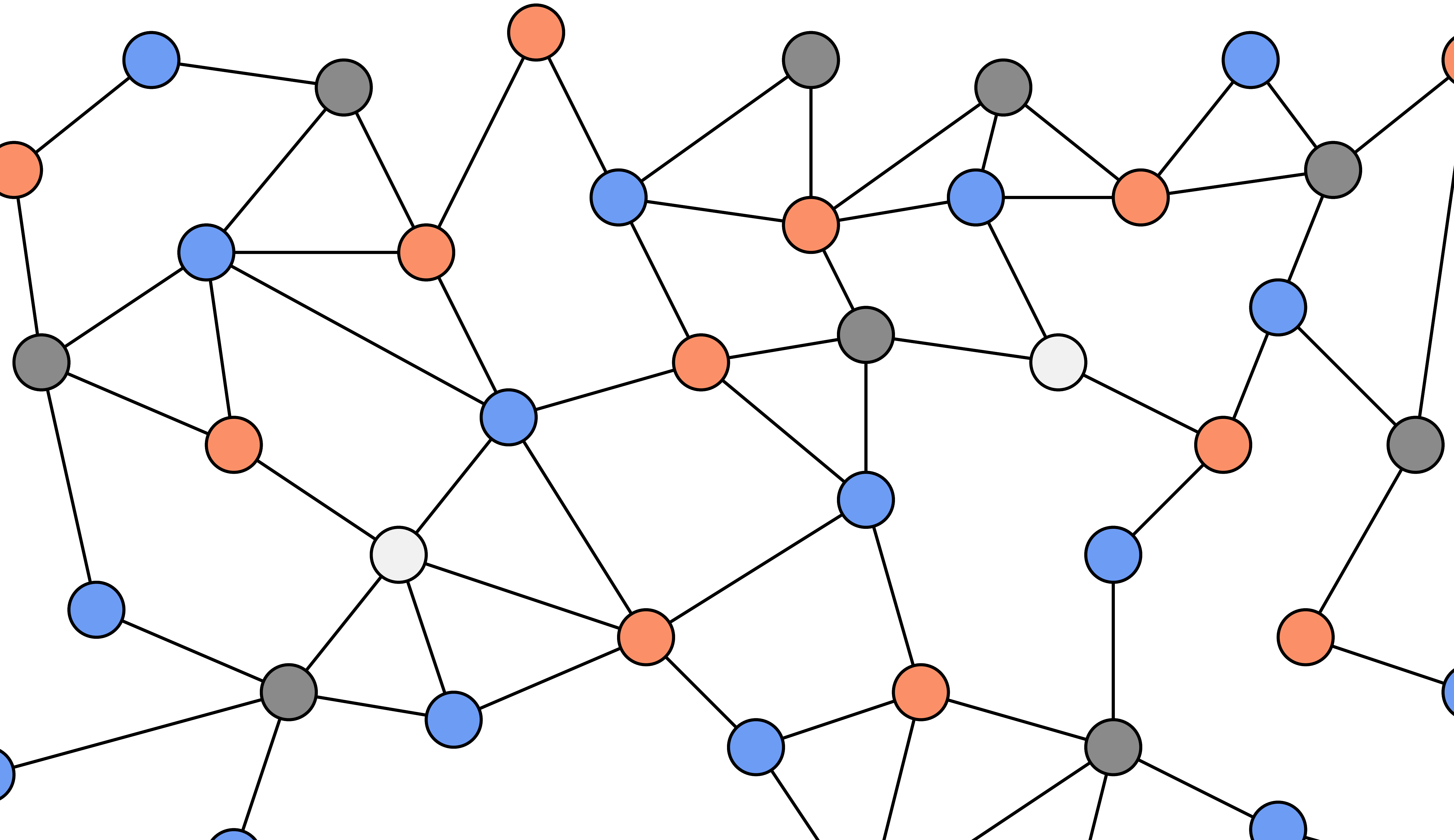
- Sketch a lower bound **proof technique** for distributed graph algorithms
- In general, **simulation** is a very powerful tool for lower bounds
- We have the beginnings of a **complexity theory**: can use heavy hammers in lower bound proofs

