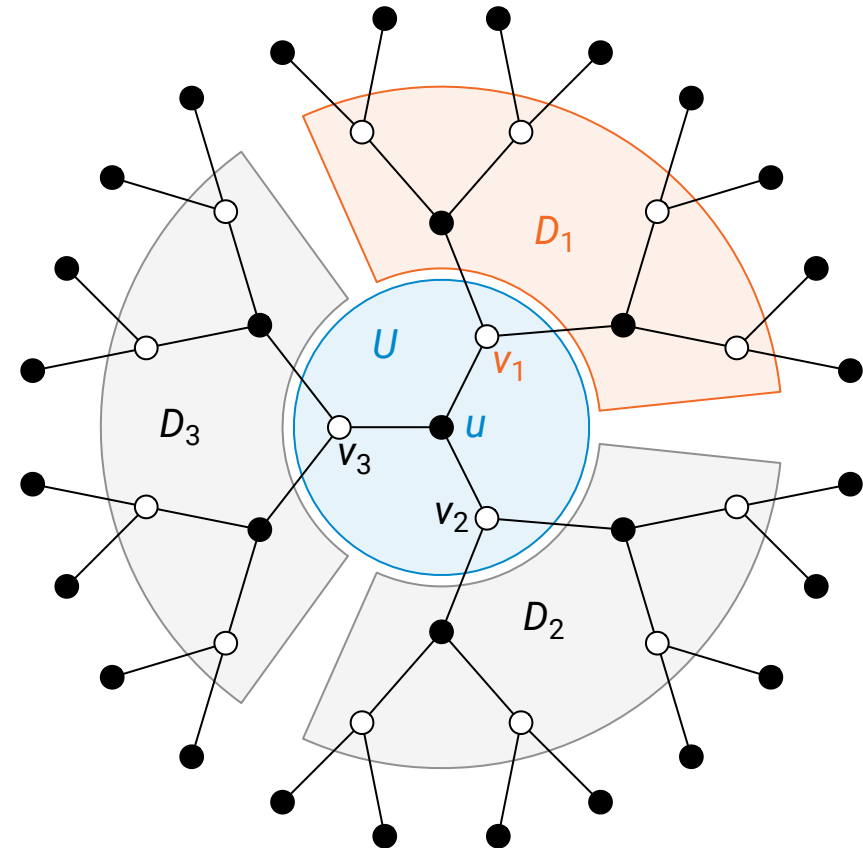


Locality lower bounds through round elimination

Jukka Suomela

Aalto University, Finland



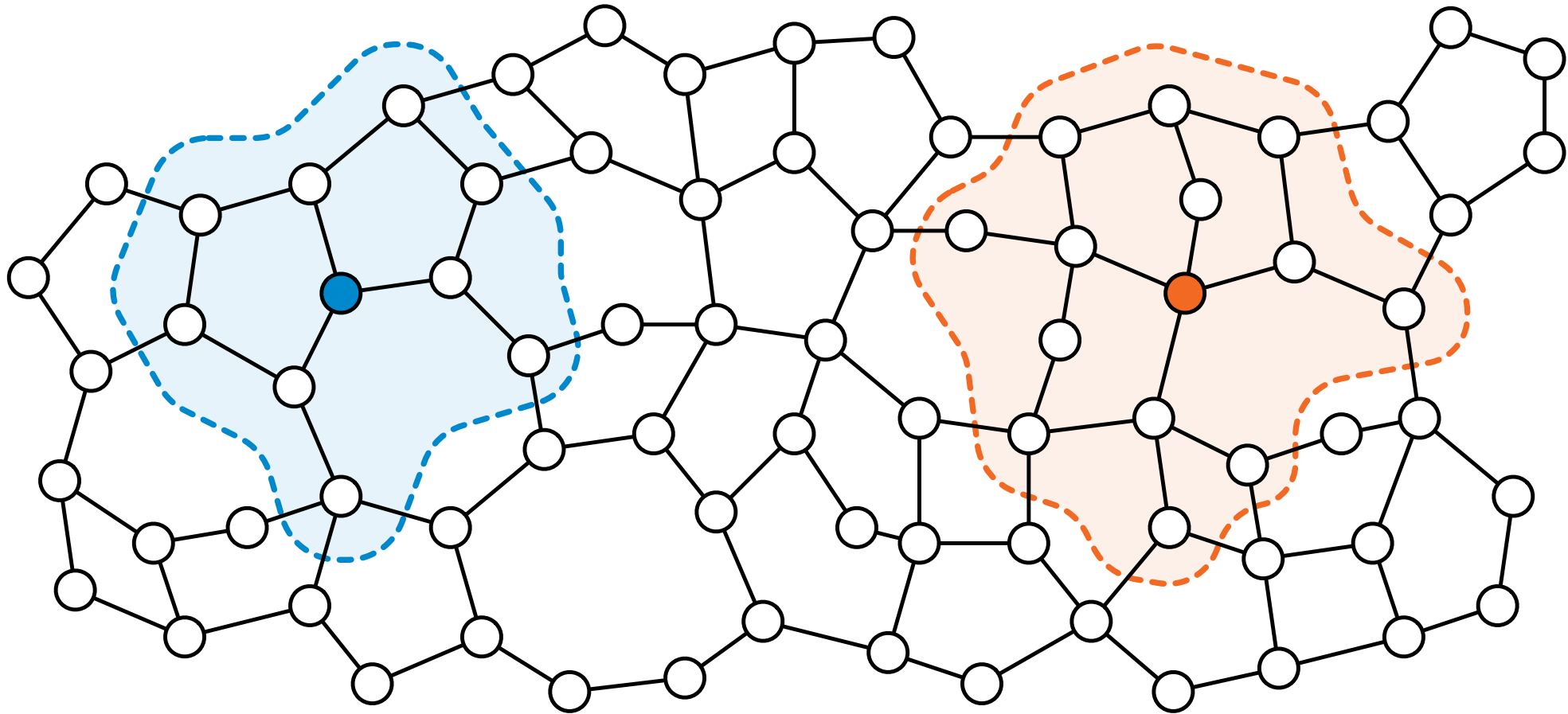
Joint work with

- Alkida Balliu
- Sebastian Brandt
- Orr Fischer
- Juho Hirvonen
- Barbara Keller
- Tuomo Lempiäinen
- Dennis Olivetti
- Mikaël Rabie
- Joel Rybicki
- Jara Uitto

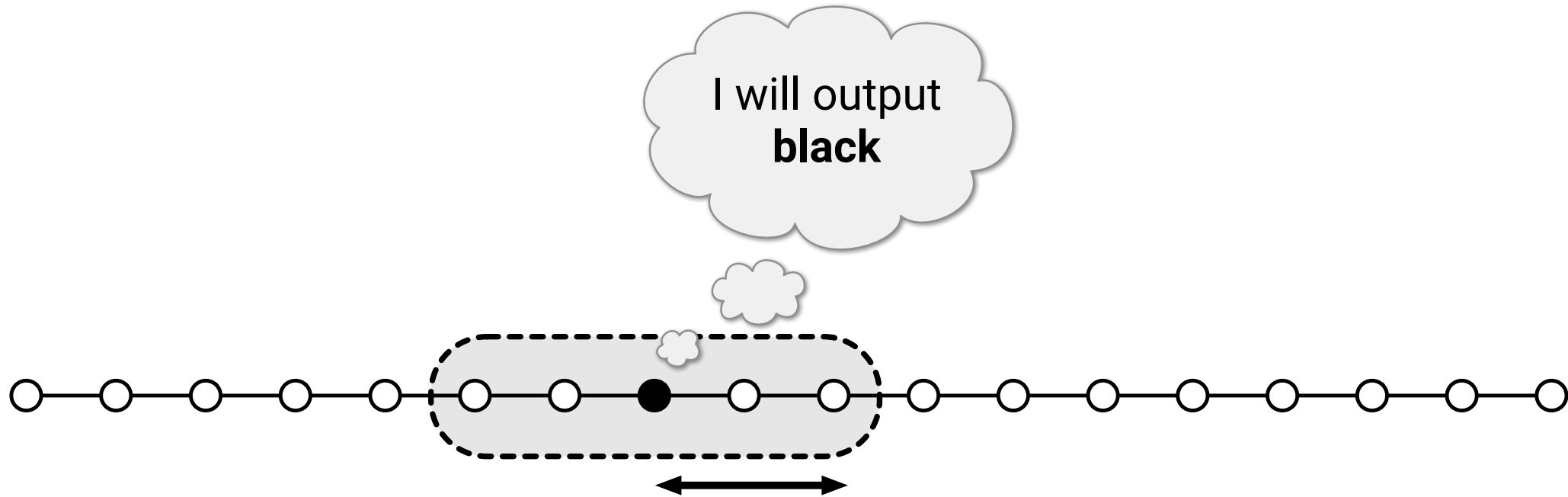
Joint work with

- Alkida Balliu
- **Sebastian Brandt**
- Orr Fischer
- Juho Hirvonen
- Barbara Keller
- Tuomo Lempiäinen
- Dennis Olivetti
- Mikaël Rabie
- Joel Rybicki
- Jara Uitto