# Modeling locality

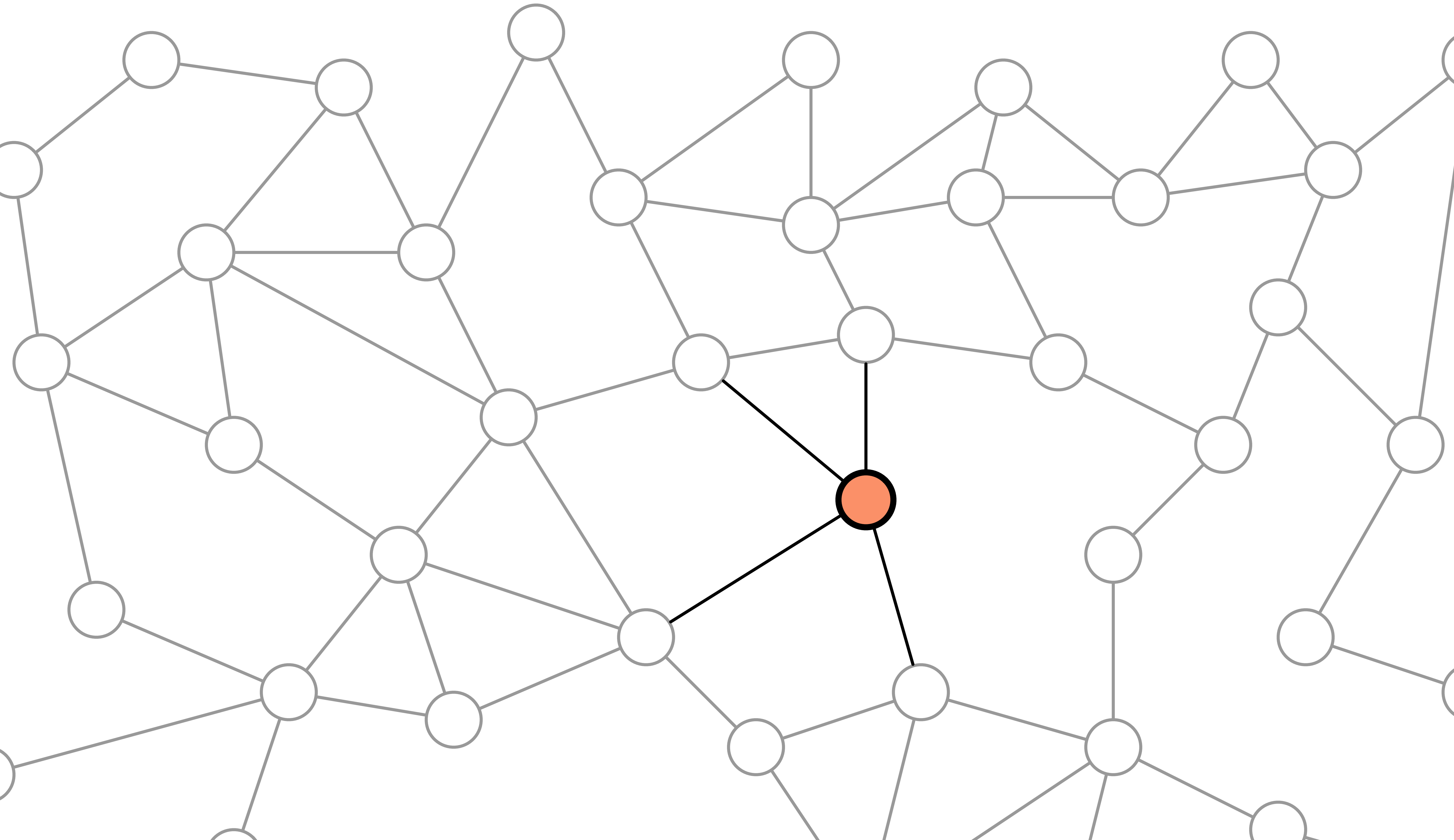
# Locality



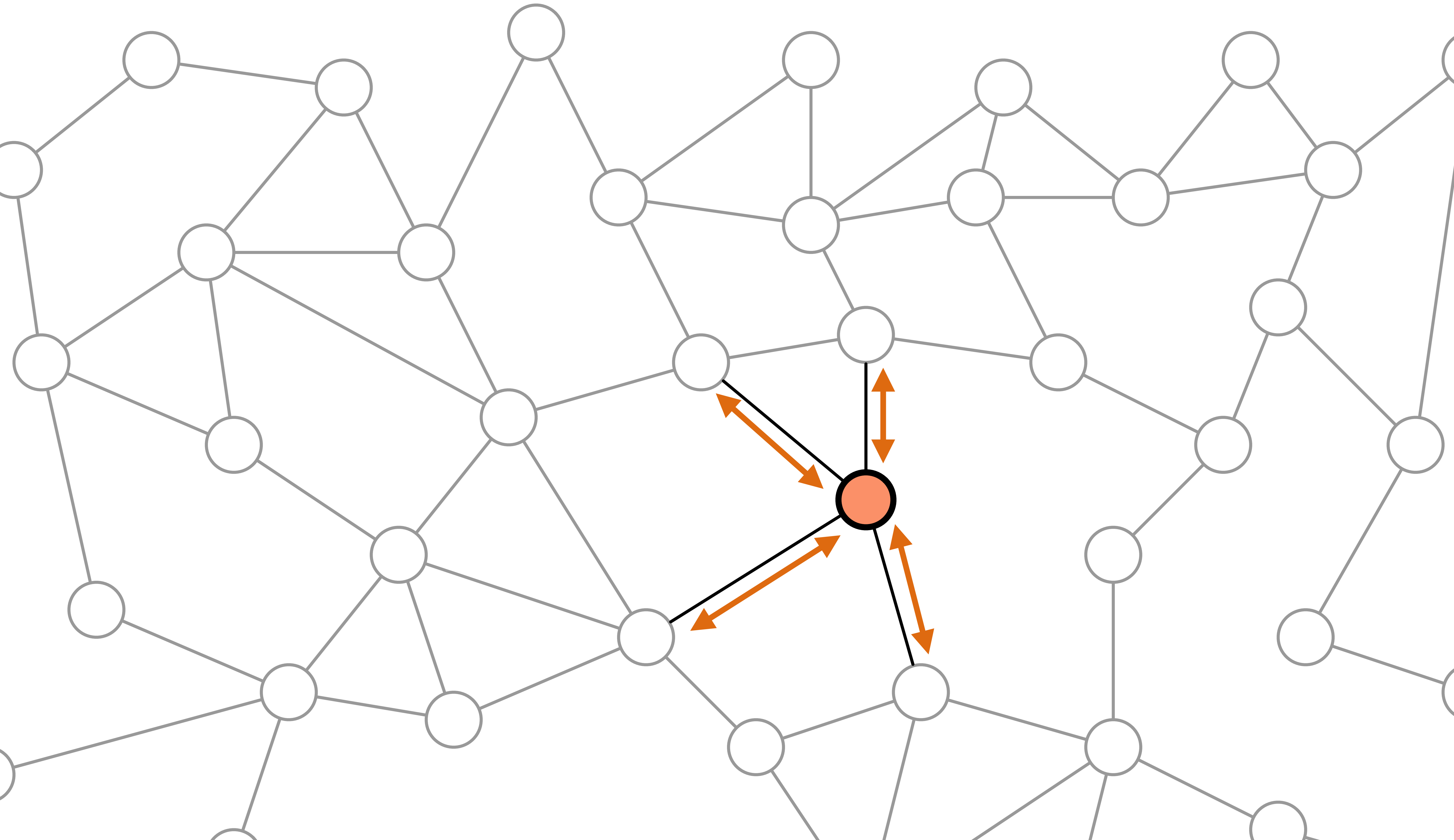
# Locality



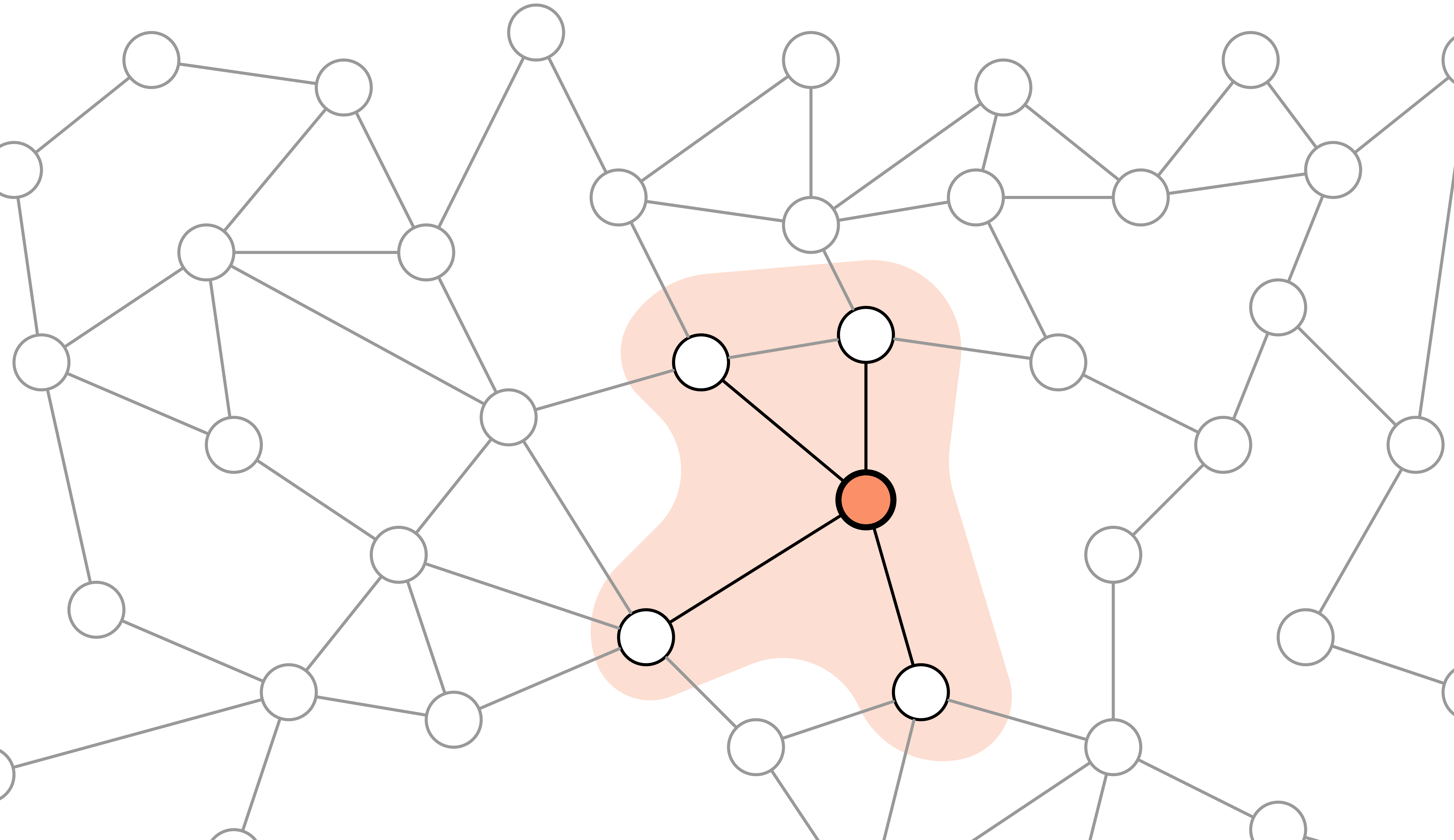
# Locality



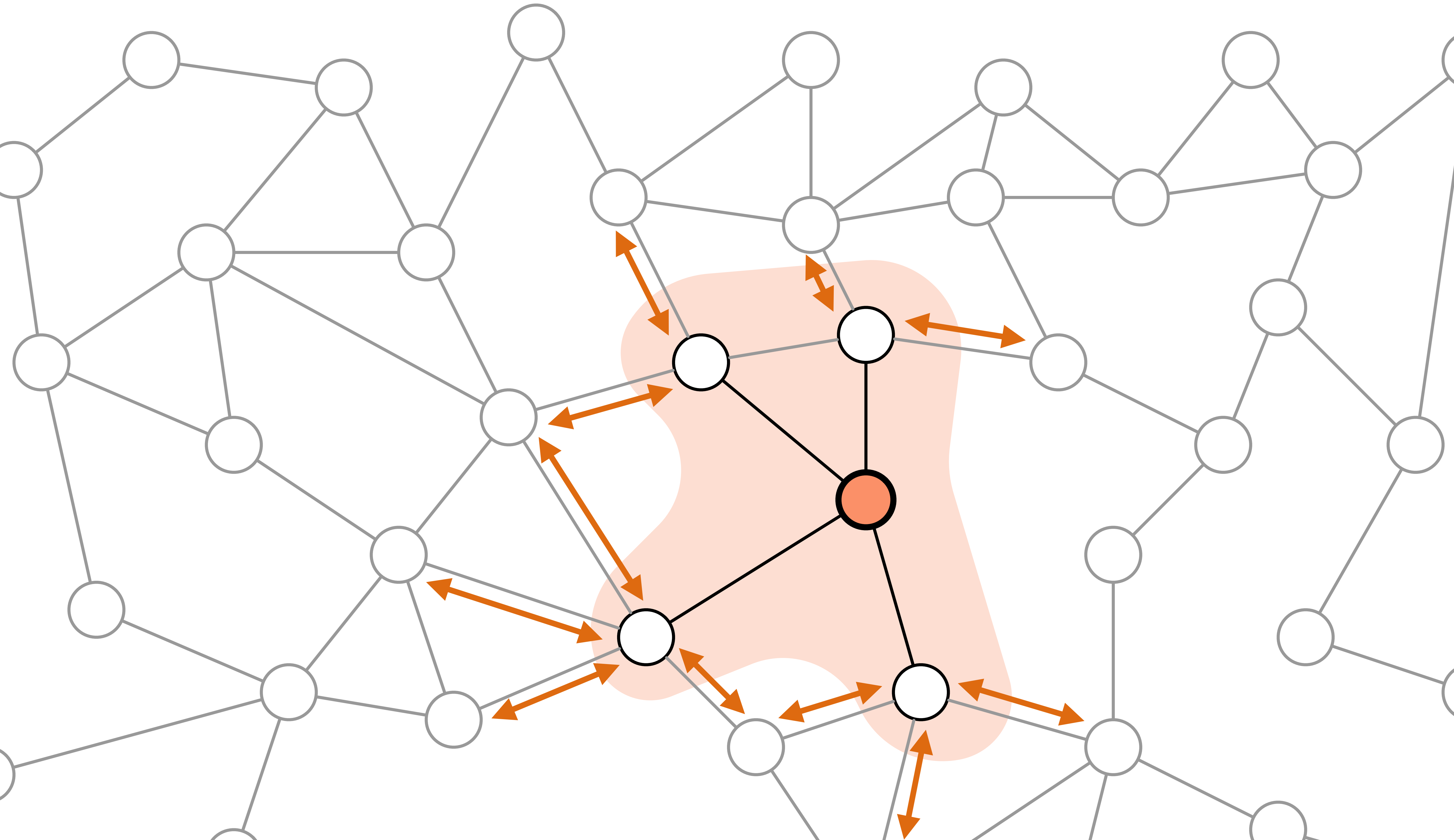
# Locality



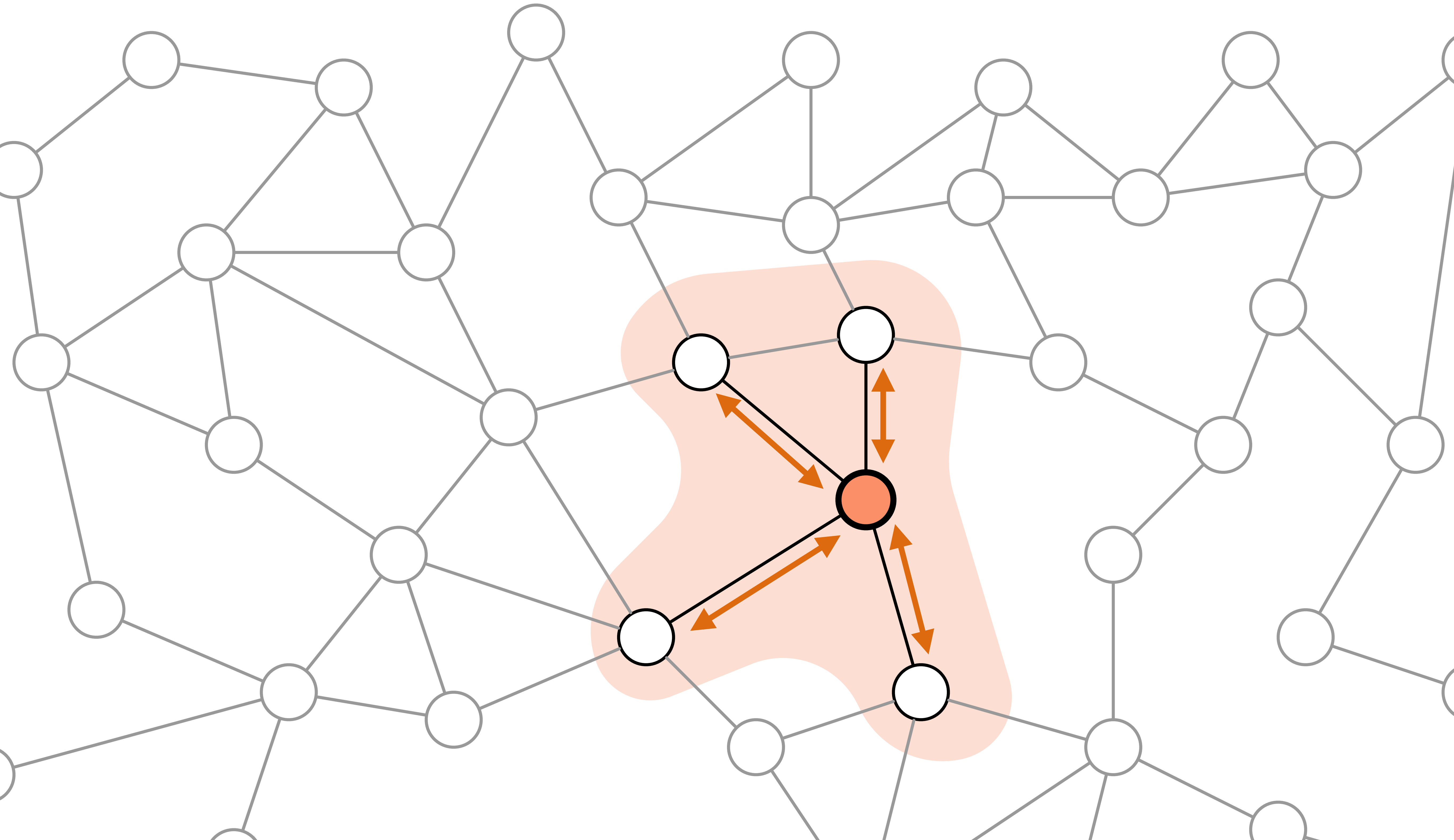
# Locality



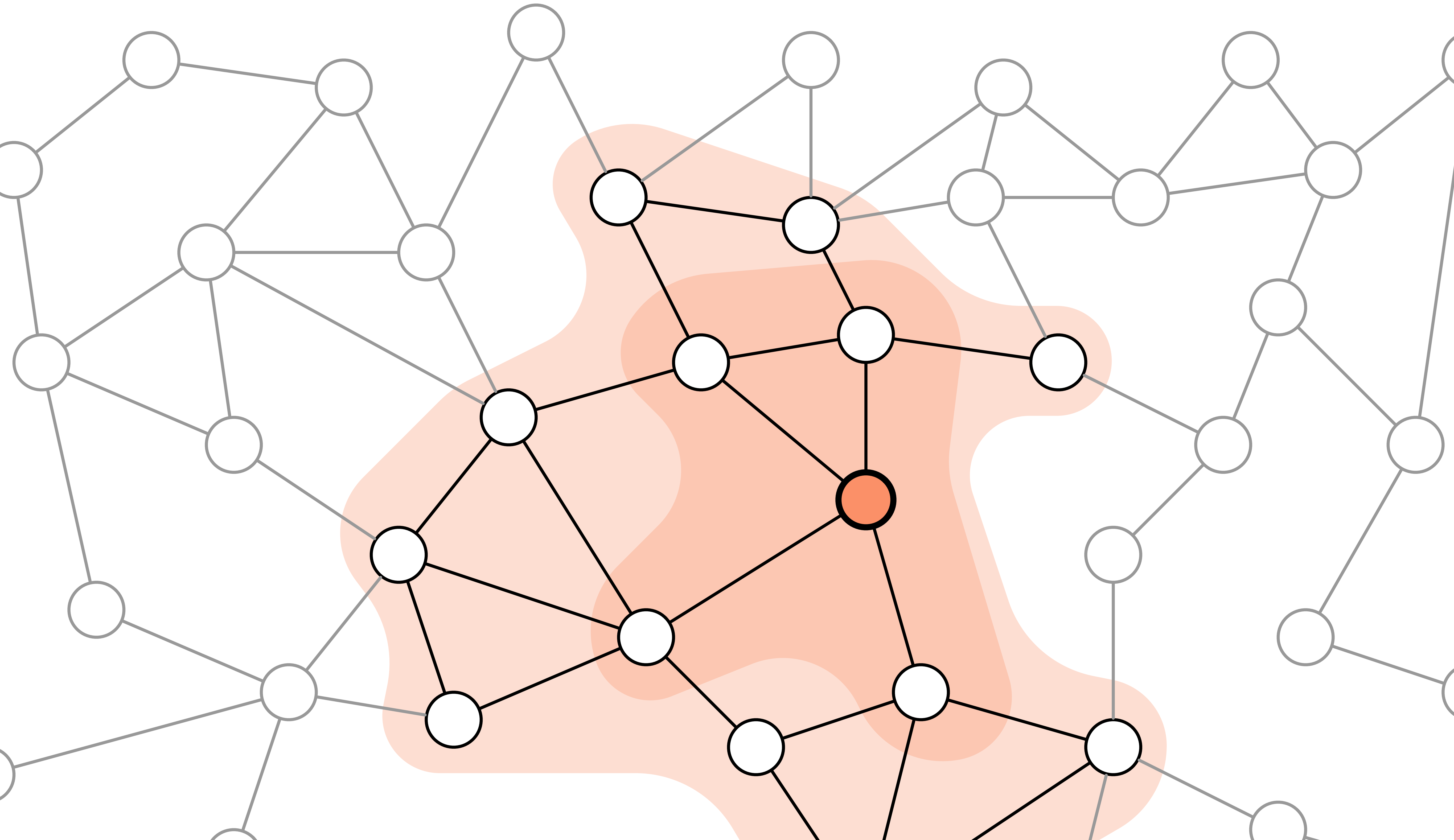
# Locality



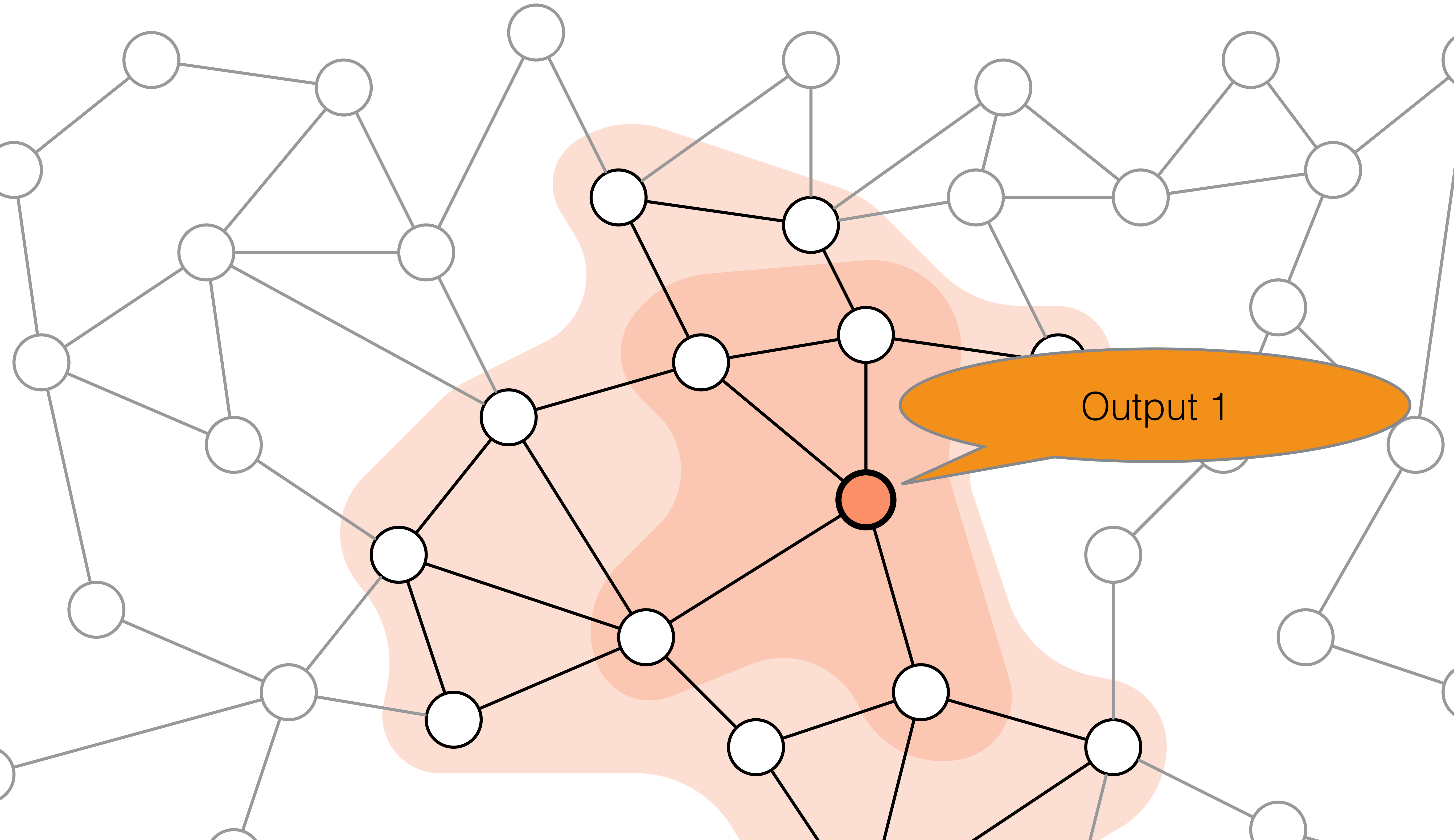
# Distributed computing



# Locality



# Locality

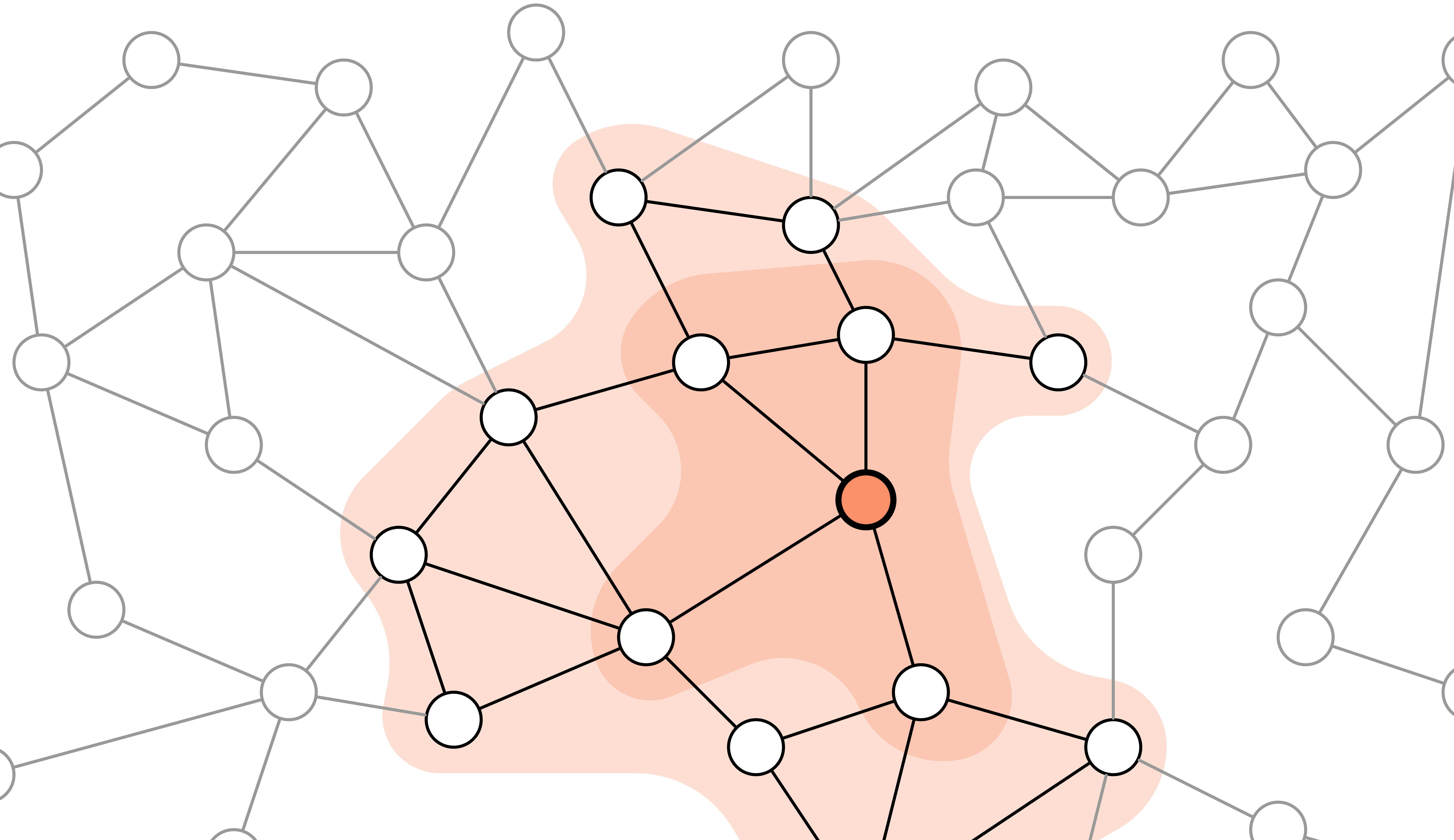


# Locality

- Everything proceeds in **synchronous communication rounds**
- Abstract away other possible challenges like **failures**, **asynchrony**, and **congestion**