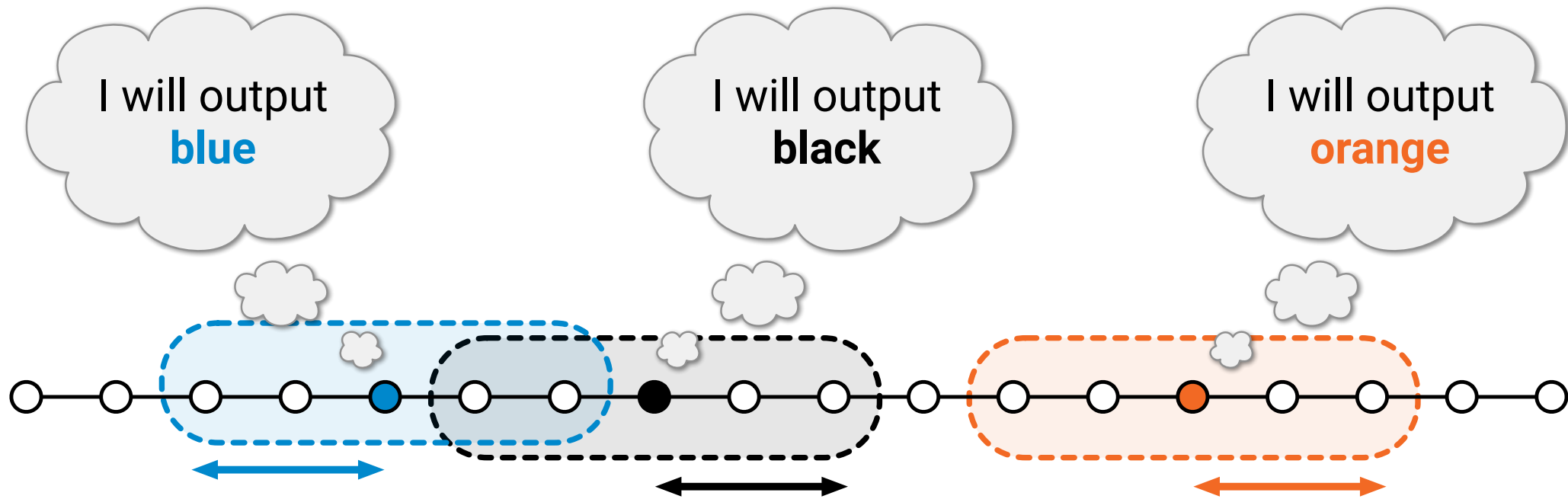
Locality = how far do I need to see to produce my own part of the solution?



Locality = how far do I need to see to produce my own part of the solution?



Locality = how far do I need to see to produce my own part of the solution?



Locality = how far do I need to see to produce my own part of the solution?

Local outputs form a globally consistent solution



Locality: formalization

- “**LOCAL**” model of distributed computing:
 - ***graph = communication network***
 - **node** = processor
 - **edge** = communication link
 - all nodes have unique identifiers
 - ***time = number of communication rounds***
 - **round** = nodes exchange messages with all neighbors
 - 1 communication round: all nodes can learn everything within distance 1
 - T communication rounds: all nodes can learn everything within distance T
- ***Time = distance***

Locality: examples

- Setting: graph with n nodes, maximum degree $\Delta = O(1)$
- **Maximal independent set:**
 $\Theta(\log^* n)$ randomized, $\Theta(\log^* n)$ deterministic
- **Sinkless orientation:**
 $\Theta(\log \log n)$ randomized, $\Theta(\log n)$ deterministic
 - orient edges such that all nodes of degree ≥ 3 have outdegree ≥ 1

How to study locality?

Proving locality upper & lower bounds

Locality: proving upper bounds

- Find a *function* that maps local neighborhoods to local outputs
- Design a fast distributed *message-passing algorithm*
- Design a slow distributed algorithm and apply “*speedup*” arguments to turn it into a fast distributed algorithm
 - e.g. $o(n) \rightarrow O(\log^* n)$ for “LCL problems” in cycles
- Design a fast *centralized sequential* algorithm model and turn it into a fast distributed algorithm
 - e.g. greedy strategy \rightarrow SLOCAL algorithm \rightarrow LOCAL algorithm

Locality: proving lower bounds

- ***Indistinguishability***
 - same local view \rightarrow same output
- ***Adaptive constructions***
 - inductively construct a bad input for this specific algorithm
- ***Ramsey-type arguments***
 - “monochromatic set” \approx bad choice of identifiers
- ***Speedup & derandomization arguments and reductions***
 - locality $R \rightarrow$ locality $R' \rightarrow$ not possible

Locality: proving lower bounds

- **Indistinguishability**
 - same local view \rightarrow same output
- **Adaptive constructions**
 - inductively construct a bad input for this specific algorithm
- **Ramsey-type arguments**
 - “monochromatic set” \approx bad choice of identifiers
- **Speedup & derandomization arguments and reductions**
 - locality $R \rightarrow$ locality $R' \rightarrow$ not possible

Today's focus:
“round elimination”
technique for proving
locality lower bounds

Round elimination

Round elimination technique

- **Given:**
 - algorithm A_0 solves problem P_0 in T rounds
- **We construct:**
 - algorithm A_1 solves problem P_1 in $T - 1$ rounds
 - algorithm A_2 solves problem P_2 in $T - 2$ rounds
 - algorithm A_3 solves problem P_3 in $T - 3$ rounds
 - ...
 - algorithm A_T solves problem P_T in 0 rounds
- But P_T is nontrivial, so A_0 cannot exist

Linial (1987, 1992): coloring cycles

- **Given:**

- algorithm A_0 solves **3-coloring** in $T = o(\log^* n)$ rounds

- **We construct:**

- algorithm A_1 solves **2^3 -coloring** in $T - 1$ rounds
- algorithm A_2 solves **2^{2^3} -coloring** in $T - 2$ rounds
- algorithm A_3 solves **$2^{2^{2^3}}$ -coloring** in $T - 3$ rounds
- ...
- algorithm A_T solves **$o(n)$ -coloring** in **0** rounds

- But **$o(n)$ -coloring** is nontrivial, so A_0 cannot exist

Brandt et al. (2016): sinkless orientation

- **Given:**
 - algorithm A_0 solves **sinkless orientation** in $T = o(\log n)$ rounds
- **We construct:**
 - algorithm A_1 solves **sinkless coloring** in $T - 1$ rounds
 - algorithm A_2 solves **sinkless orientation** in $T - 2$ rounds
 - algorithm A_3 solves **sinkless coloring** in $T - 3$ rounds
 - ...
 - algorithm A_T solves **sinkless orientation** in 0 rounds
- But **sinkless orientation** is nontrivial, so A_0 cannot exist

Round elimination can be automated

Brandt 2019


- **Good news:** always possible for **any graph problem P_0** that is “locally checkable”
 - if problem P_0 has complexity T , we can always find in a mechanical manner problem P_1 that has complexity $T - 1$
 - holds for tree-like neighborhoods (e.g. high-girth graphs)
- **Bad news:** this does not directly give a lower bound
 - P_1 is not necessarily any **natural graph problem**
 - P_1 does not necessarily have a **small description**
 - how do we prove that P_1, P_2, P_3 , etc. are **nontrivial problems**?

Round elimination and fixed points

- Sometimes we are very lucky:
 - $P_0 = \text{sinkless orientation}$
 - $P_1 = \text{something}$ (no need to understand it)
 - $P_2 = \text{sinkless orientation}$ ⚡
- If you are feeling optimistic: just apply round elimination in a mechanical manner for a small number of steps and see if you reach a **fixed point** or **cycle**
 - or you reach a well-known hard problem
- Open question: **exactly when does this happen?**

Round elimination and “rounding down”

- Sometimes some amount of creativity is needed:
 - $P_0 = k$ -coloring cycles
 - $P_1 =$ something complicated with 2^k possible output labels
 - **define:** $Q_1 = 2^k$ -coloring cycles
 - **observation:** solution to P_1 implies a solution to Q_1

P_0 takes **exactly T** rounds
→ P_1 takes **exactly $T - 1$** rounds
→ Q_1 takes **at most $T - 1$** rounds
→ ...
→ Q_T takes **at most 0** rounds 

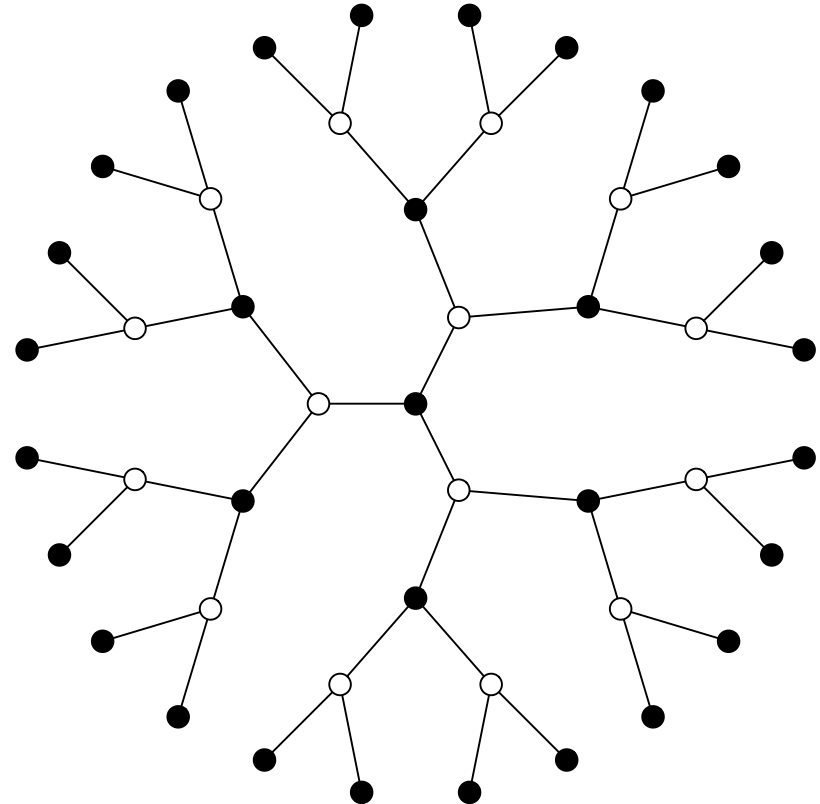
How does it work?