In **t communication rounds** each node can learn **t-hop neighborhood**

# Locality



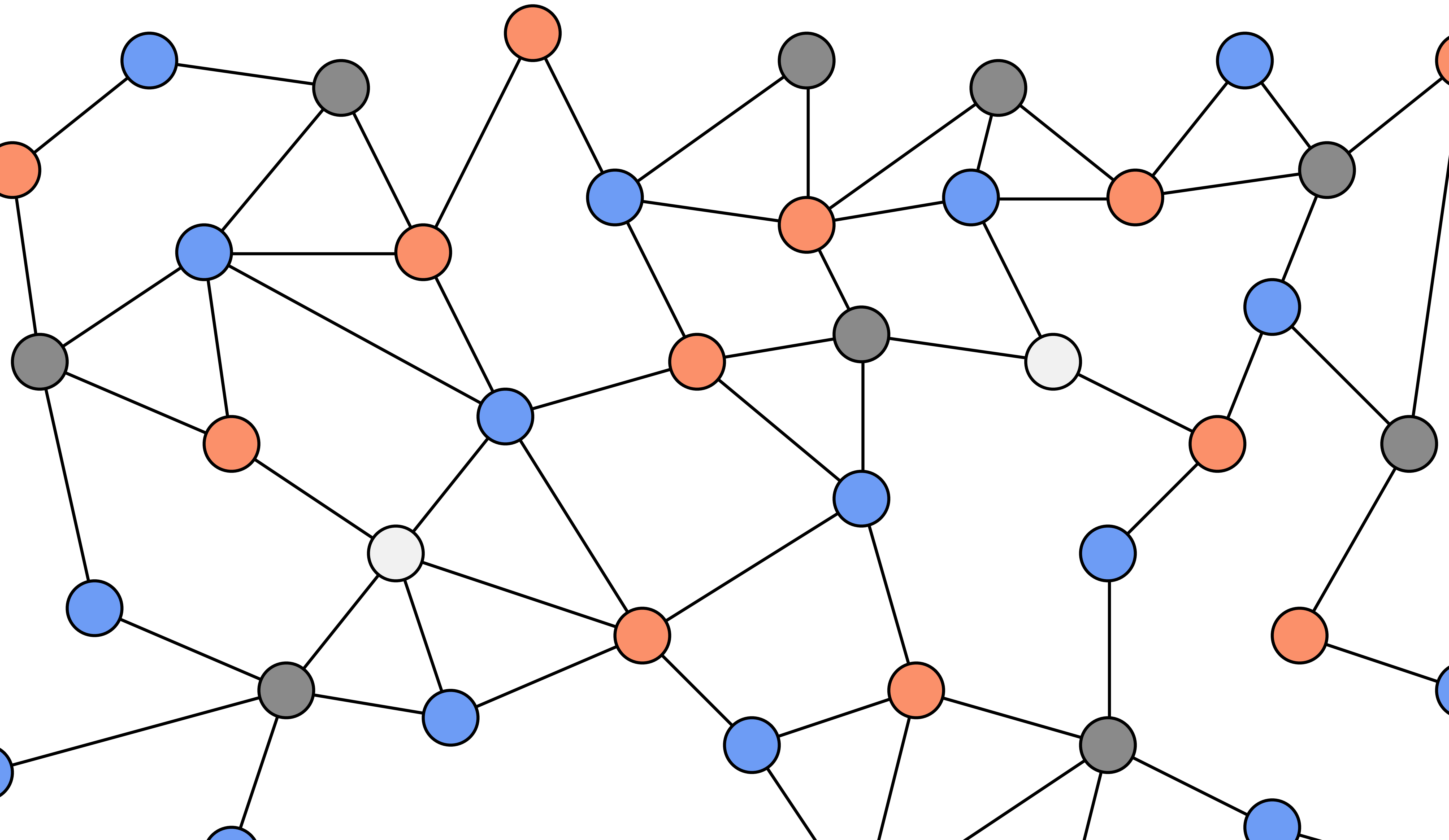
# Locality

*complexity* = **number of rounds** until all nodes have announced output

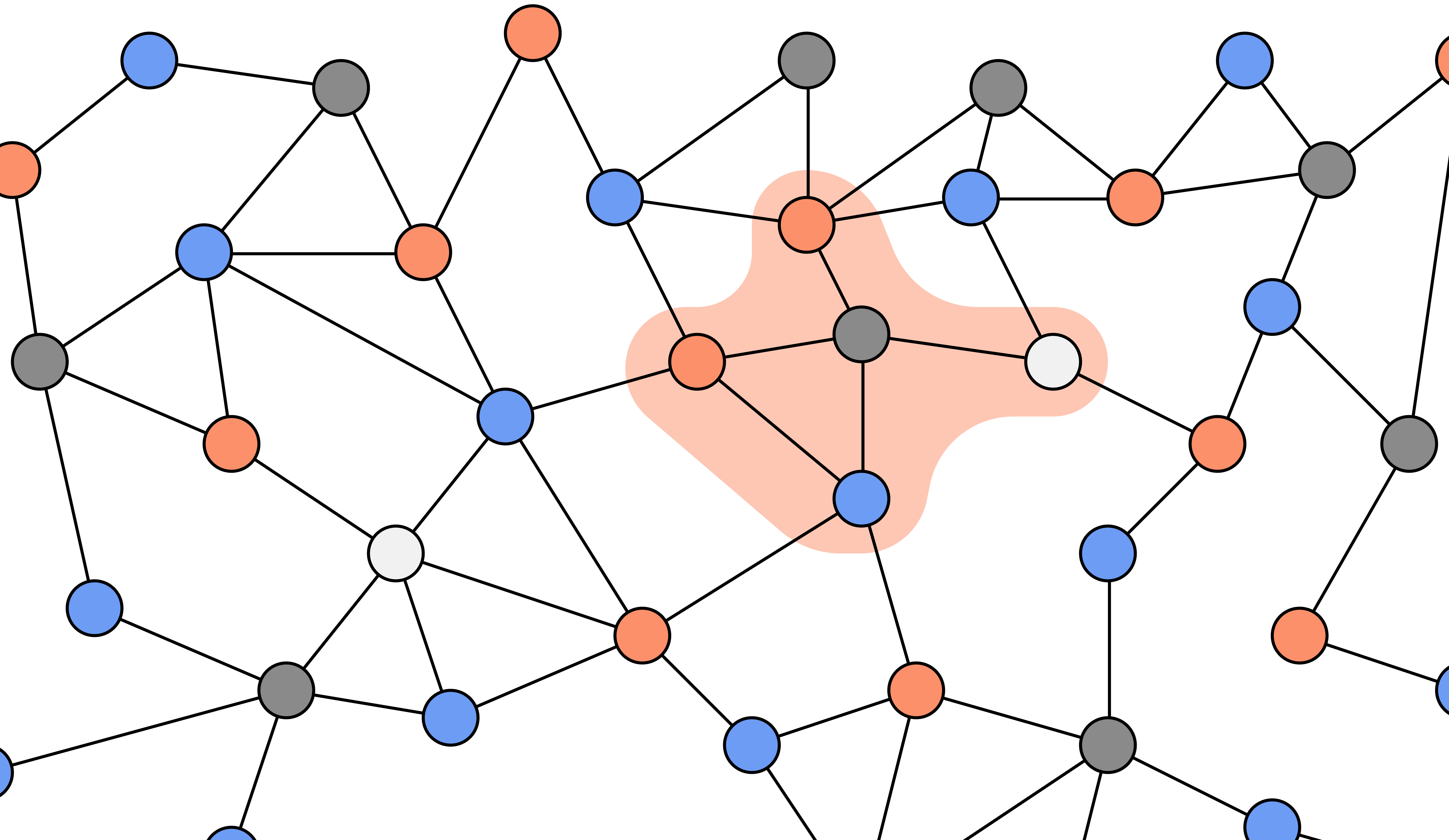
*output* = **each node** has to announce its own **local output** (e.g. its own *color*)

*algorithm* = mapping of **t-neighbourhoods** to **outputs** (*topology* → *color*)

# Locally checkable labelings



# Locally checkable labelings



# Details

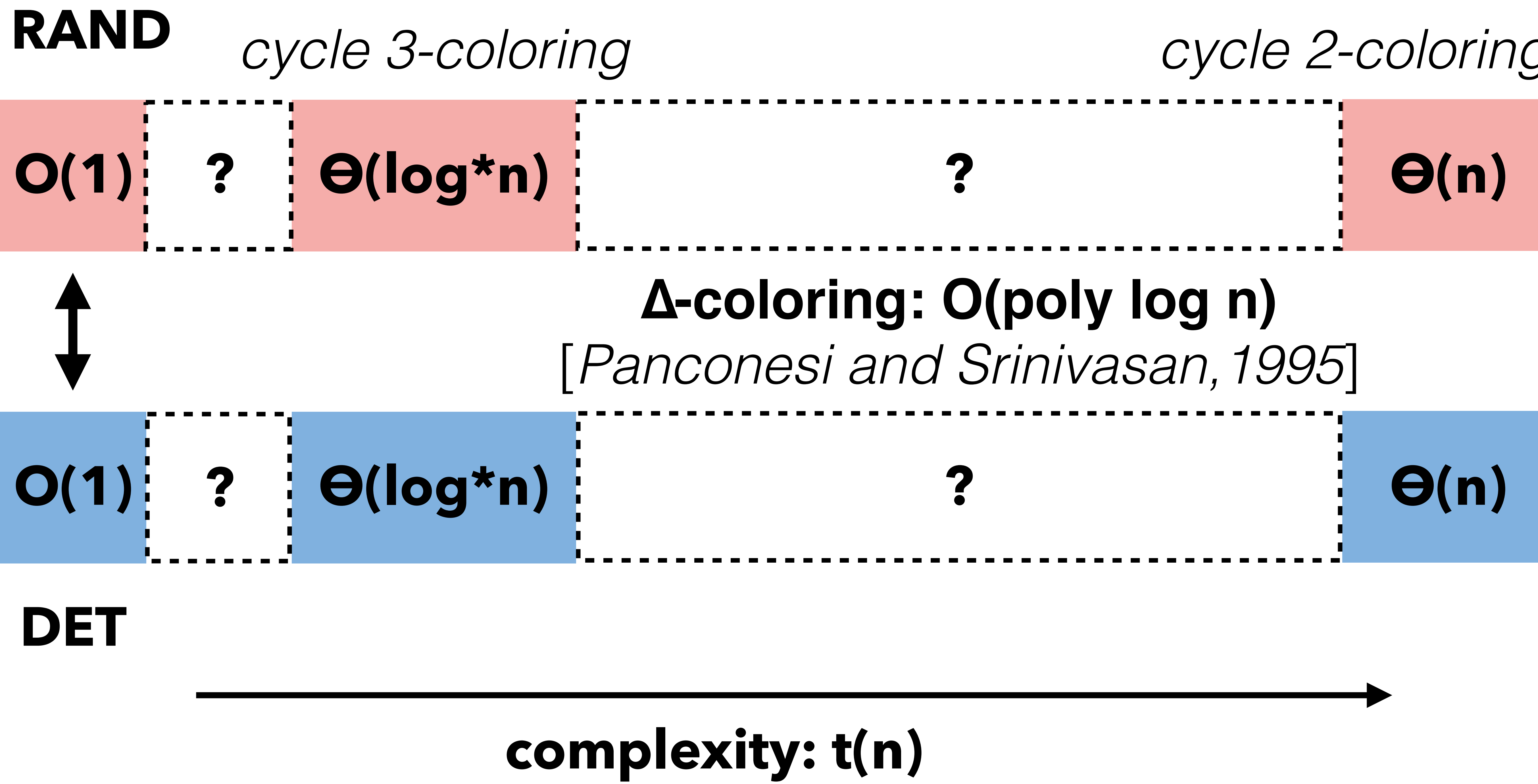
*Assuming standard LOCAL model*

- **LCLs** assume *bounded maximum degree* **Δ = O(1)**
- Every node has a unique name in **poly(n)** (*except when they don't!*)
- Value of **n** is *known* to the nodes (*strong models imply stronger lower bounds*)

# LCL complexity zoo

# LCL complexity zoo

*State of the art circa 2015*

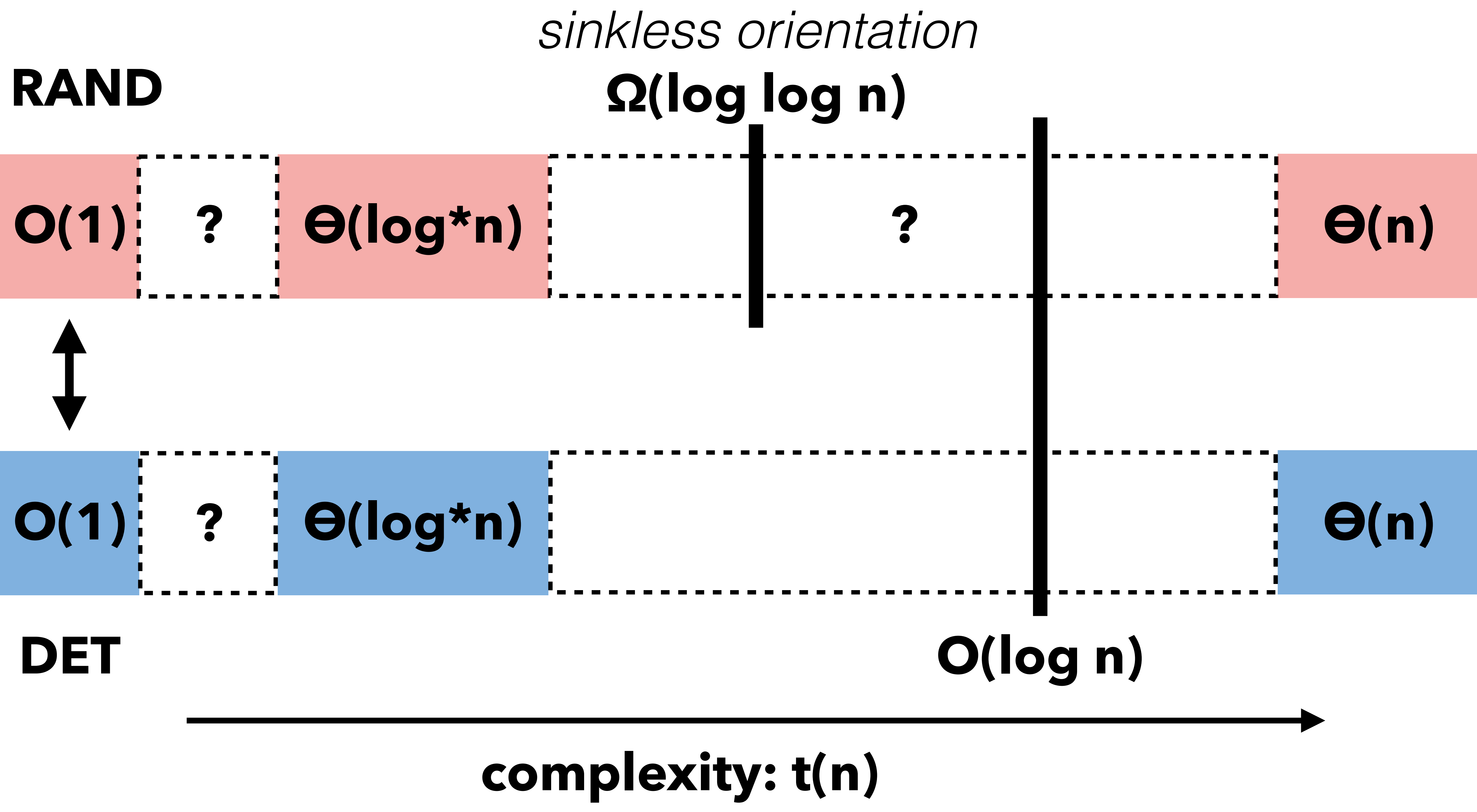


# Intermediate problems

*Sinkless orientation* requires $\mathbf{\Omega(\log_\Delta \log n)}$ randomized time

[*Brandt et al., STOC 2016*]

# LCL complexity zoo



# Implications

- **Distributed Lovász local lemma** at least as hard as *sinkless orientation*
- **Δ-coloring** at least as hard as *sinkless orientation*

# Lower bound:
# sinkless orientation

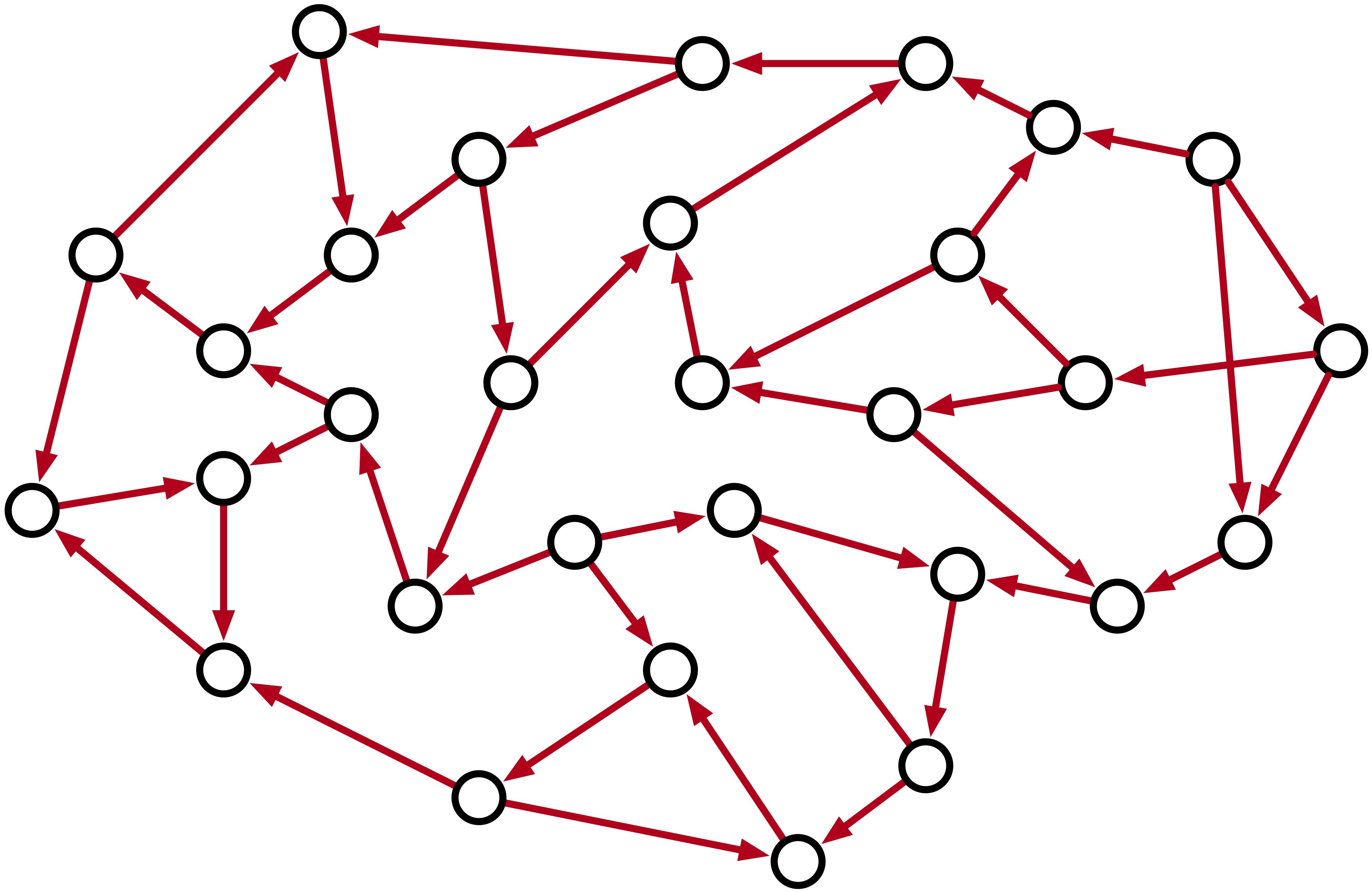
# Based on*

*A lower bound for the distributed Lovász local lemma,* Brandt, Fischer, Hirvonen, Keller, Lempiäinen, Rybicki, Suomela, and Uitto, STOC 2016

*An exponential separation between randomized and deterministic complexity in the LOCAL model* Chang, Kopelowitz, and Pettie, FOCS 2016

*The Complexity of Distributed Edge Coloring with Small Palettes,* Chang, He, Li, Pettie, and Uitto, SODA 2018

# Sinkless orientation



All edges are oriented with no sinks

# The lower bound

*Sinkless orientation* requires $\mathbf{\Omega(\log_\Delta \log n)}$ randomized time

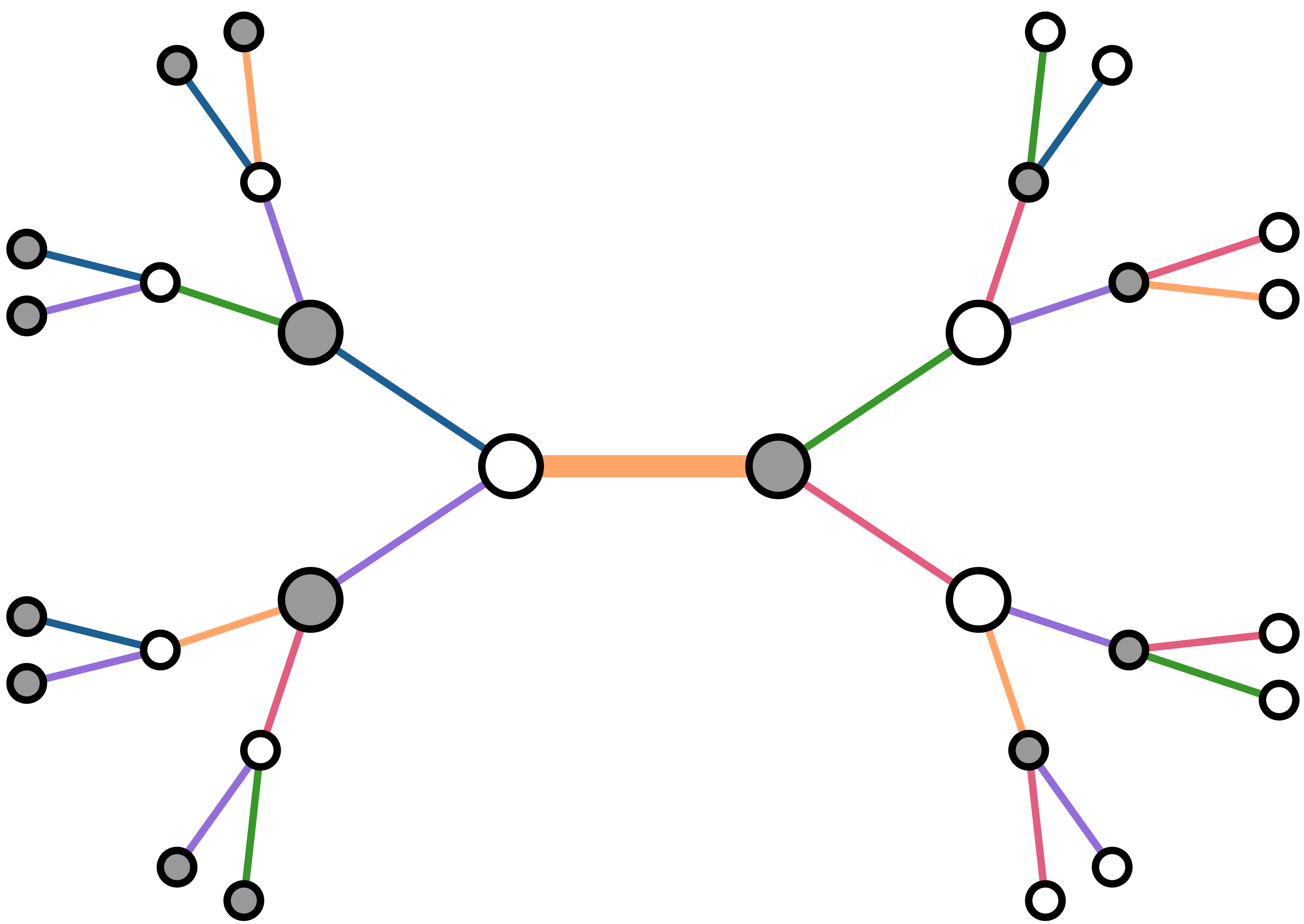
*Sinkless orientation* requires $\mathbf{\Omega(\log_\Delta n)}$ deterministic time

# A (simple) deterministic lower bound

We will start by proving a lower bound for a *simpler, deterministic* model:

Finding a **sinkless orientation** requires $\boldsymbol{\Omega(\log_\Delta n)}$ communication rounds in this model

# Lower bound: sinkless orientation



**(simple) model:**
**d-regular** graphs,
**2-vertex** col.
**c-edge** col.
(for **c >> d**)

graphs have large
(logarithmic) girth

# (Very) high level proof

1. In high-girth graphs a **$o(\log_\Delta n)$**-round algorithm for *sinkless orientation* implies a **0-round** algorithm for *sinkless orientation*
2. There is no **0-round** algorithm for *sinkless orientation* in high-girth graphs

# Lower bound: sinkless orientation

For **algorithm A**, define **running time profile**

$\mathbf{t} = (t_1, t_2, \ldots, t_c)$

**=**

Edges of color **i** must halt after $\mathbf{t_i}$ rounds*

# Lower bound: sinkless orientation

Assume algorithm has running time profile
**t** = (t,t,…,t)

**=**

Edges of **all colors** halt in *t* communication rounds

# Lower bound: sinkless orientation

For example, assume **d=3** and **c=5**

$\mathbf{t} = (t,t,t,t,t)$

*speed up color 5 by simulation*

$\mathbf{t}^{(1)} = (t,t,t,t,t-1)$

*speed up color 4 by simulation*

$\mathbf{t}^{(2)} = (t,t,t,t-1,t-1)$

# Lower bound: sinkless orientation

**t** = (t,t,t,t,t)

↓ *speed up each color*

**t-1** = (t-1,t-1,t-1,t-1,t-1)

↓ *repeat **t** times*

**0** = (0,0,0,0,0)

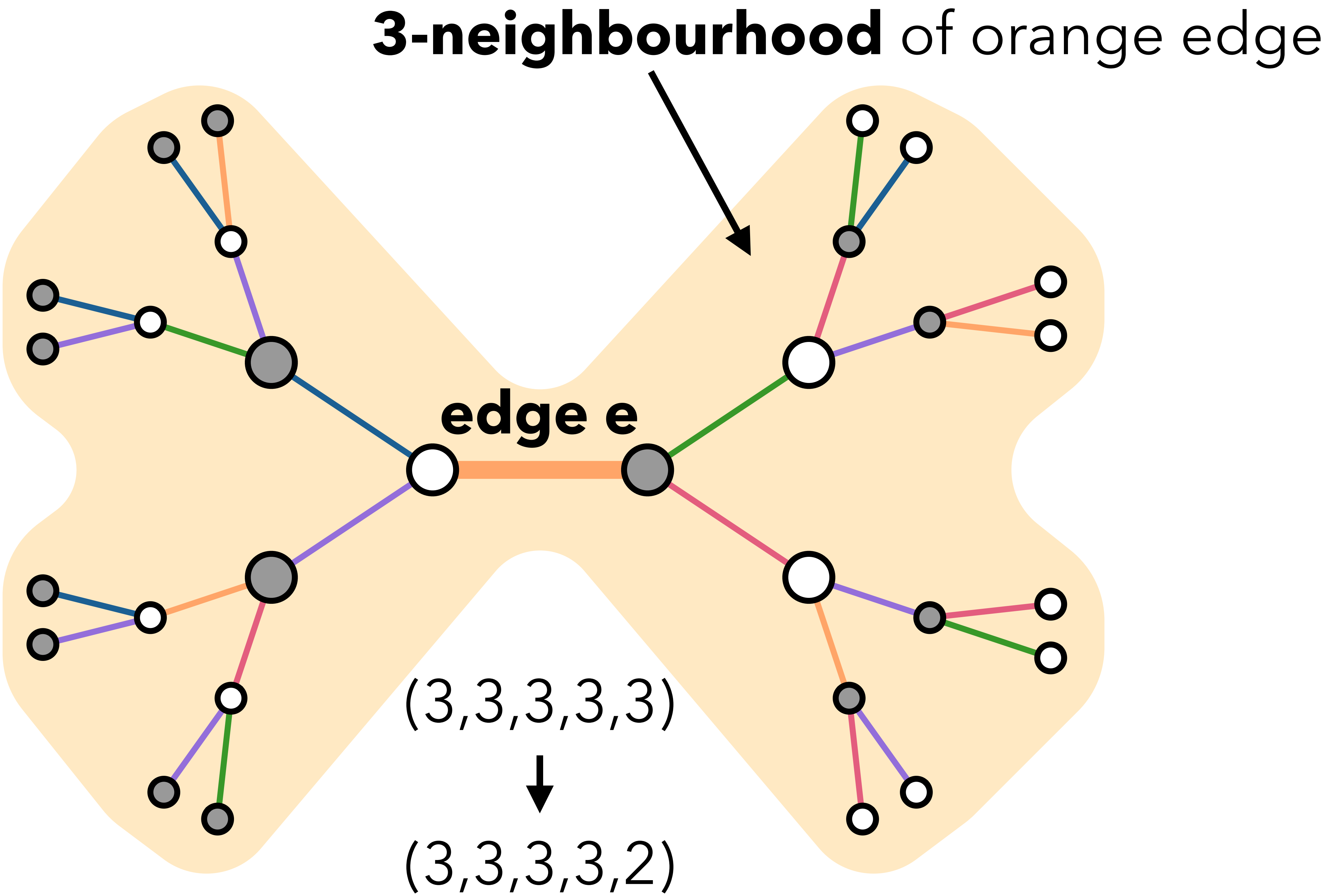
# Lower bound: sinkless orientation

algorithm with running time profile
$\mathbf{0} = (0,0,0,0,0)$

easy to show that this is impossible!

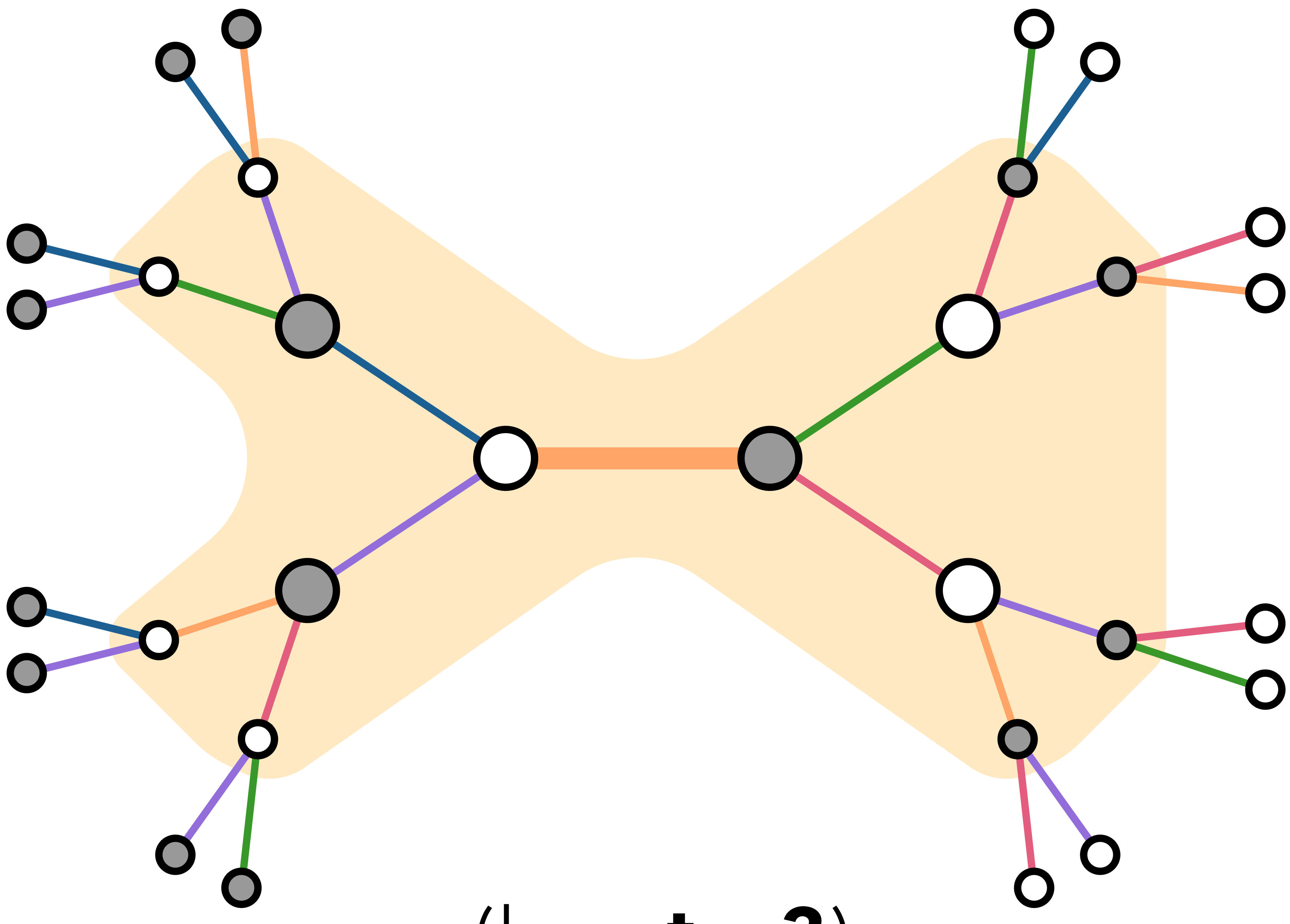
We can apply argument if initial $\mathbf{t = o(\log_\Delta n)}$

# Simulation



# Simulation

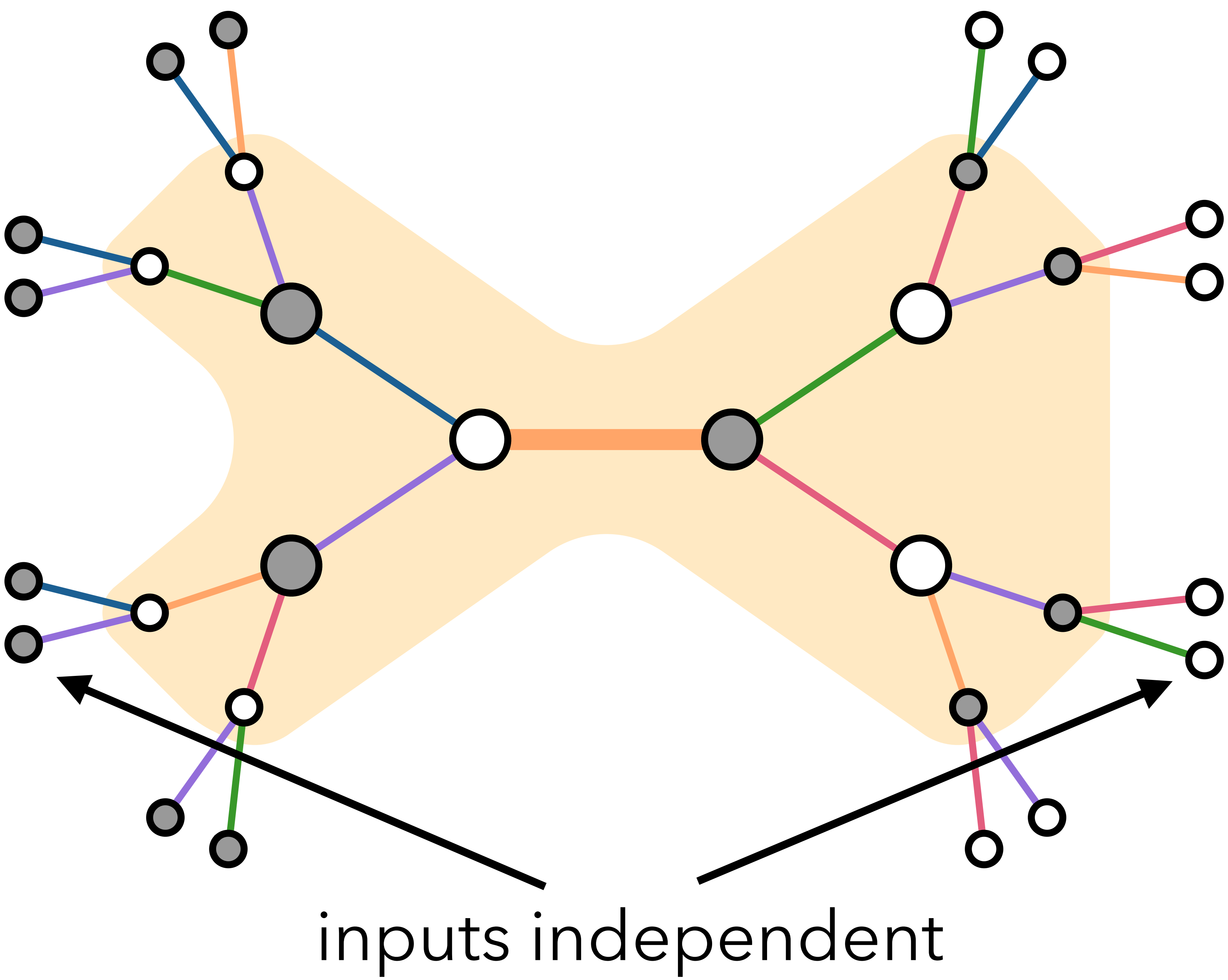
possible outputs given **2-neighbourhood**?



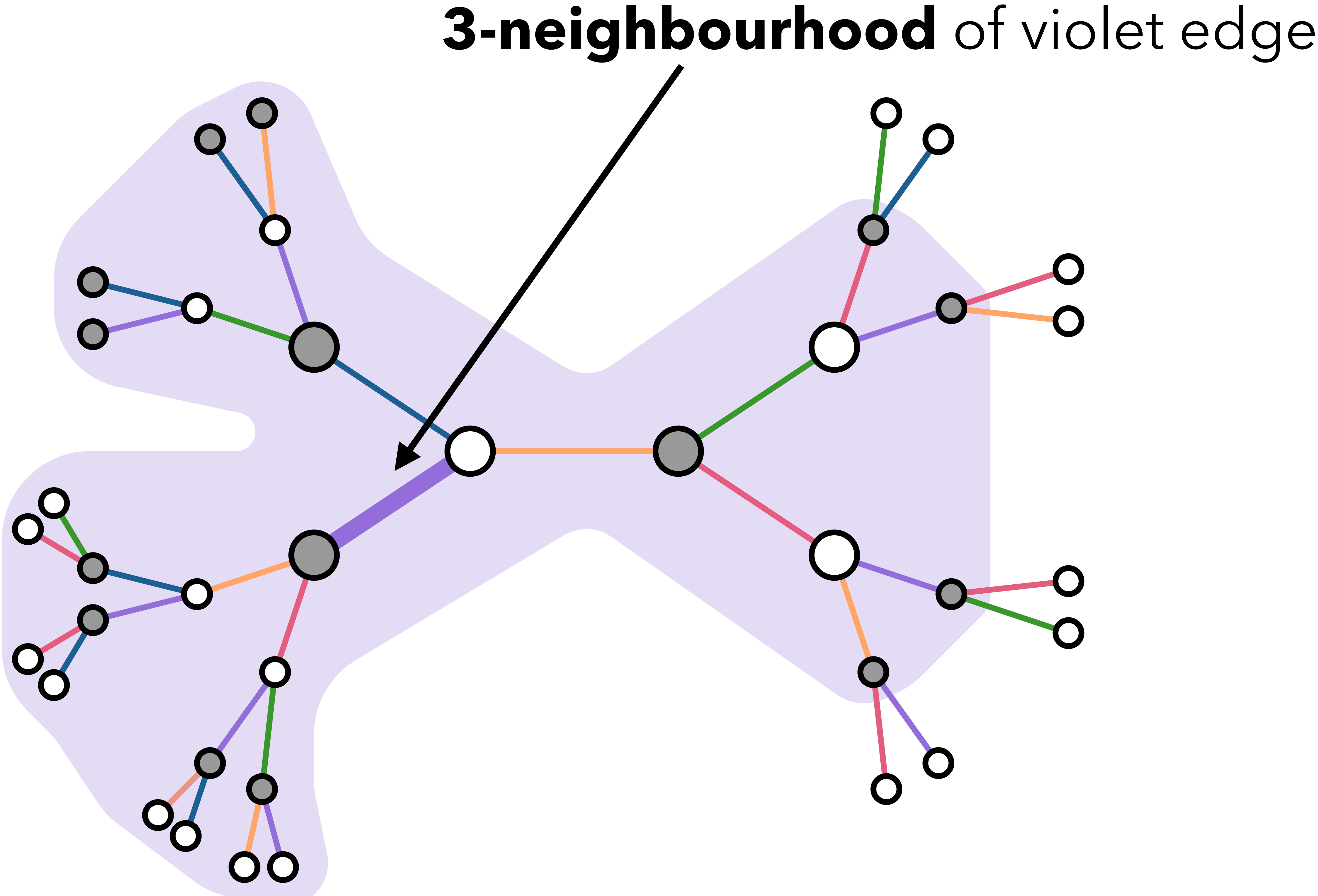
(here **t = 3**)

# Simulation

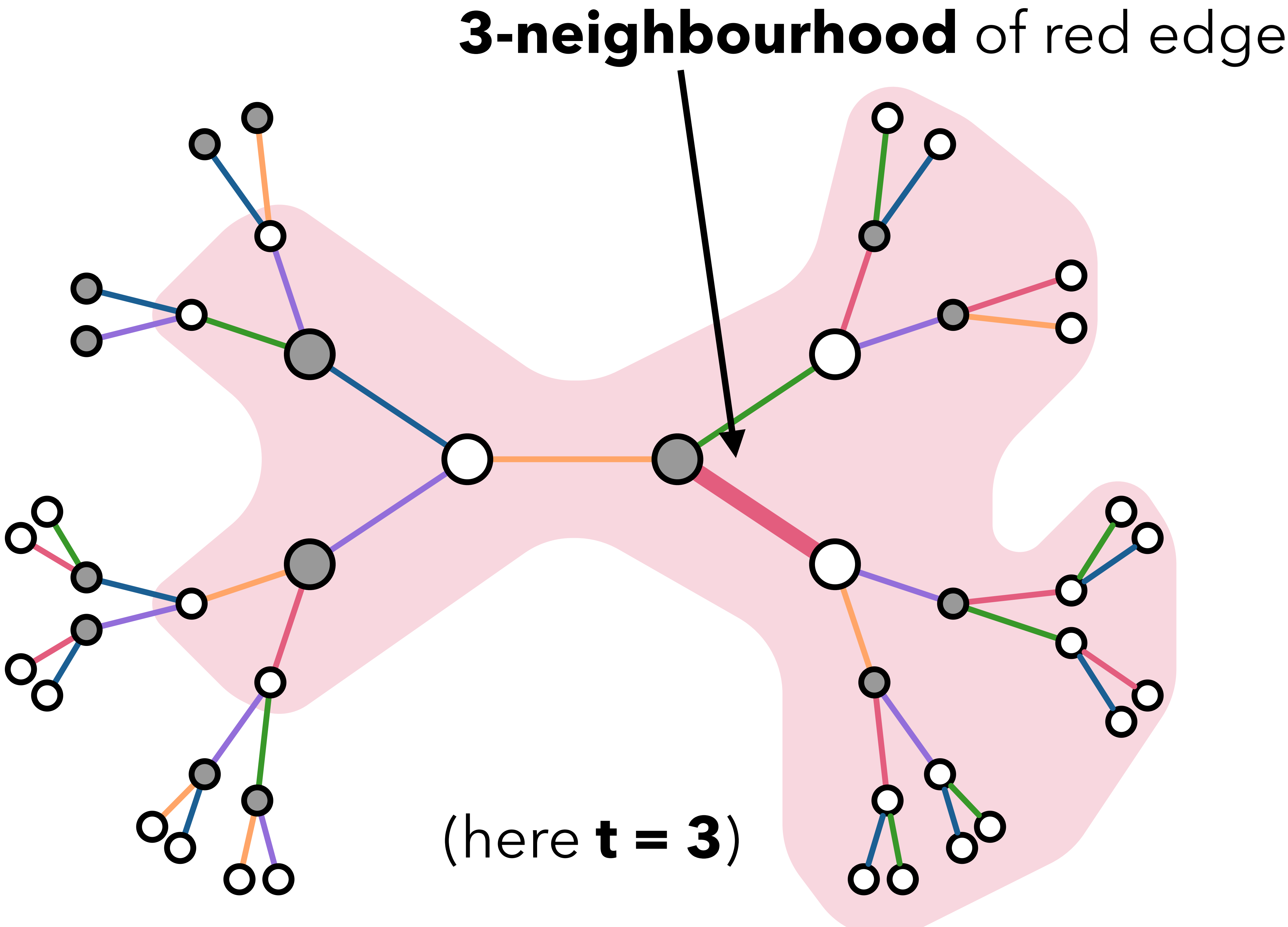
possible outputs given **2-neighbourhood**?