Correct formalism

- We will need the *right formalism* for the graph problems that we study
- It will look seemingly arbitrary and very restrictive at first
- No worries, you can *encode* a broad range of **locally checkable problems** in this formalism with some effort
 - maximal matching, maximal independent set, vertex coloring, edge coloring, sinkless orientation ...

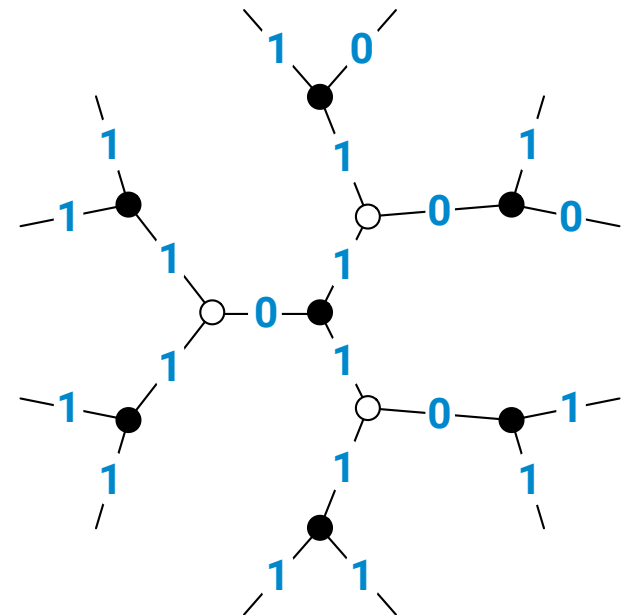
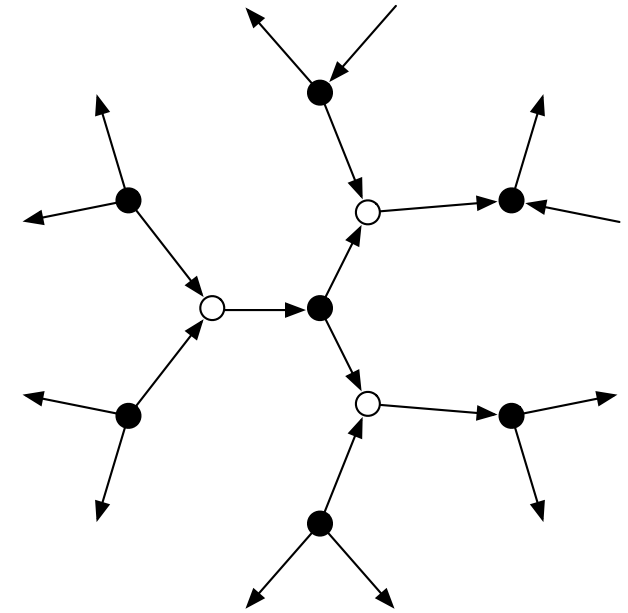
Correct formalism: edge labeling in bipartite graphs

- Assumption: input graph properly 2-colored (“*white*” / “*black*”)
- Finite set of possible **edge labels**
- **White** constraint:
 - feasible multiset of labels on edges adjacent to a white node
- **Black** constraint:
 - feasible multiset of labels on edges adjacent to a black node



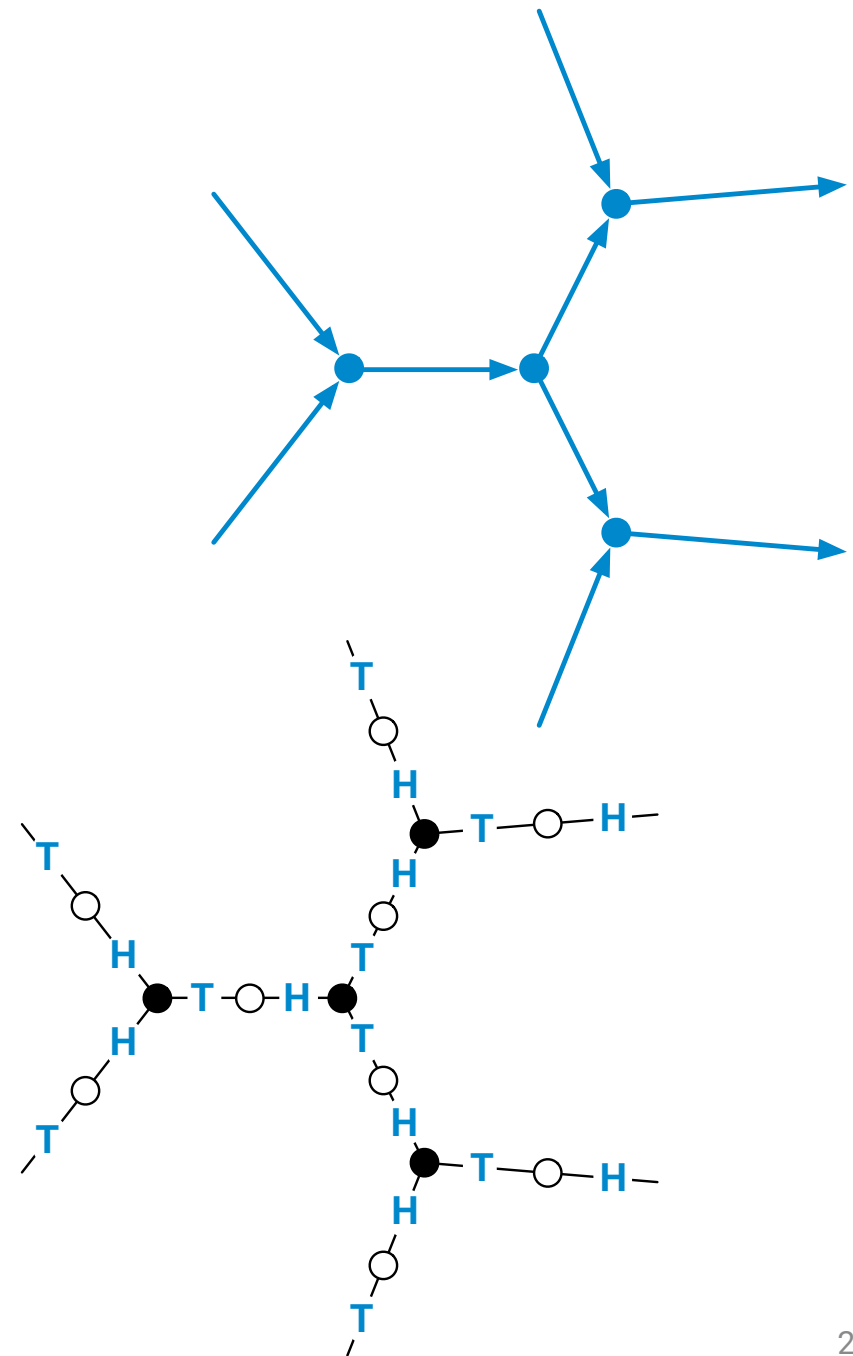
Example 1: sinkless orientation

- Setting: bipartite 3-regular graphs
- Encoding: use original graph
 - “0” = orient from white to black
 - “1” = orient from black to white
- **White** constraint:
 - {0, 0, 0}, {0, 0, 1} or {0, 1, 1}
- **Black** constraint:
 - {0, 0, 1}, {0, 1, 1} or {1, 1, 1}