# Outputs of incident edges

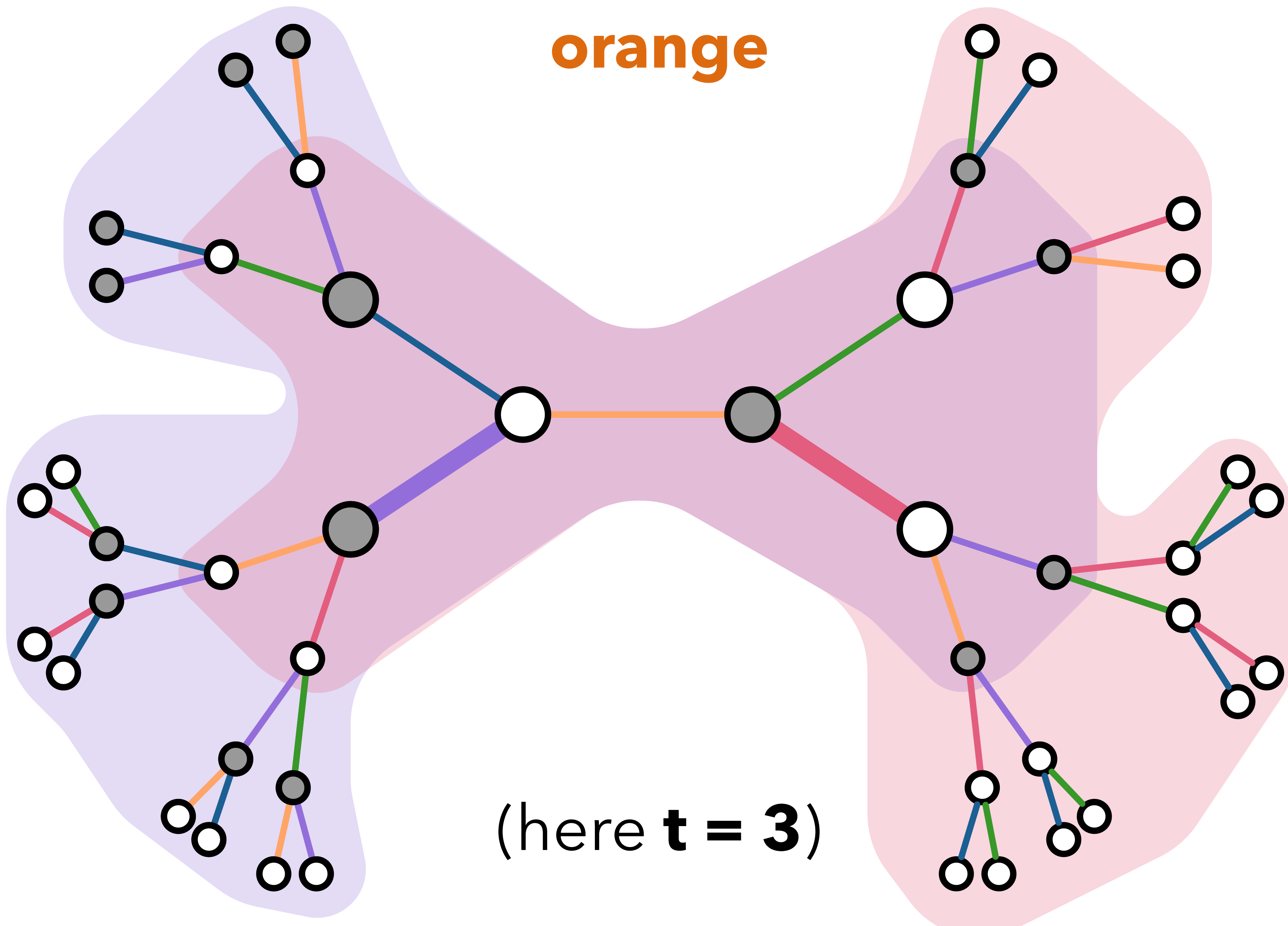


# Outputs of incident edges



# Outputs of incident edges

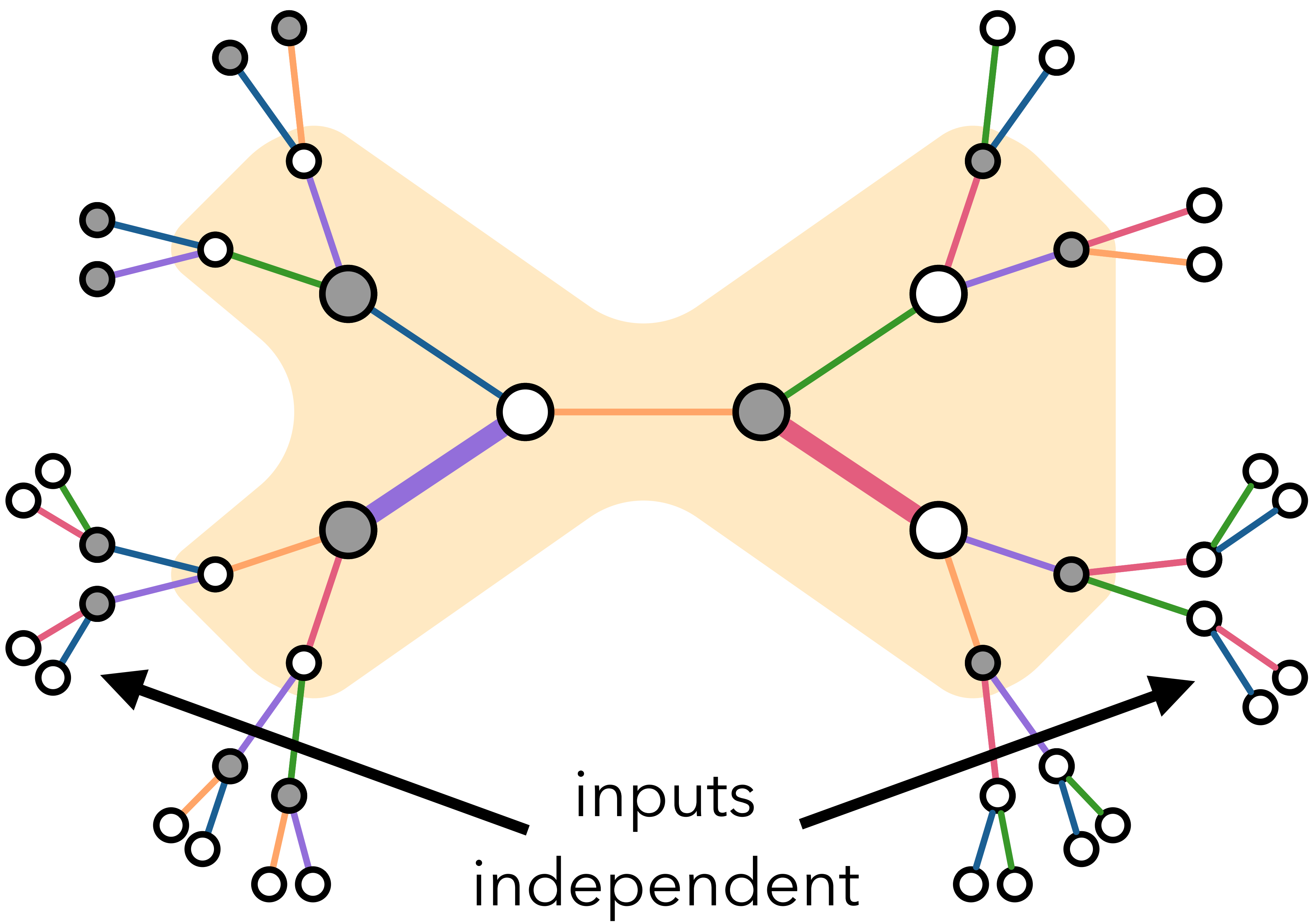
intersection of **3-neighbourhoods = 2-neighbourhood of orange**



(here **t = 3**)

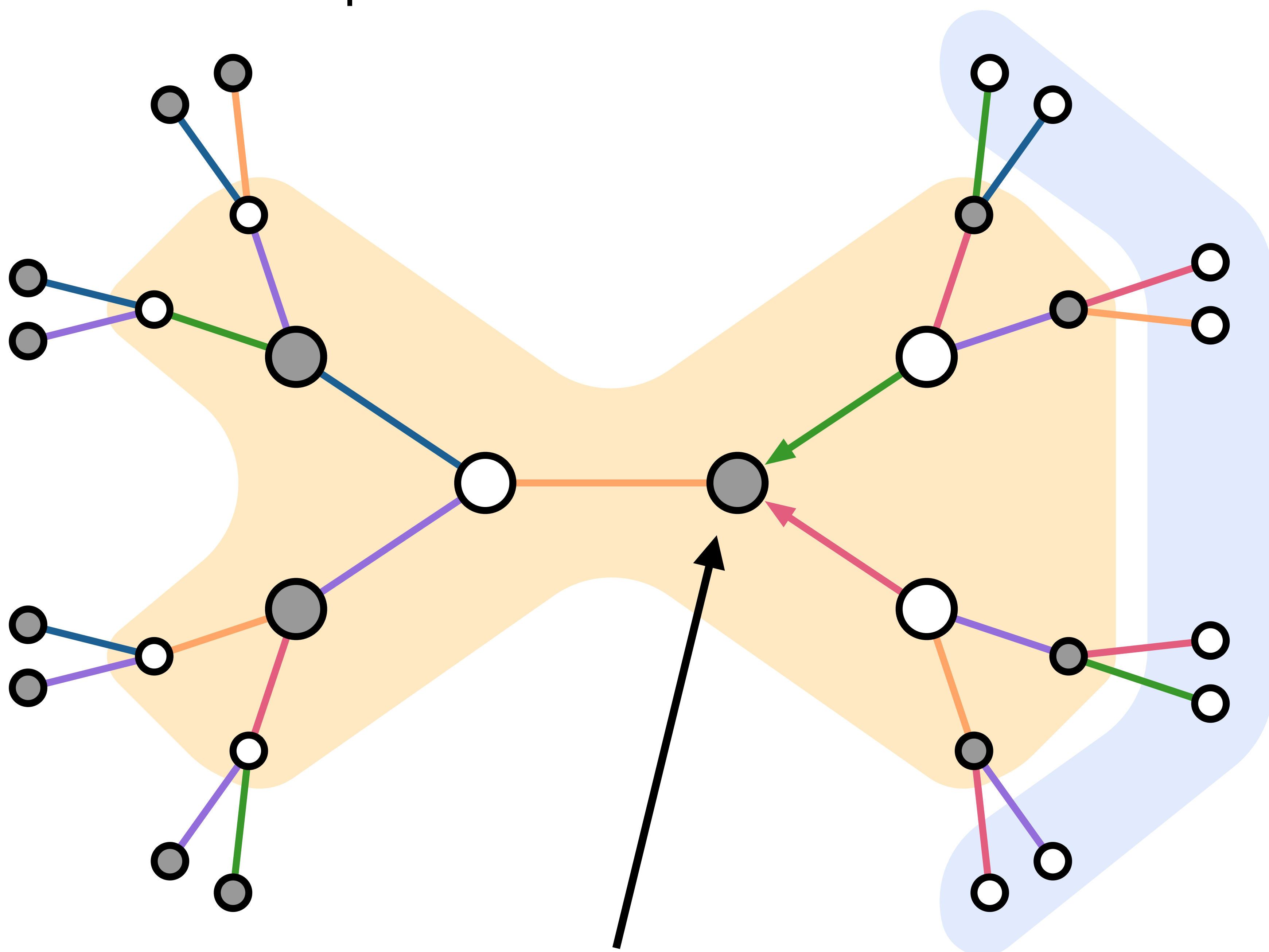
# Outputs of incident edges

outputs on the two sides are **independent** given **orange**



# Outputs of incident edges

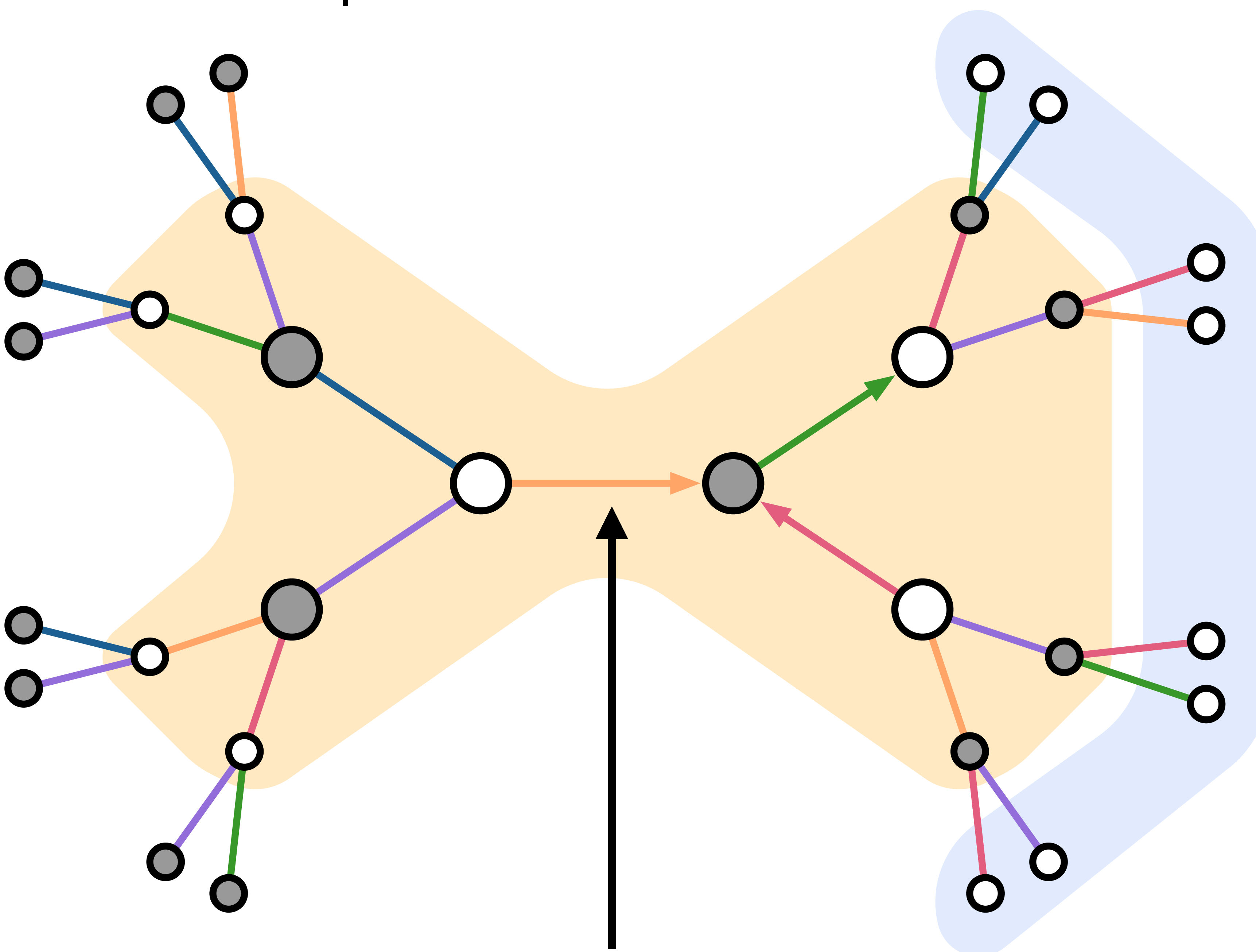
is it possible for endpoint to be a **sink** for the other edges?



∃ input s.t. other edges pointed towards node?