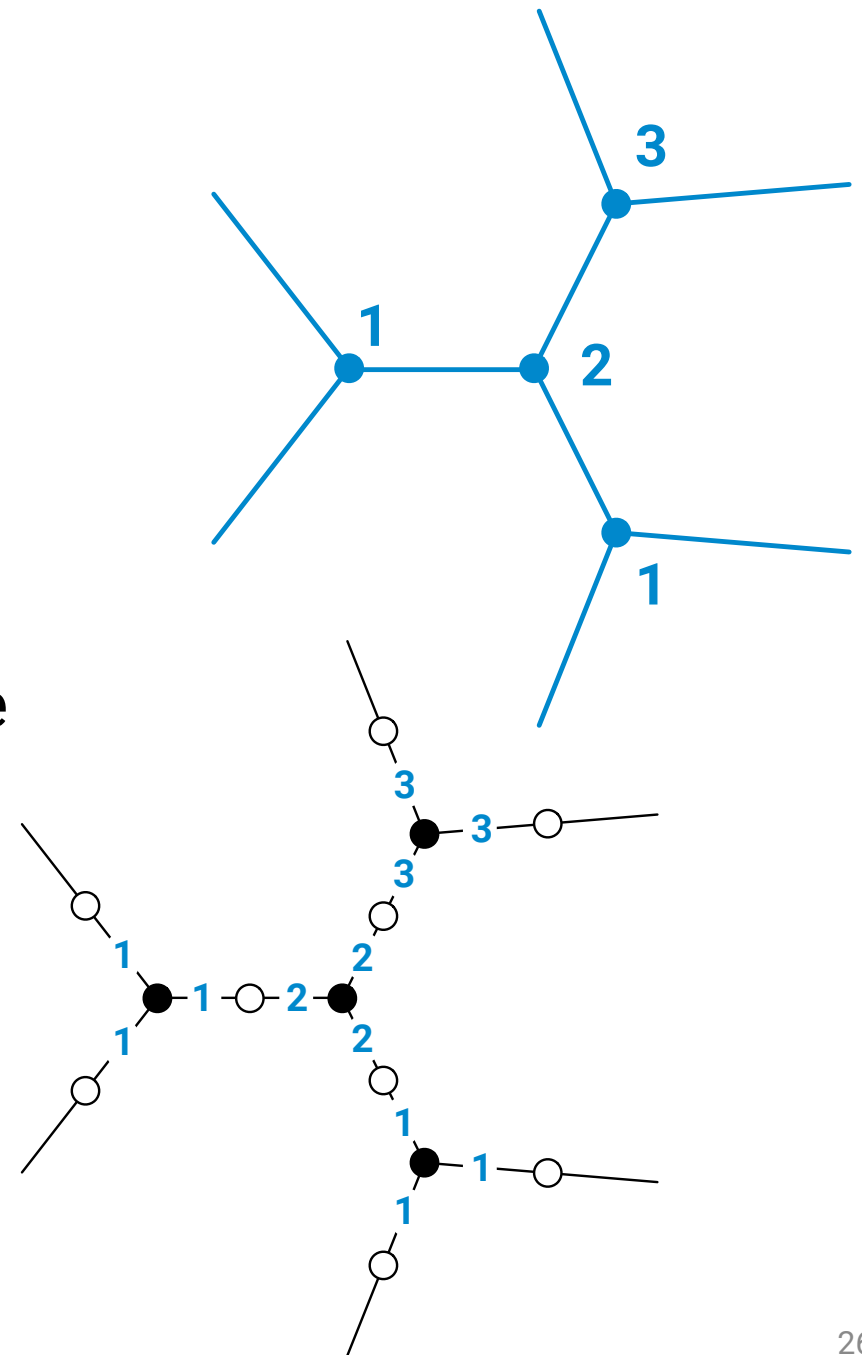
Example 2: sinkless orientation

- Setting: 3-regular graphs
- Encoding: subdivide edges
 - **white** = edge, **black** = node
 - “**H**” = head, “**T**” = tail
- **White** constraint:
 - {**H**, **T**}
- **Black** constraint:
 - {**H**, **H**, **T**}, {**H**, **T**, **T**} or {**T**, **T**, **T**}



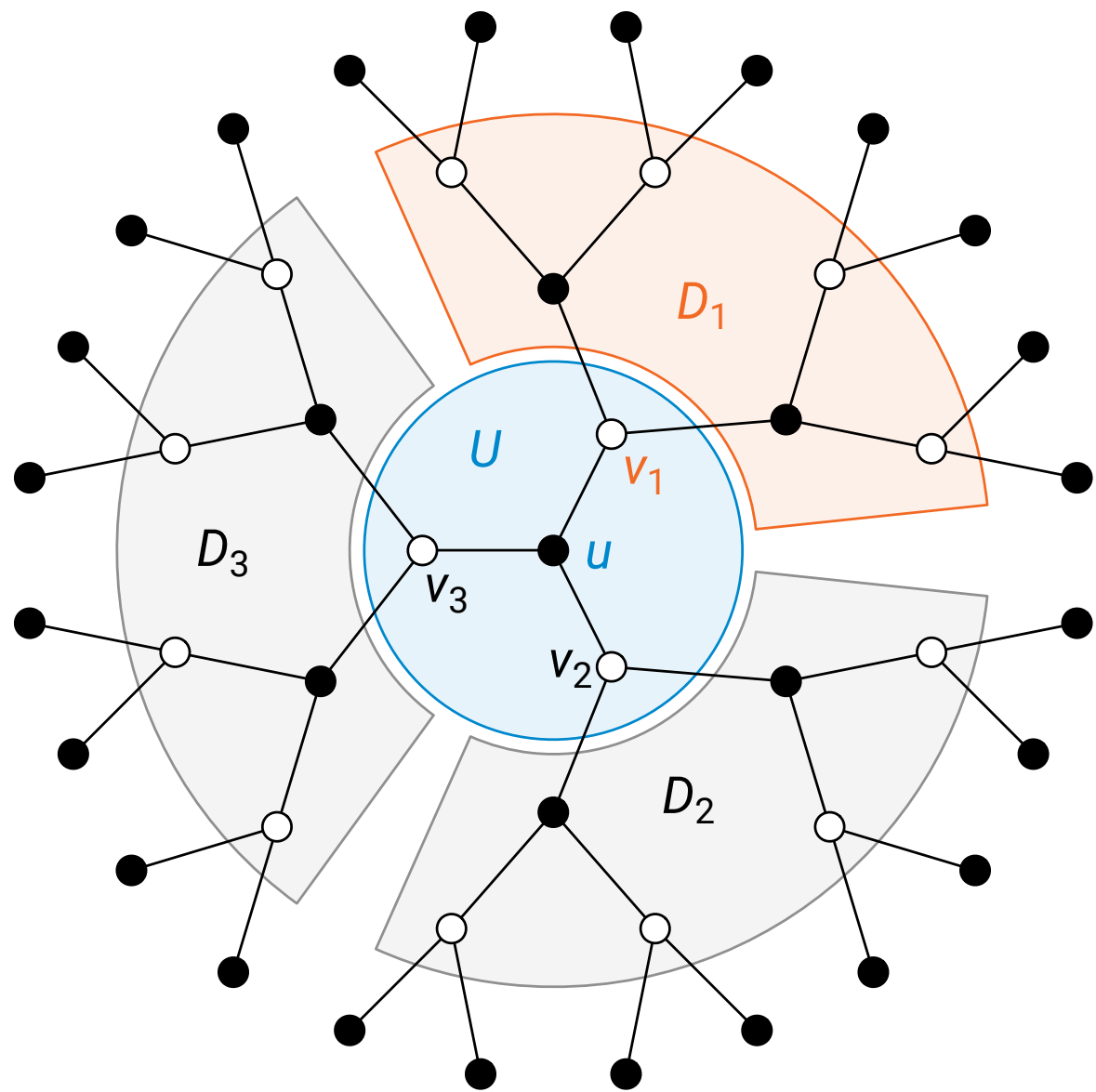
Example 3: vertex coloring

- Setting: 3-regular graphs
- Encoding: subdivide edges
 - **white** = edge, **black** = node
 - “1”, “2”, “3” = color of incident black node
- **White** constraint:
 - {1, 2} or {1, 3} or {2, 3}
- **Black** constraint:
 - {1, 1, 1}, {2, 2, 2} or {3, 3, 3}



Correct formalism: white and black algorithms

- **White** algorithm:
 - each **white** node produces labels on its incident edges
 - **black** nodes do nothing
 - satisfies white and black constraints
- **Black** algorithm:
 - each **black** node produces labels on its incident edges
 - **white** nodes do nothing
 - satisfies white and black constraints
- White and black complexity within ± 1 round of each other



Round elimination

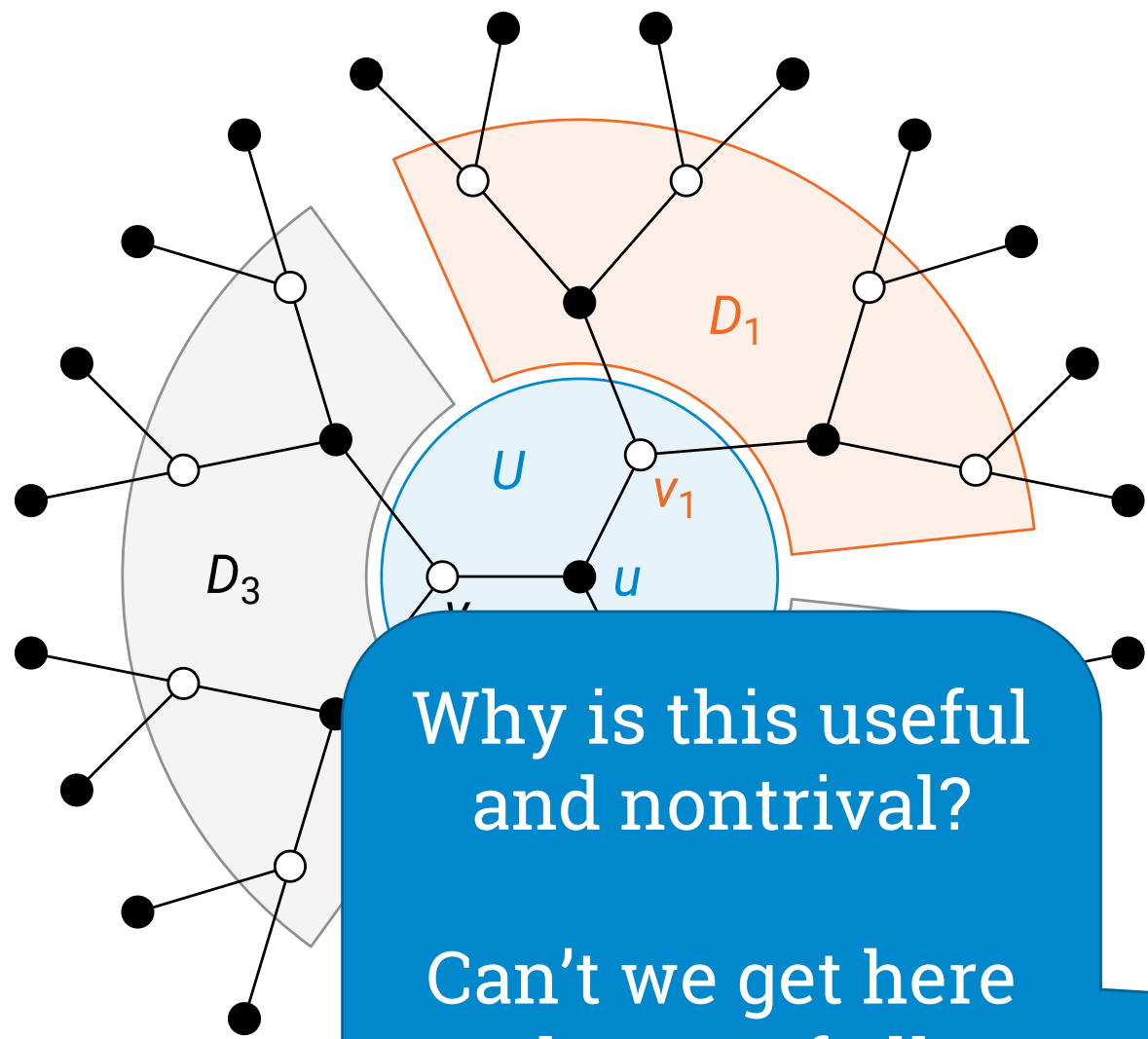
Given: **white algorithm A** that runs in $T = 2$ rounds

- v_1 in **A** sees U and D_1

Construct: **black algorithm A'** that runs in $T - 1 = 1$ rounds

- u in **A'** only sees U

A': what is the **set of possible outputs of A** for edge $\{u, v_1\}$ over all possible inputs in D_1 ?



Why is this useful and nontrivial?
 Can't we get here the set of all possible outputs?

Round elimination

Given: **white algorithm A** that runs in $T = 2$ rounds

- v_1 in **A** sees U and D_1

Construct: **black algorithm A'** that runs in $T - 1 = 1$ rounds

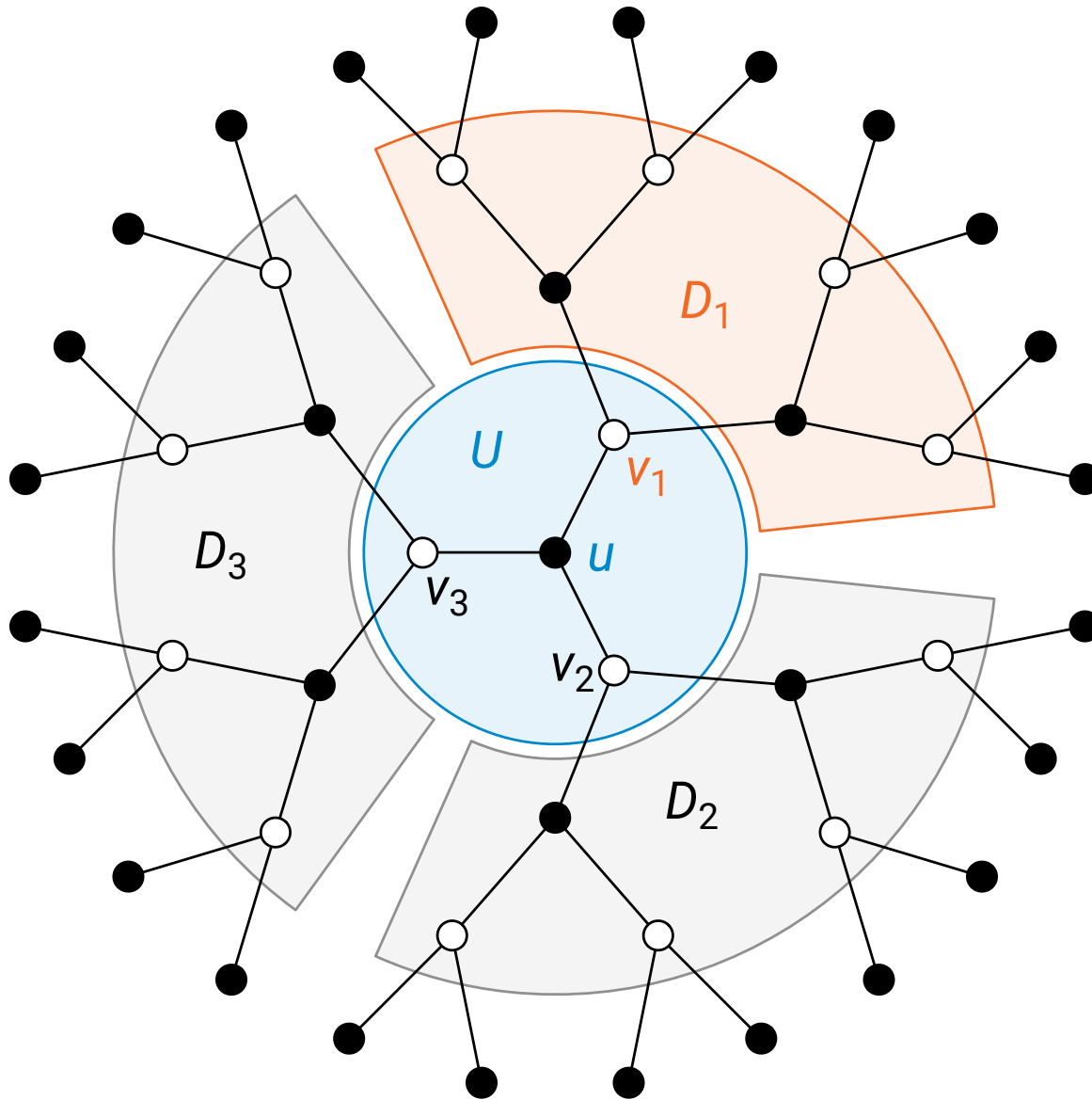
- u in **A'** only sees U

A': what is the **set of possible outputs of A** for edge $\{u, v_1\}$ over all possible inputs in D_1 ?

Example: edge coloring

Independence!

- Assume there is some extension D_1 such that v_1 labels $\{u, v_1\}$ green
- Assume there is some extension D_2 such that v_2 labels $\{u, v_2\}$ green
- Then we can construct an input in which both $\{u, v_1\}$ and $\{u, v_2\}$ are green



Algorithm A' has to do something nontrivial

Here: sets incident to black nodes have to be non-empty and disjoint

They contain enough information so that we could recover a proper edge coloring in 1 extra round

Example: edge coloring

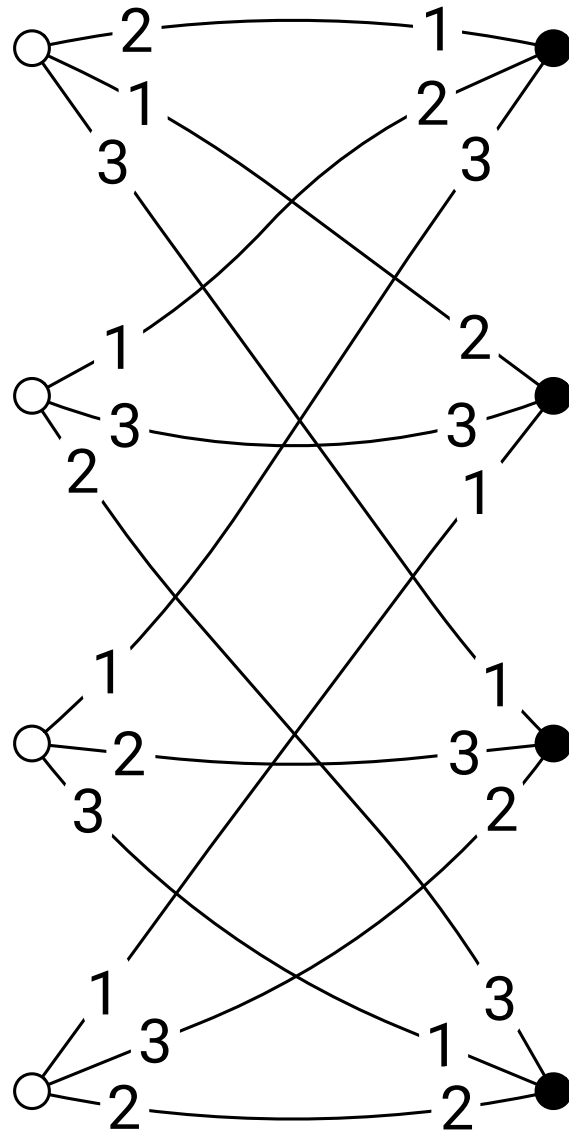
Independence!

- Assume there is some extension D_1 such that v_1 labels $\{u, v_1\}$ green
- Assume there is some extension D_2 such that v_2 labels $\{u, v_2\}$ green
- Then we can construct an input in which both $\{u, v_1\}$ and $\{u, v_2\}$ are green