# Outputs of incident edges

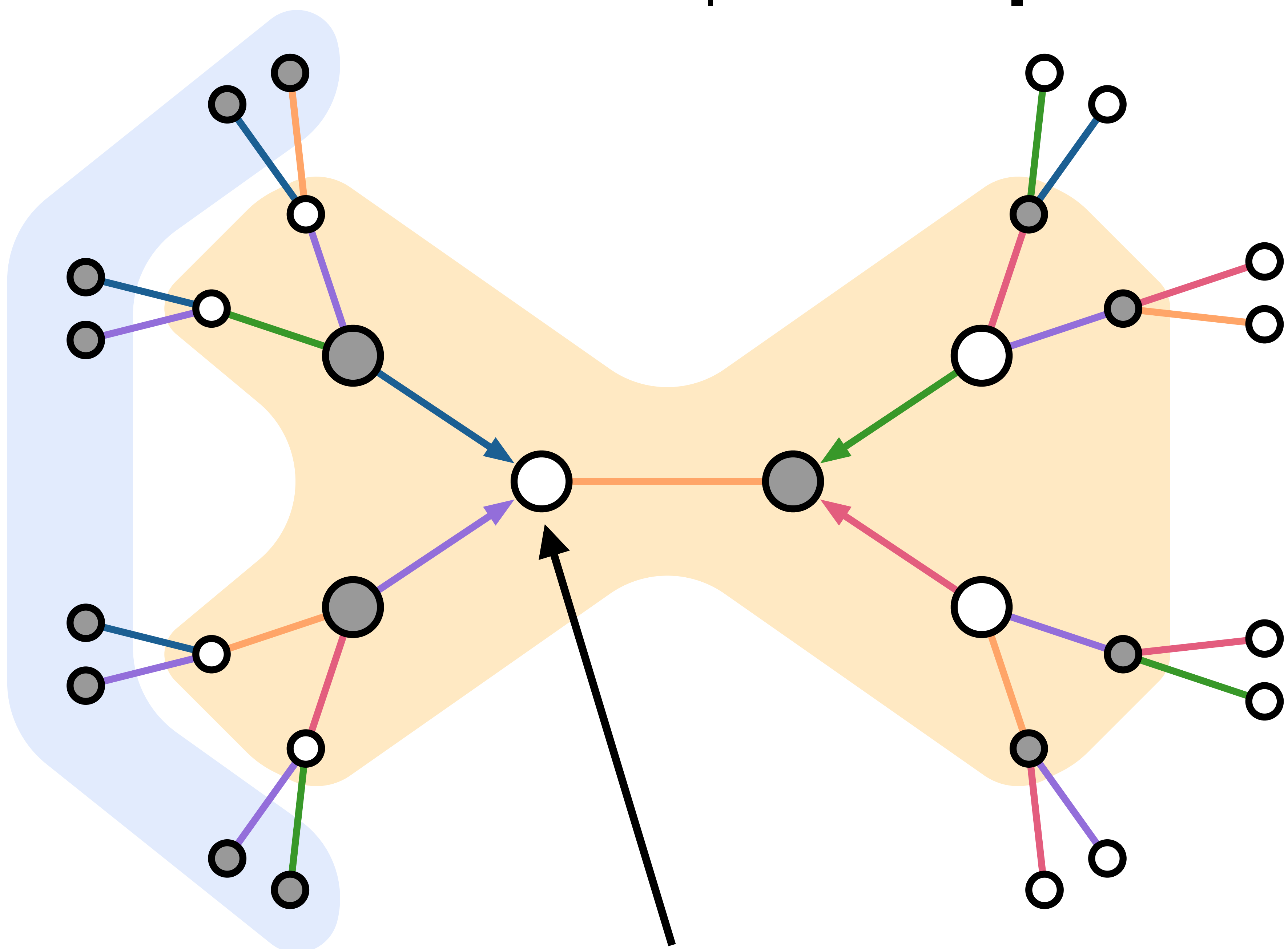
is it possible for endpoint to be a **sink** for the other edges?



if not, we can safely orient towards node

# Other endpoint a sink

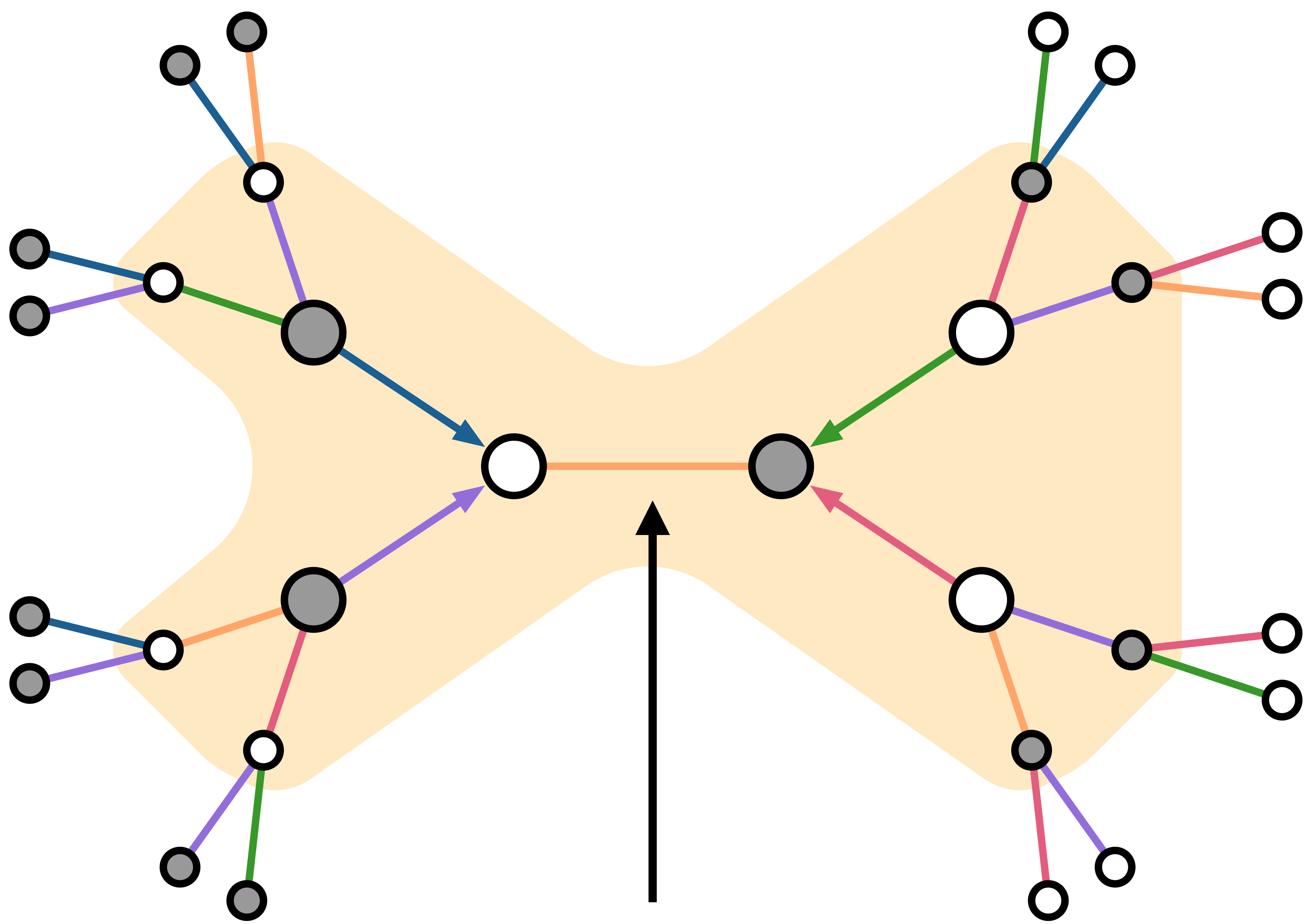
now assume the first endpoint is a **potential sink**



∃ input s.t. other edges pointed towards node?

# Other endpoint a sink

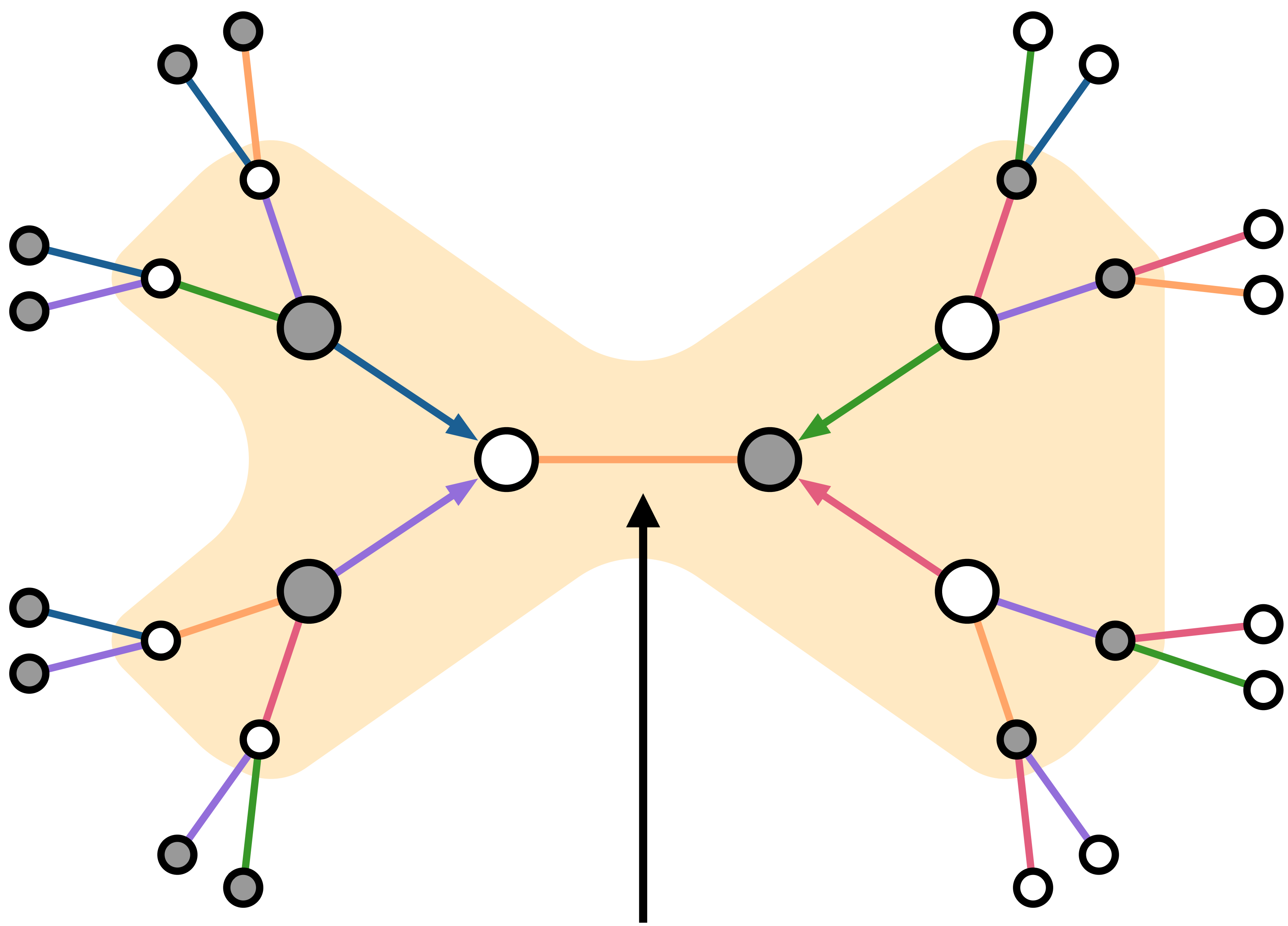
now assume both endpoints potential sinks



if yes, can engineer input such that this happens
(independence)

# Other endpoint a sink

now assume both endpoints potential sinks



no feasible output left for middle edge

# Lower bound: sinkless orientation

For example, assume **d=3** and **c=5**

**t** = (t,t,t,t,t)

$\big\downarrow$ *speed up each color*

**t-1** = (t-1,t-1,t-1,t-1,t-1)

$\big\downarrow$ *repeat **t** times*

**0** = (0,0,0,0,0)

# Problem with LOCAL model

- **Unique identifiers** induce dependencies between possible inputs of distant nodes
- Argument that we can **force a sink** unless one endpoint is safe is no longer true

# Roundabout solution: randomize

- Now consider the *randomized* setting
- In addition to the colouring, nodes have access to u.a.r. real number
- Can get identifiers w.h.p.

**Theorem**: sinkless orientation requires $\mathbf{\Omega(\Delta^{-1}\log_\Delta \log n)}$ rounds

# Lower bound: updated strategy

**A** : $\mathbf{t} = (t,t,t,t,t)$ *error with prob. <* $\boldsymbol{p}$

*speed up color 5 by simulation*

**A′** : $\mathbf{t}^{(1)} = (t,t,t,t,t\text{-}1)$ *error with prob. <* $\boldsymbol{3p^{1/3}}$

*speed up color 4 by simulation*

**A′′** : $\mathbf{t}^{(2)} = (t,t,t,t\text{-}1,t\text{-}1)$

# Lower bound:
# updated strategy

$\mathbf{A_t}$ **:** $\mathbf{t}$ = (t,t,t,t,t) *error with prob. <* $\boldsymbol{p}$

*speed up each color*

$\mathbf{A_{t-1}}$ **:** $\mathbf{t\text{-}1}$ = (t-1,t-1,t-1,t-1,t-1) *error with prob. <* $\boldsymbol{O(p^{-3^{\wedge}(2d-1)})}$

*repeat* ***t*** *times*

$\mathbf{A_0}$ **:** $\mathbf{0}$ = (0,0,0,0,0) *error with prob. <* $\boldsymbol{O(p^{-3^{\wedge}(t(2d-1))})}$

# Lower bound: updated strategy

start with alg. **A**, running time **t**, error prob. $\mathbf{p_0}$

↓

algorithm **A'** with running time **0**, error prob. < $\mathbf{O(p^{-3\wedge(t(2d-1))})}$

**0 rounds:** must have error probability $\mathbf{p > 1/8^d}$

↓

$\mathbf{t = \Omega(\Delta^{-1} \log \log n)}$

# Lower bound: updated strategy

**A** : $\mathbf{t} = (t,t,t,t,t)$ *error with prob.* $< \boldsymbol{p}$

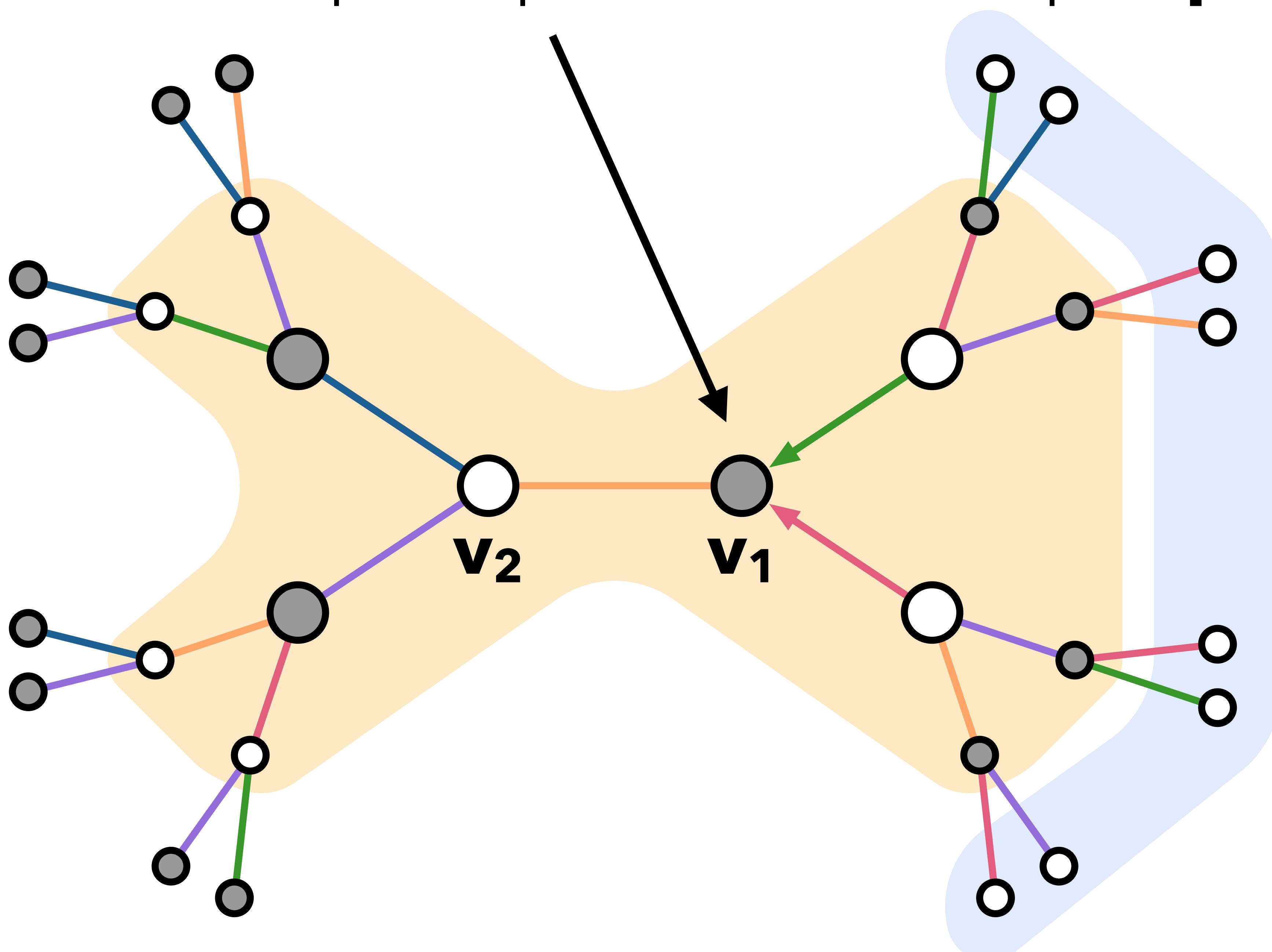
*speed up a color by simulation*

**A′** : $\mathbf{t}^{(1)} = (t,t,t,t,t-1)$ *error with prob.* $< \boldsymbol{3p^{1/3}}$

$\mathbf{t}^{(2)} = (t,t,t,t-1,t-1)$

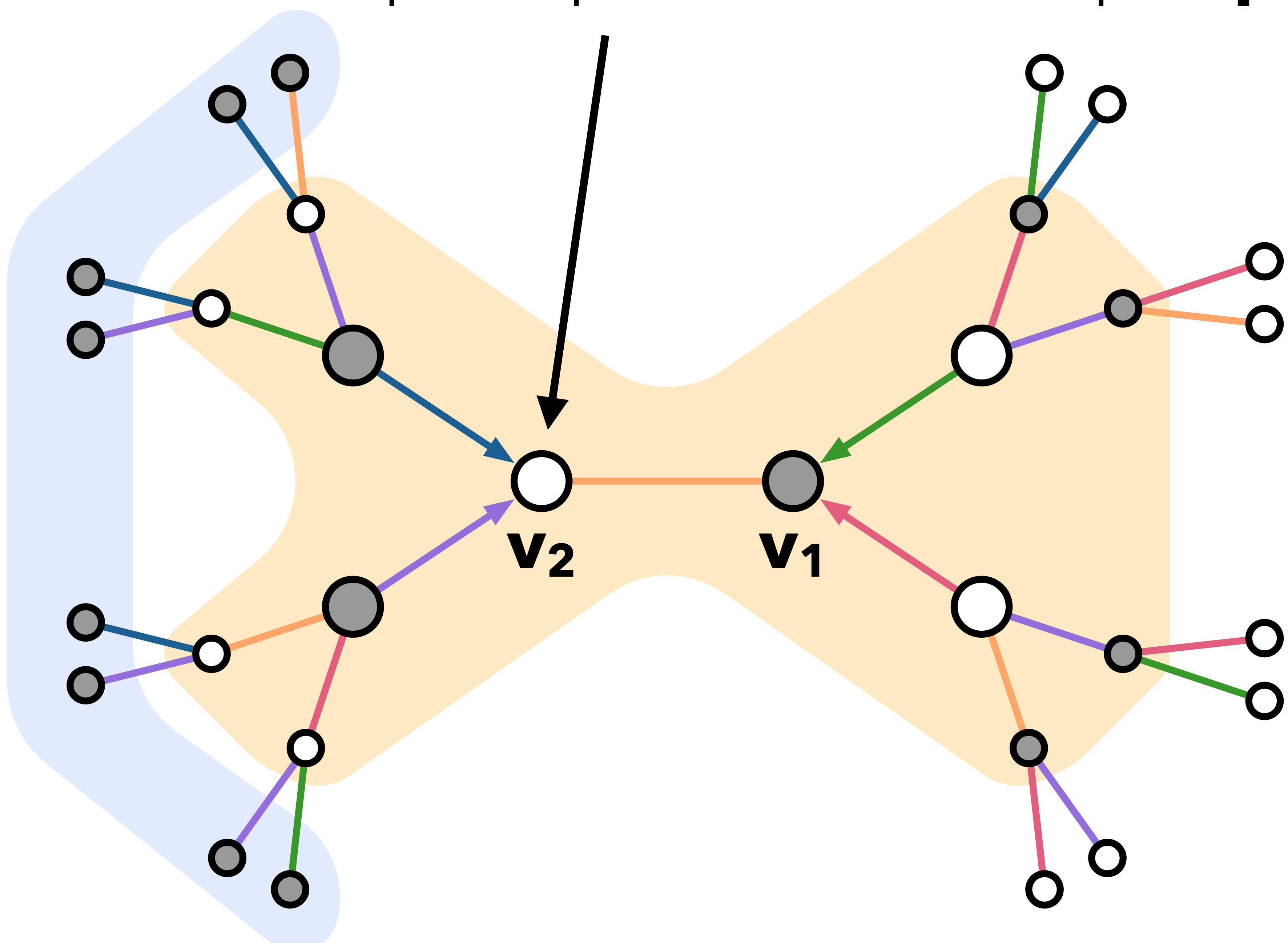
# Outputs of incident edges

black endpoint potential **sink** w.p. **> p**?