Example: bipartite maximal matching

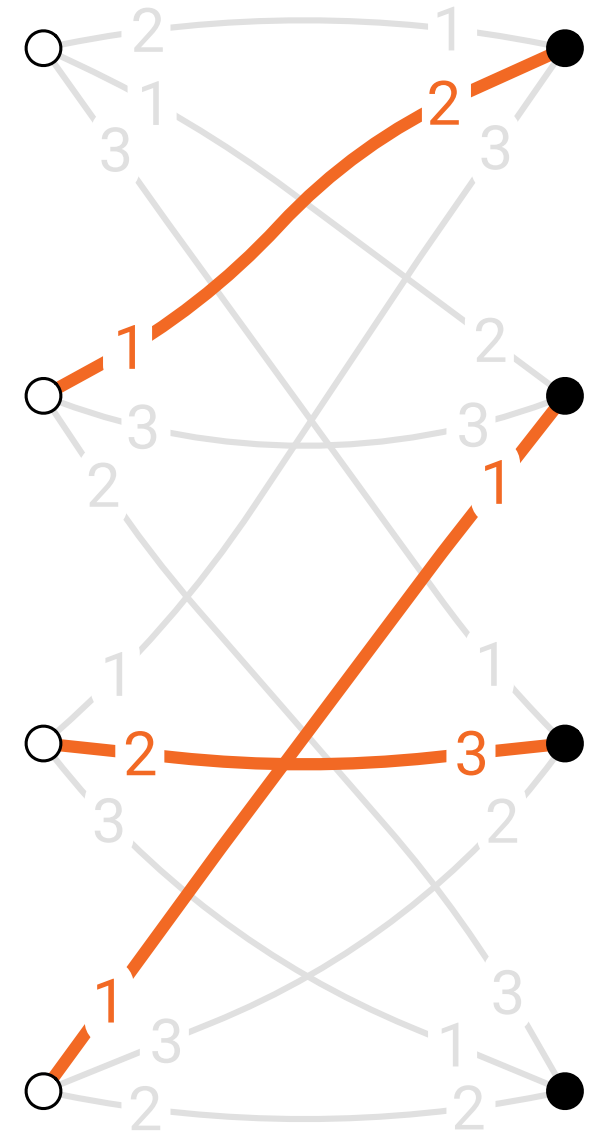
computer network with port numbering

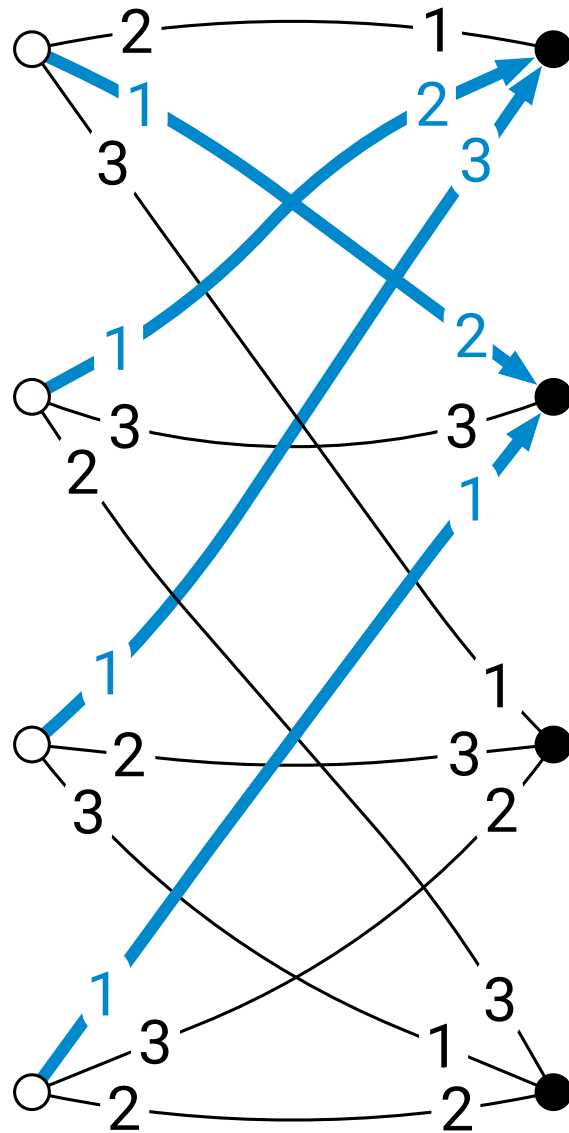


bipartite, 2-colored graph

Δ -regular (here $\Delta = 3$)

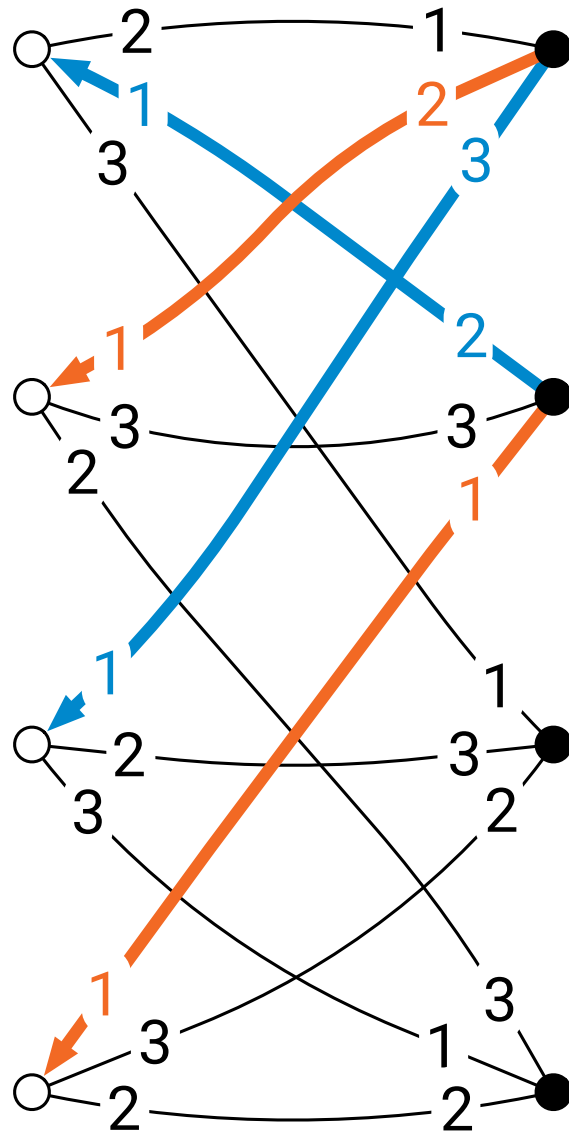
output: *maximal matching*





Very simple algorithm

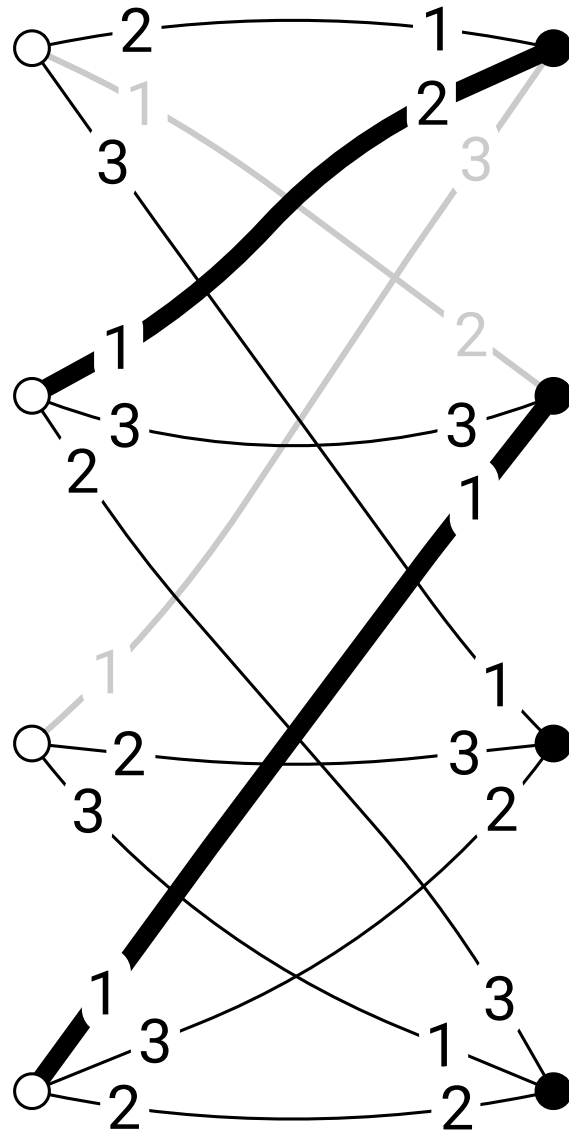
unmatched white nodes:
send *proposal* to port 1



Very simple algorithm

unmatched white nodes:
send *proposal* to port 1

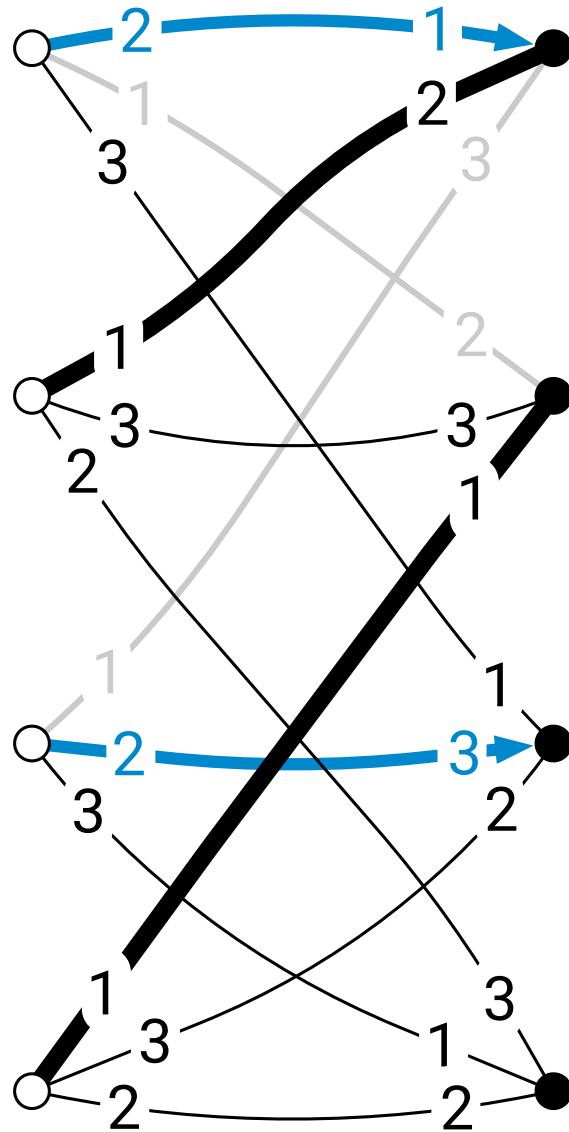
black nodes:
accept the first proposal you
get, *reject* everything else
(break ties with port numbers)



Very simple algorithm

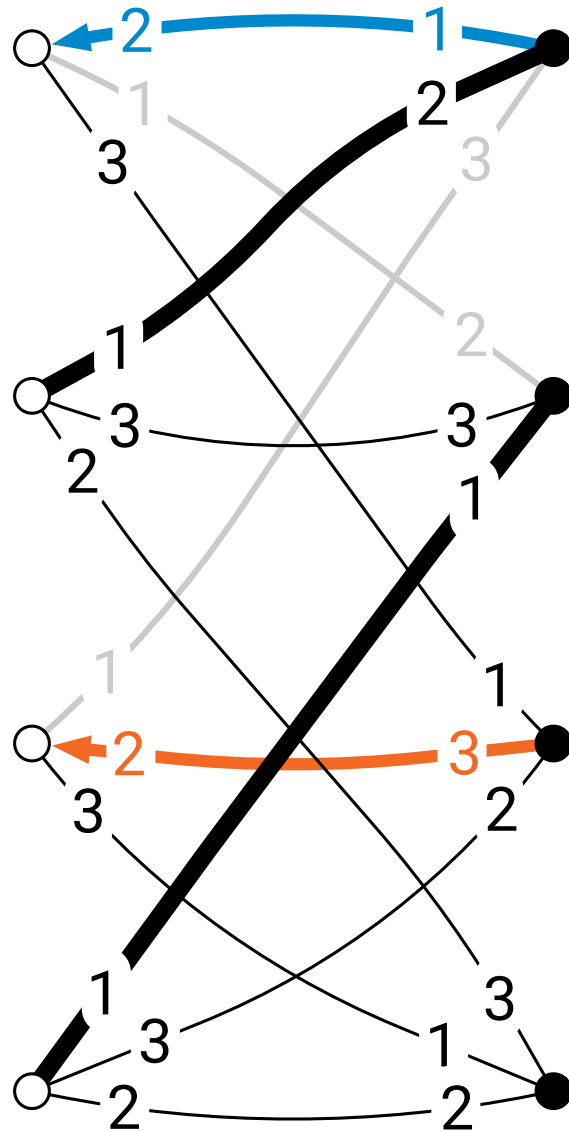
unmatched white nodes:
send *proposal* to port 1

black nodes:
accept the first proposal you
get, *reject* everything else
(break ties with port numbers)



Very simple algorithm

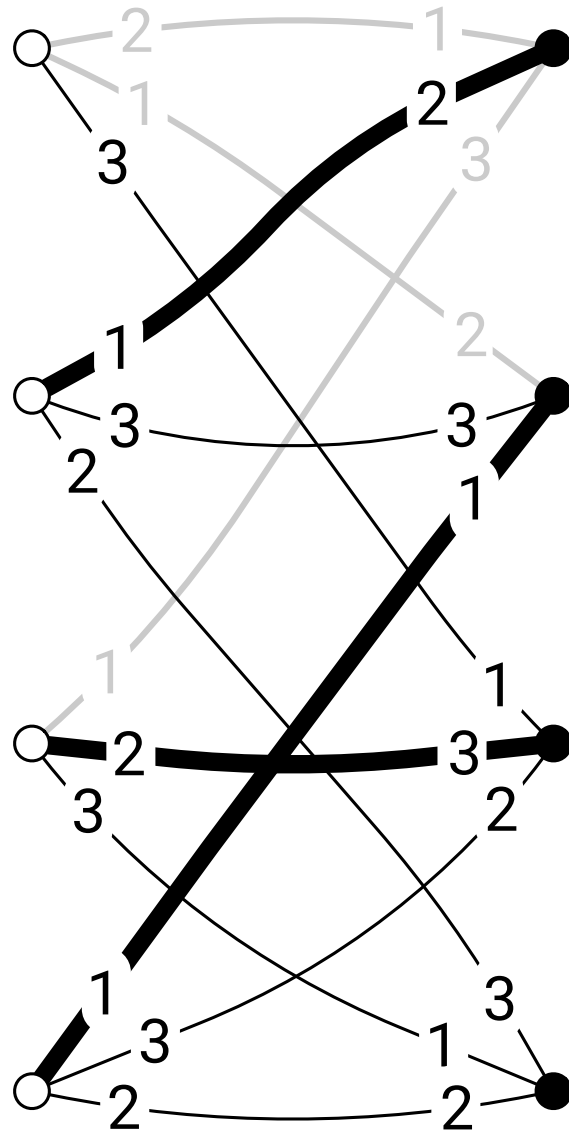
unmatched white nodes:
send *proposal* to port 2



Very simple algorithm

unmatched white nodes:
send *proposal* to port 2

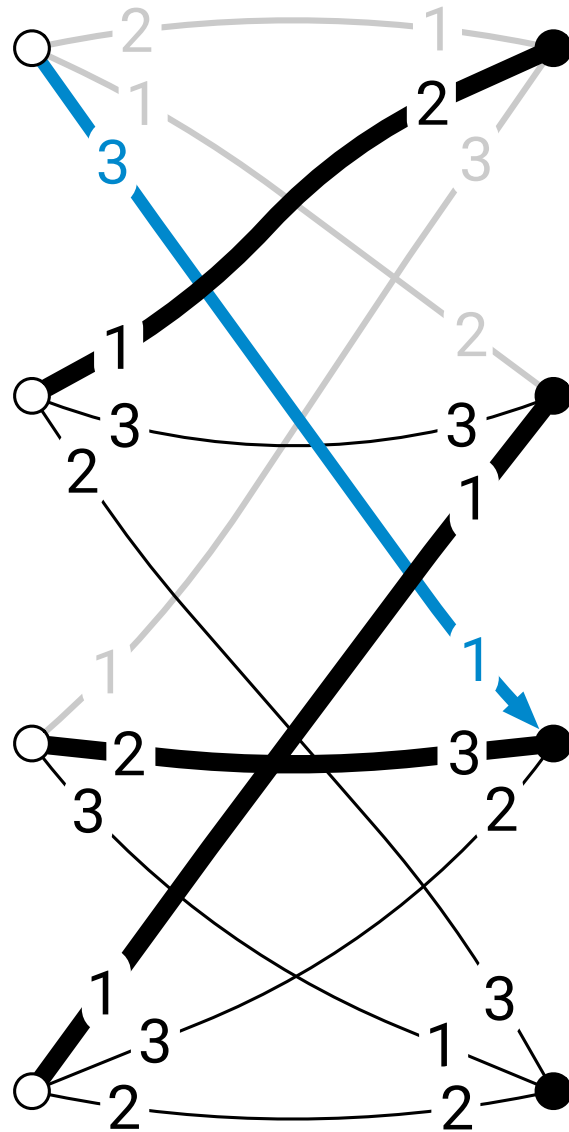
black nodes:
accept the first proposal you
get, *reject* everything else
(break ties with port numbers)



Very simple algorithm

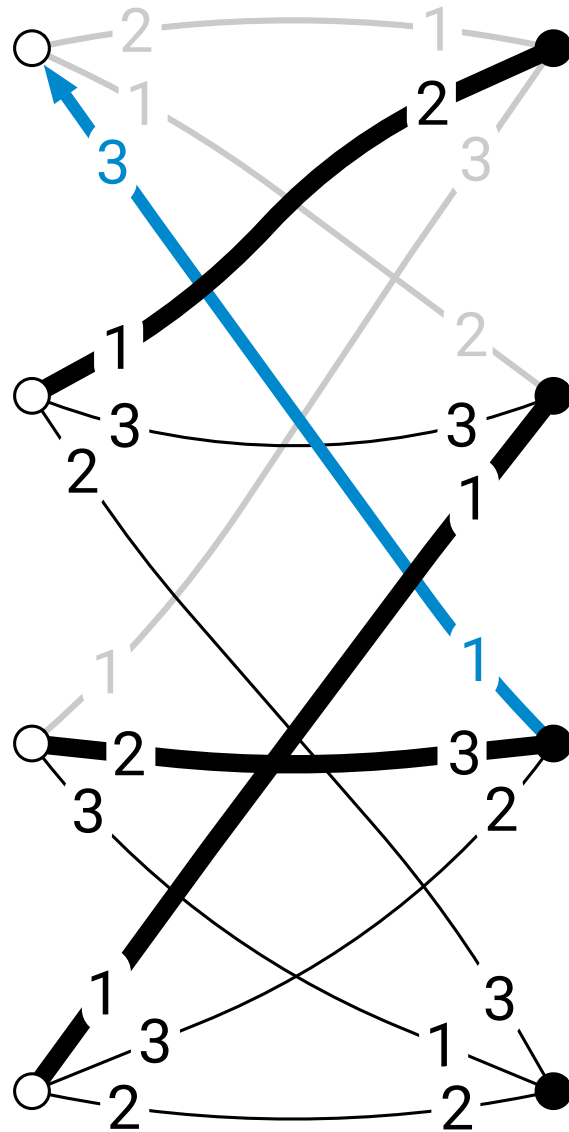
unmatched white nodes:
send *proposal* to port 2

black nodes:
accept the first proposal you
get, *reject* everything else
(break ties with port numbers)



Very simple algorithm

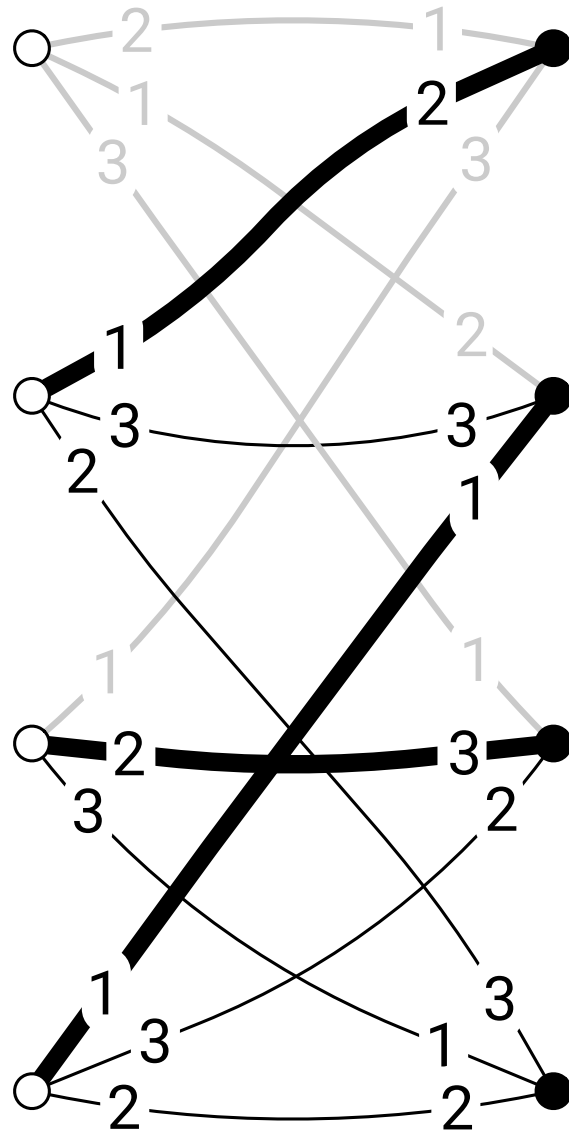
unmatched white nodes:
send *proposal* to port 3