# Outputs of incident edges

white endpoint potential **sink** w.p. **> p**?



# Back to deterministic

**Theorem** (*Chang et al., FOCS 2016*): Assume that for LCL L there exists an algorithm with running time **$t = o(\log_\Delta n)$**, then there exists an algorithm with running time **$t' = O(\log^* n)$**

↓

**Corollary**: sinkless orientation requires **$\Omega(\log_\Delta n)$** deterministic time

# Automatic speed-up

- Another **black box simulation**
- A given **algorithm A** is ”fooled” to run faster: compute locally unique ”identifiers” (a colouring) and run **A** on those
- Efficient solving of LCLs reduces to **coloring + constant time**

# Back to randomized

**Theorem** (*Chang et al., FOCS 2016*): randomized complexity of an LCL on instances of **size n** is at least the deterministic complexity on instances of **size (log n)$^{1/2}$**

**Corollary:** sinkless orientation requires **$\Omega(\log_\Delta \log n)$** randomized time

# What just happened?

deterministic: $\boldsymbol{\Omega(\log_\Delta n)}$
*Proof technique doesn't work for identifiers*

*automatic connection*
randomized:
$\boldsymbol{\Omega(\log \log n)}$

*IDs → randomness*
randomized:
$\boldsymbol{\Omega(\Delta^{-1} \log \log n)}$

*automatic speed-up*
deterministic:
$\boldsymbol{\Omega(\log n)}$

# What just happened?

deterministic: $\mathbf{\Omega(\log_\Delta n)}$
*Proof technique doesn't work for identifiers*

*automatic connection*
randomized:
$\mathbf{\Omega(\log \log n)}$

*IDs → randomness*
randomized:
$\mathbf{\Omega(\Delta^{-1} \log \log n)}$

→

*automatic speed-up*
deterministic:
$\mathbf{\Omega(\log n)}$

# Automatic simulation speed-up

# Deterministic speed-up

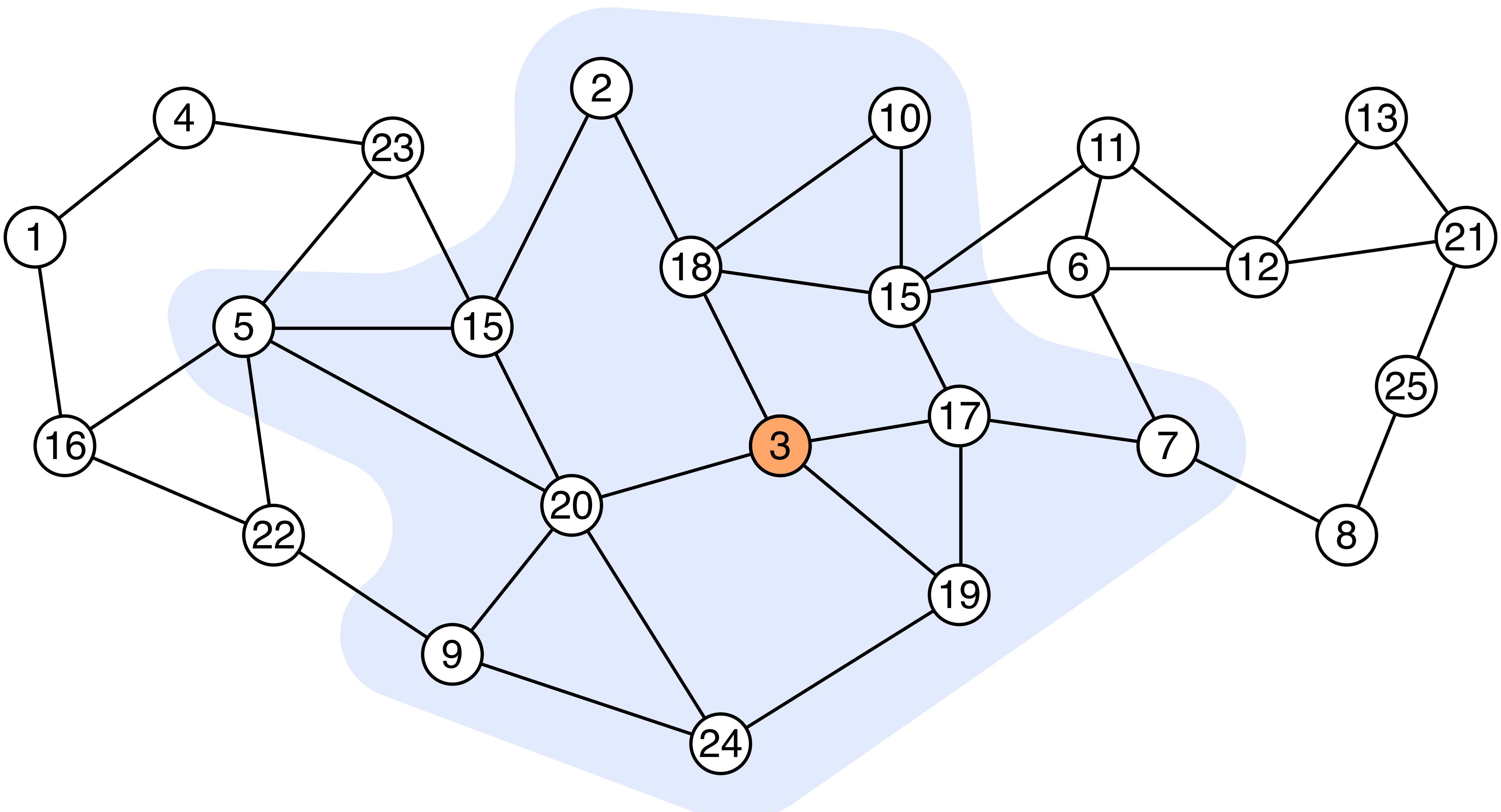
**Theorem** (*Chang et al., FOCS 2016*): Assume that for LCL L there exists an algorithm with running time $\mathbf{t = o(\log_\Delta n)}$, then there exists an algorithm with running time $\mathbf{t' = O(\log^* n)}$

# Algorithm’s view

- Assume **algorithm A** for LCL L with running time **t = ϴ(log log n)**
- Algorithm knows **n**, runs for **t(n)** rounds, stops
- What can the algorithm see?

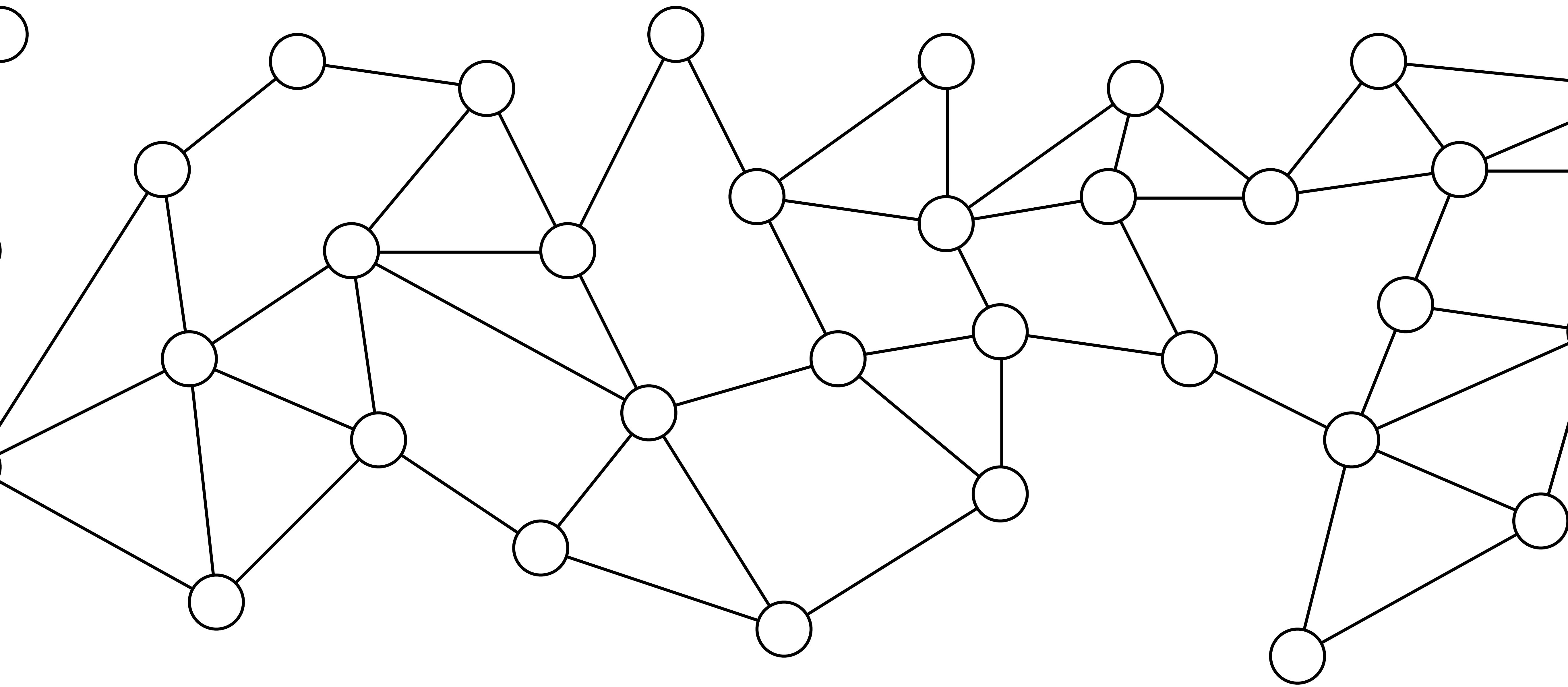
# Algorithm’s view

**Δ = 5:** For some **t(n)** assume **t(25) = 2**



# Algorithm's view

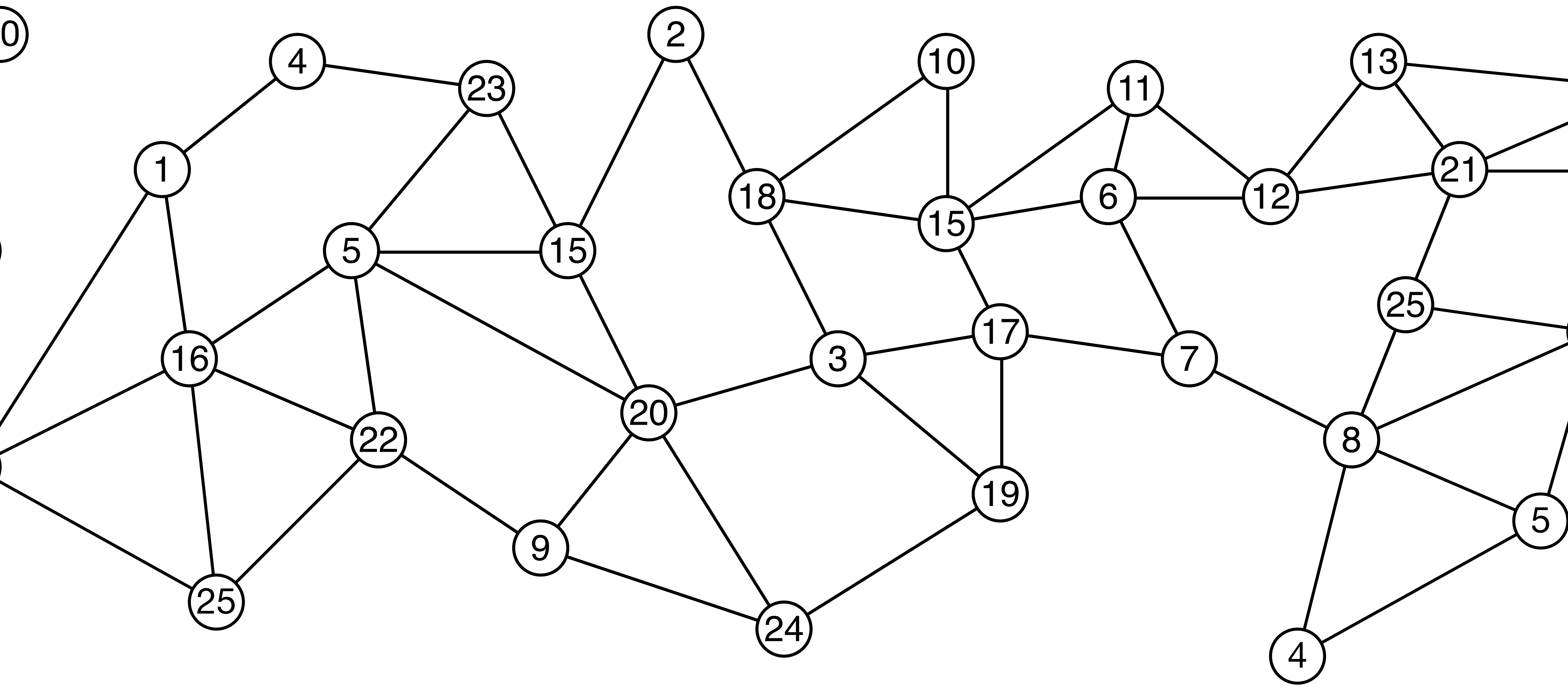
Now consider a **graph G** of size **n >> 25**



# Algorithm’s view

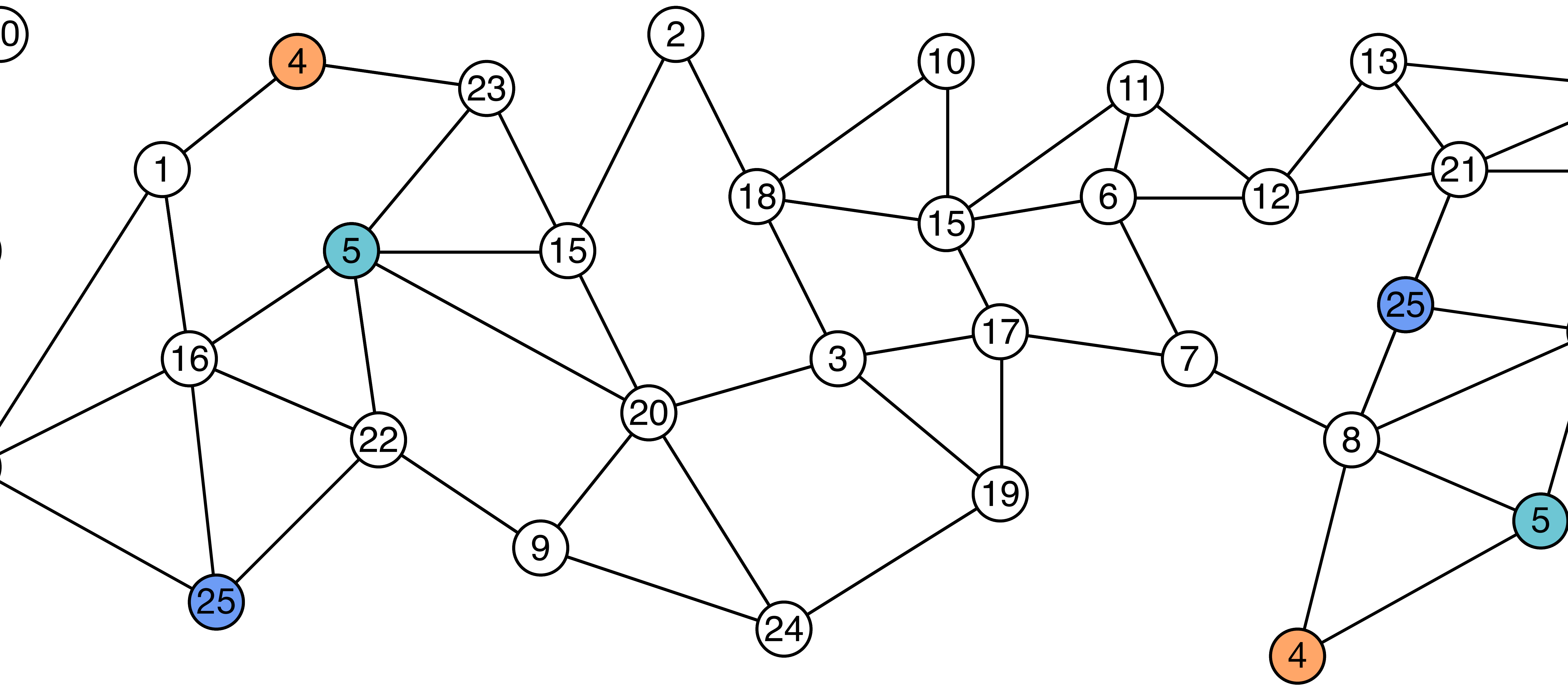
Label s.t. *every node sees every label appear only once* =

**distance O(1)-colouring**



# Algorithm’s view

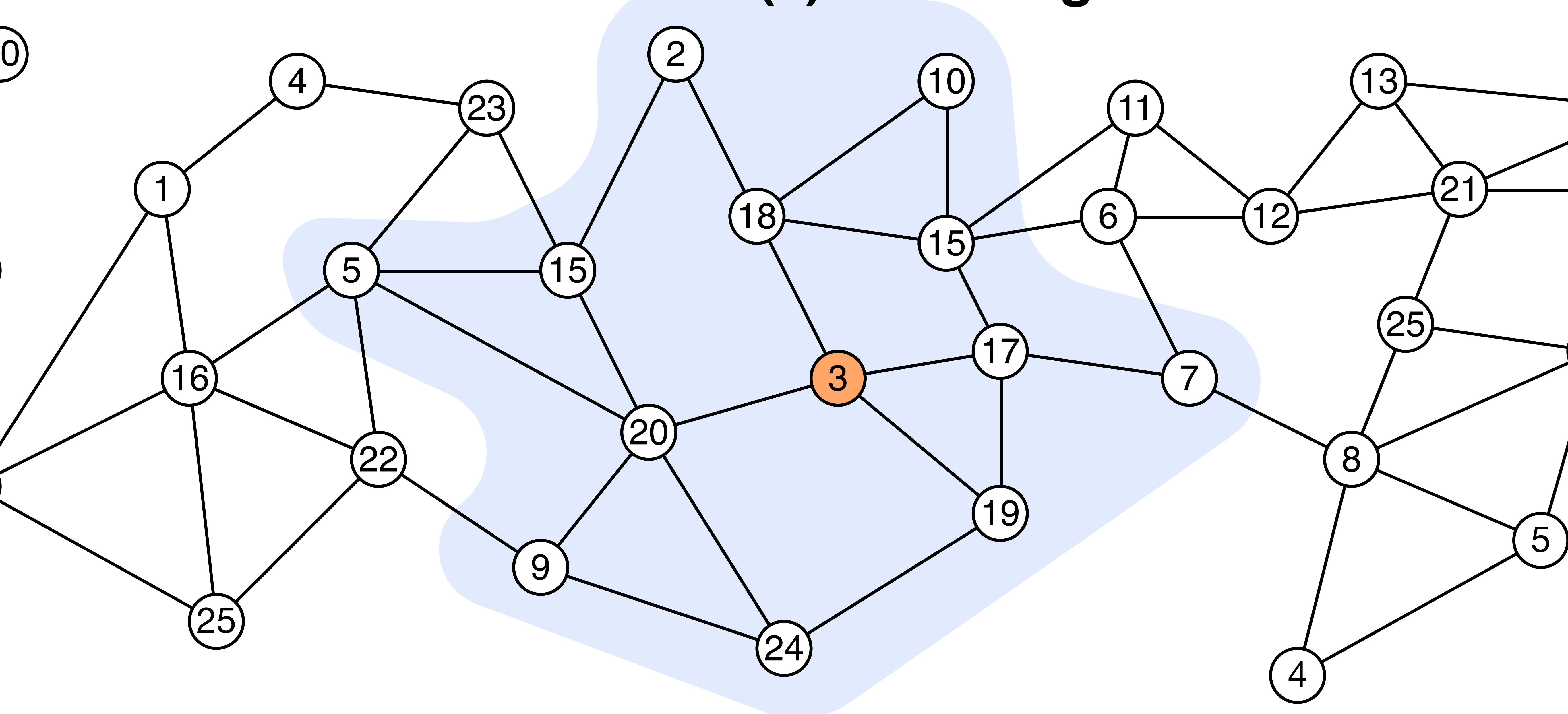
Label s.t. *every node sees every label appear only once* =
**distance O(1)-colouring**



# Algorithm’s view

Label s.t. *every node sees every label appear only once* =

**distance O(1)-colouring**




= looks *locally* like an **instance of size 25**

# Simulation speed-up

- Given algorithm **A** with running time $\mathbf{t = o(\log_\Delta n)}$
- Since **A** is *sublogarithmic*, must be some $\mathbf{n_0}$ s.t. **A** *doesn't see the whole graph* on any instance of size $\mathbf{n_0}$ (i.e. even on *expanders*)
- Now compute distance $\mathbf{t(n_0)+O(1)}$**-coloring** of **G** in time $\mathbf{O(\log^* n)}$
- Run **A** on that coloring

# Simulation speed-up

- Simulation output is *well defined* because *all local views* could come from an instance of **size $n_0$**
- Simulation output is *correct* because in every local neighborhood output follows **rules of the LCL**

# Deterministic speed-up

**Theorem** (*Chang et al., FOCS 2016*): Assume that for LCL L there exists an algorithm with running time $\mathbf{t = o(\log_\Delta n)}$, then there exists an algorithm with running time $\mathbf{t' = O(\log^* n)}$

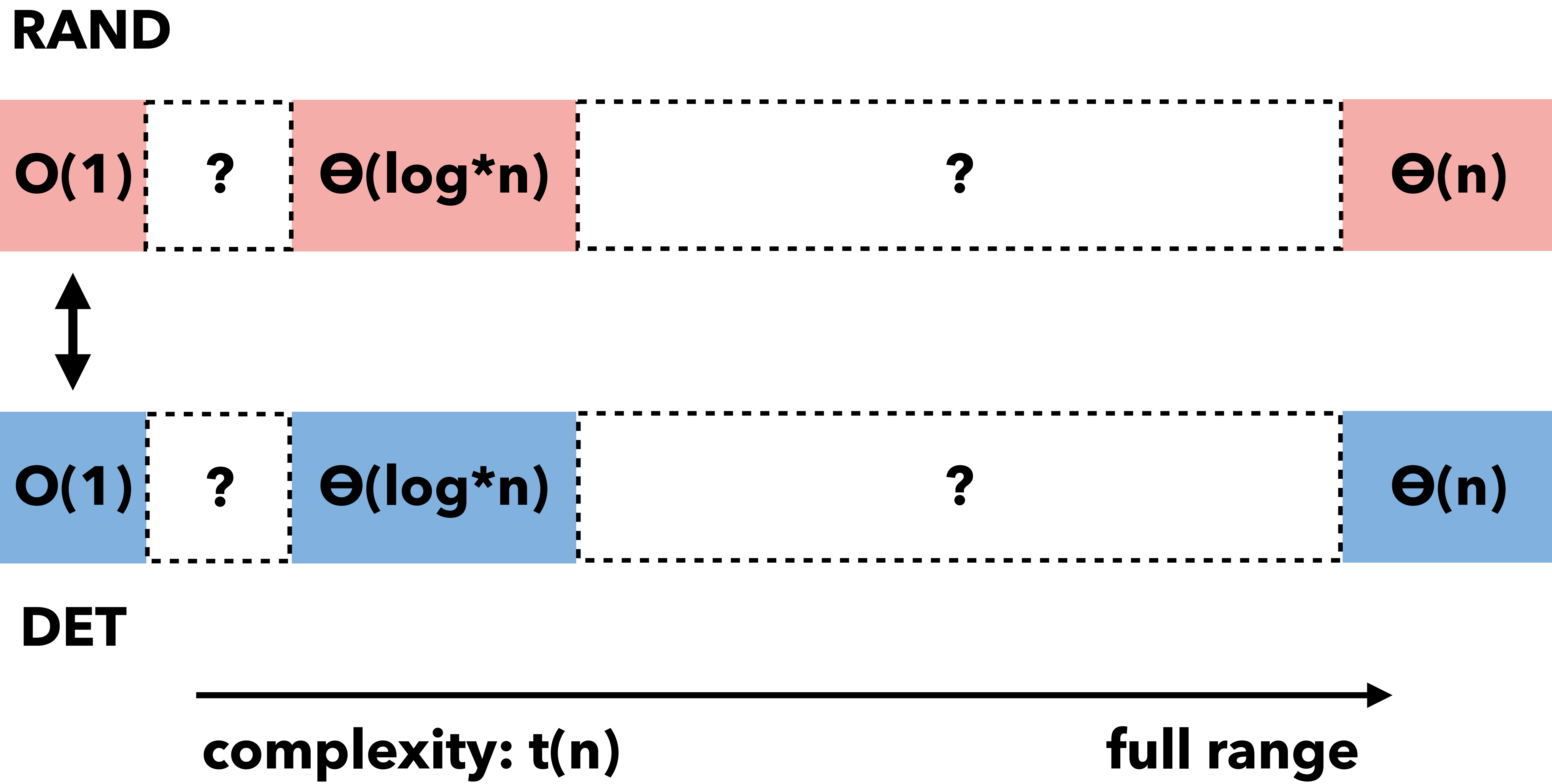
# Deterministic speed-up

- Every *sublogarithmic-time solvable LCL* decomposes into **coloring + constant time**
- *How far can we take the simplified form of sublogarithmic-time algorithms*?
- *Simulation speed-up for other families of problems*?

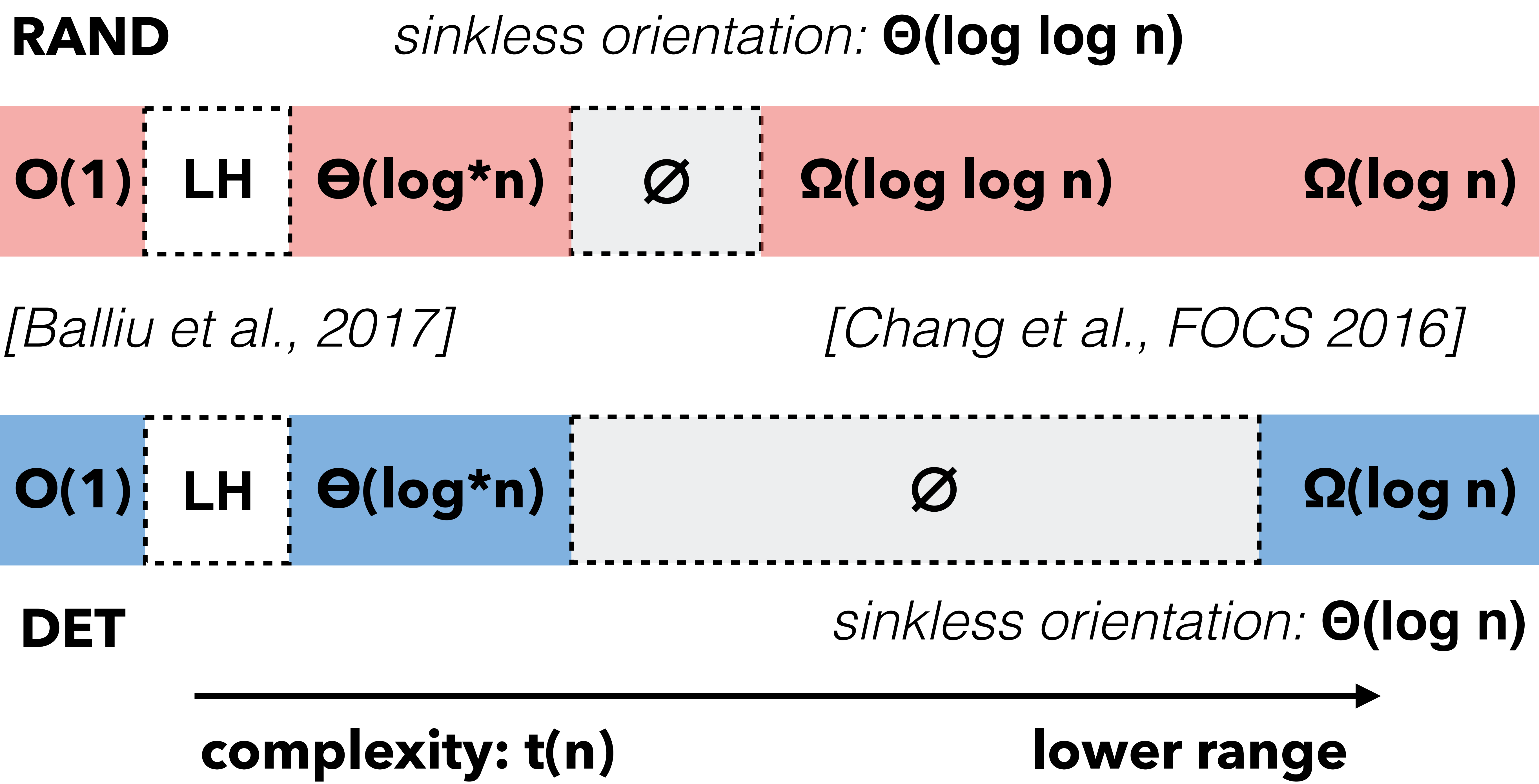
# The complexity zoo: recent developments

# LCL complexity zoo

*State of the art circa 2015*

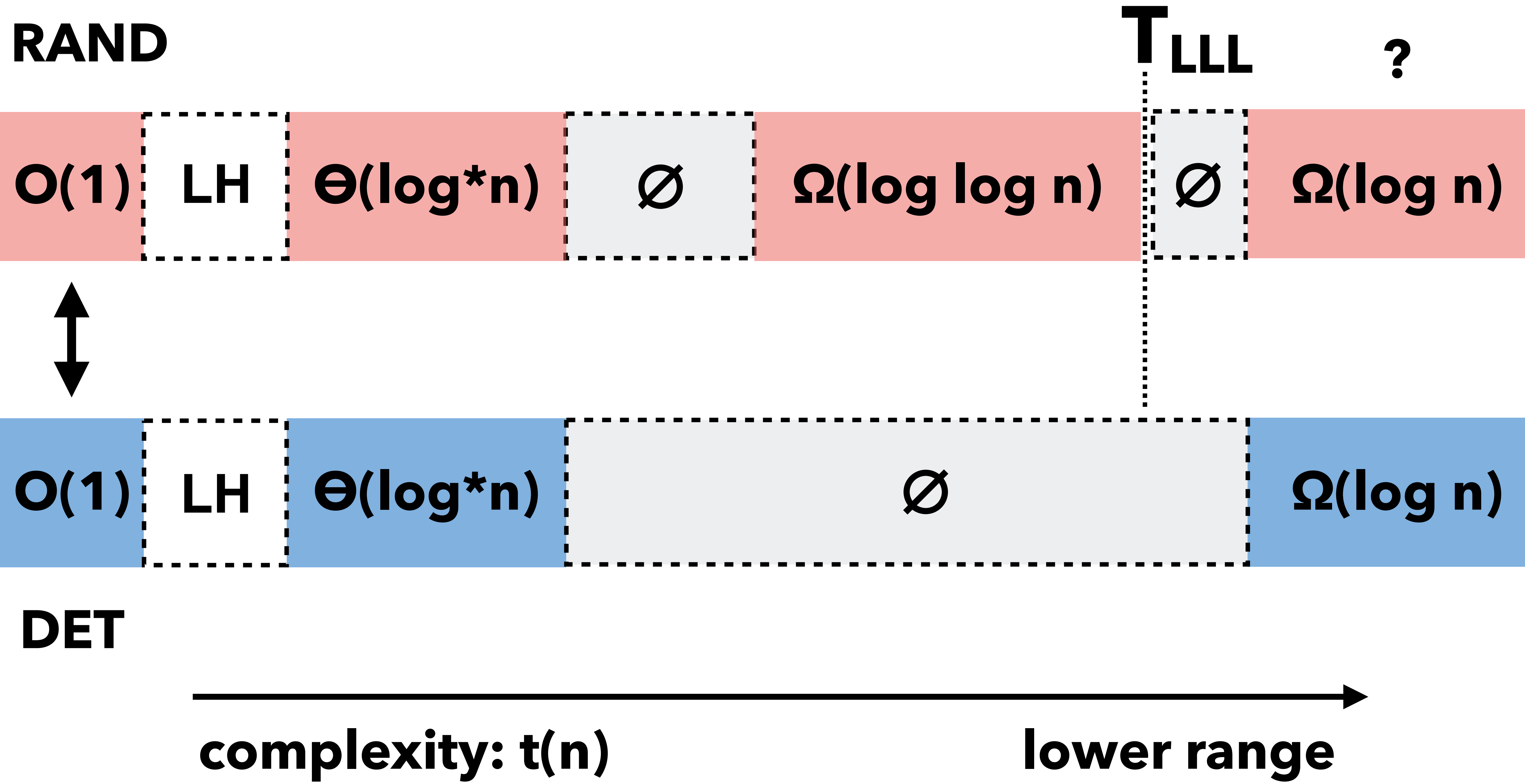


# LCL complexity zoo



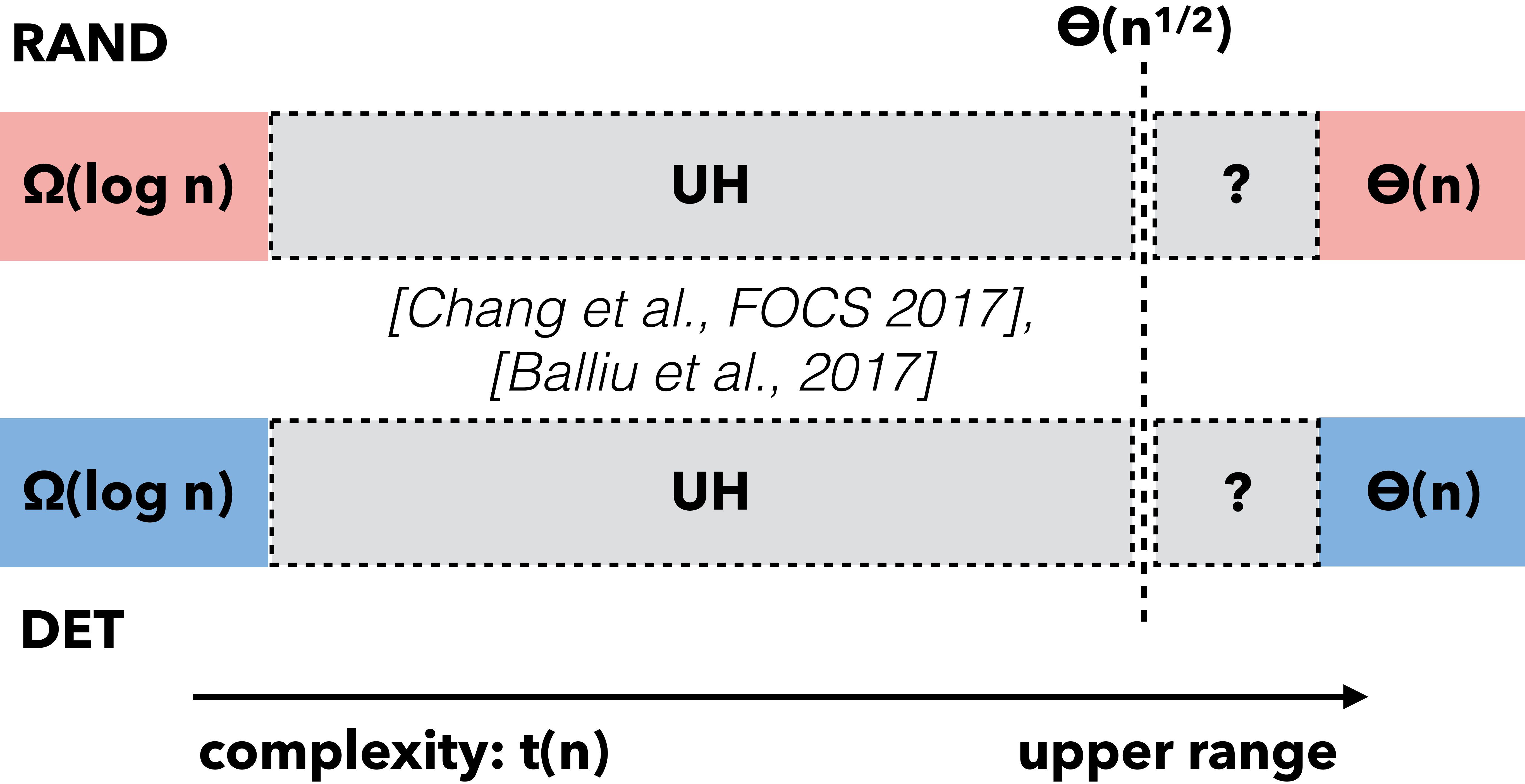
# LCL complexity zoo

*State of the art circa 2017*



# LCL complexity zoo

*State of the art circa 2017*



# Recapping

- Proof technique arguably simple
- A new proof hammer: *where are the nails*?
  - Simulation invariants: *graph girth, success probability, color palette*
  - **New simulation invariants** e.g. related to $\boldsymbol{\Delta}$?
- **Simulation** is an extremely powerful general approach

Thank you!
Questions?