Very simple algorithm

unmatched white nodes:
send *proposal* to port 3

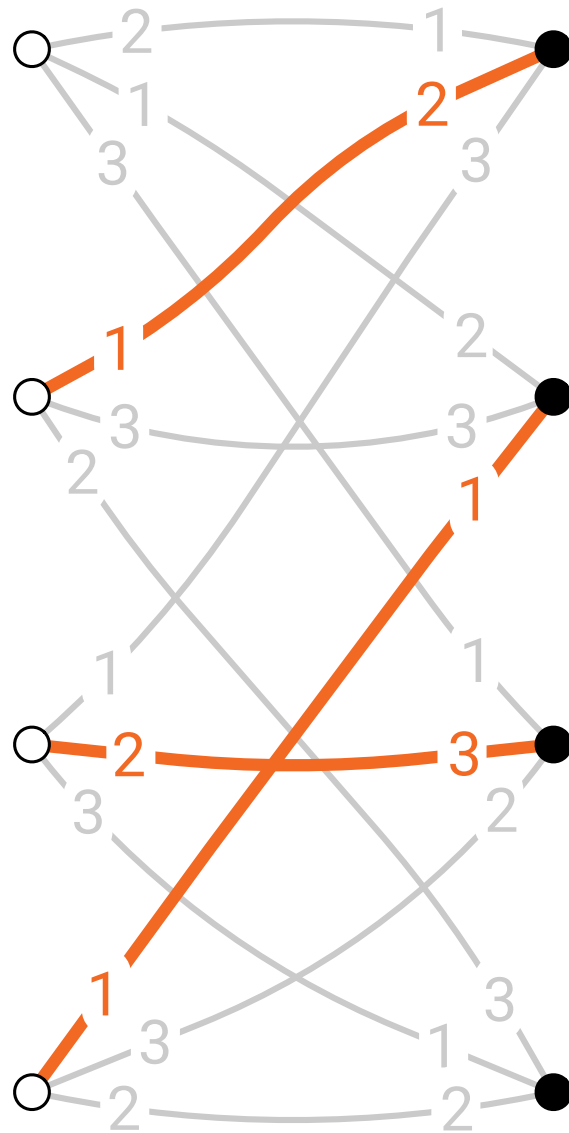
black nodes:
accept the first proposal you
get, *reject* everything else
(break ties with port numbers)



Very simple algorithm

unmatched white nodes:
send *proposal* to port 3

black nodes:
accept the first proposal you
get, *reject* everything else
(break ties with port numbers)



Very simple algorithm

Finds a *maximal matching* in $O(\Delta)$ communication rounds

Note: running time does not depend on n

Bipartite maximal matching

- Maximal matching in very large 2-colored Δ -regular graphs
- Simple algorithm: $O(\Delta)$ rounds, independently of n
- *Is this optimal?*
 - $o(\Delta)$ rounds?
 - $O(\log \Delta)$ rounds?
 - 4 rounds??

Lower-bound proof

Round elimination technique for maximal matching

- **Given:**
 - algorithm A_0 solves problem $P_0 = \text{maximal matching}$ in T rounds
- **We construct:**
 - algorithm A_1 solves problem P_1 in $T - 1$ rounds
 - algorithm A_2 solves problem P_2 in $T - 2$ rounds
 - algorithm A_3 solves problem P_3 in $T - 3$ rounds
 - ...
 - algorithm A_T solves problem P_T in 0 rounds
- But P_T is nontrivial, so A_0 cannot exist

What are
the right
problems
 P_i here?

Round elimination technique for maximal matching

- **Given:**
 - algorithm A_0 solves problem $P_0 = \text{maximal matching}$ in T rounds
- **We construct:**
 - algorithm A_1 solves problem P_1 in $T - 1$ rounds
 - algorithm A_2 solves problem P_2 in $T - 2$ rounds
 - algorithm A_3 solves problem P_3 in $T - 3$ rounds
 - ...
 - algorithm A_T solves problem P_T in 0 rounds
- But P_T is nontrivial, so A_0 cannot exist



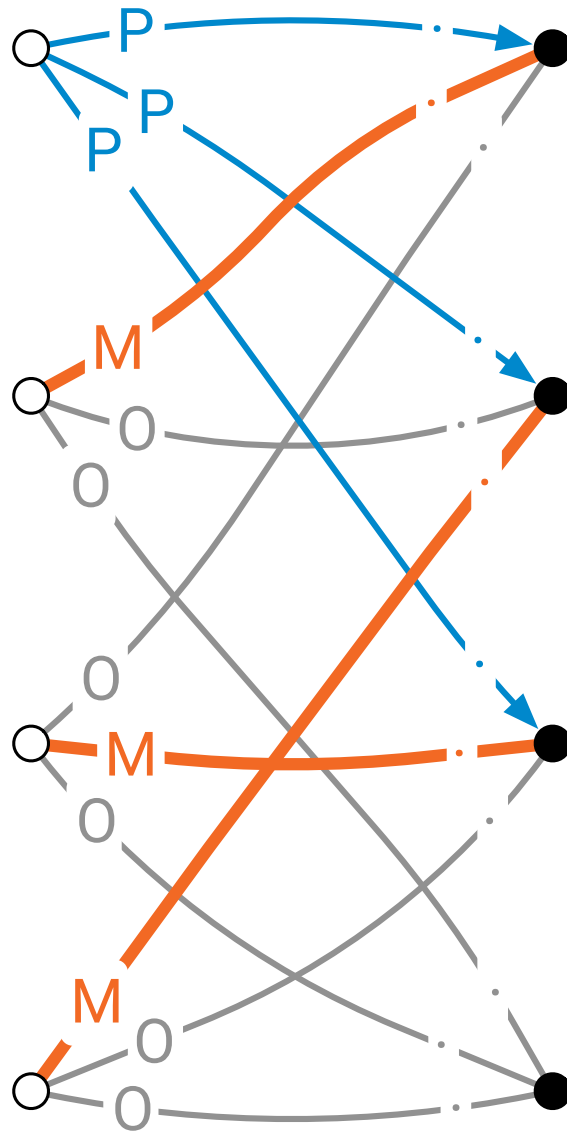
Let's start with P_0 ...

Representation for maximal matchings

white nodes “active”

output one of these:

- $1 \times M$ and $(\Delta-1) \times 0$
- $\Delta \times P$



M = “matched”

P = “pointer to matched”

0 = “other”

black nodes “passive”

accept one of these:

- $1 \times M$ and $(\Delta-1) \times \{P, 0\}$
- $\Delta \times 0$

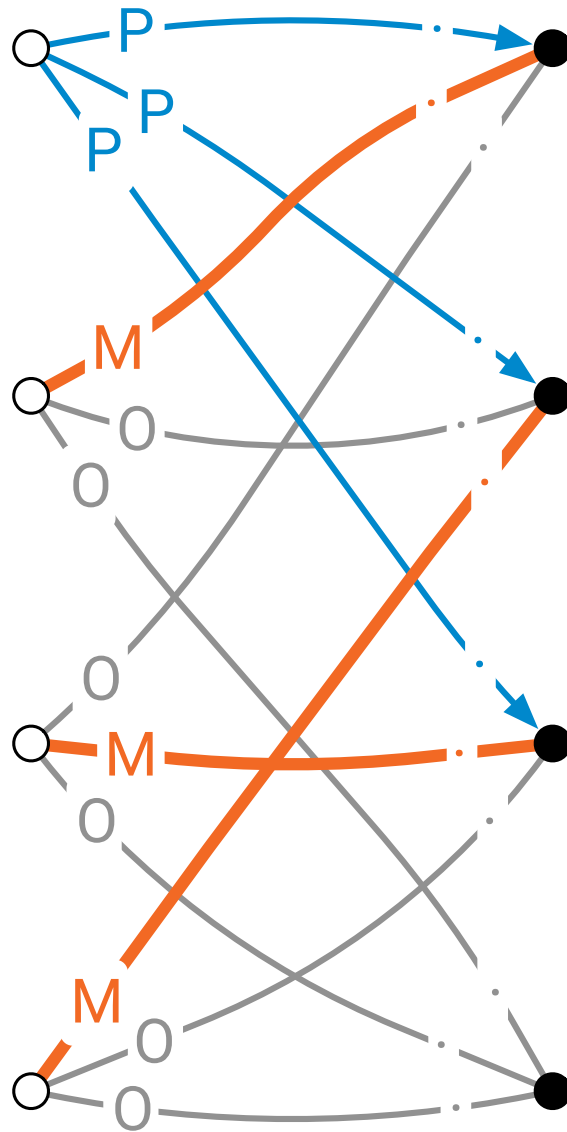
Representation for maximal matchings

white nodes "active"

output one of these:

- $1 \times M$ and $(\Delta-1) \times O$
- $\Delta \times P$

$$W = MO^{\Delta-1} \mid P^{\Delta}$$



M = "matched"

P = "pointer to matched"

O = "other"

black nodes "passive"

accept one of these:

- $1 \times M$ and $(\Delta-1) \times \{P, O\}$
- $\Delta \times O$

$$B = M[PO]^{\Delta-1} \mid O^{\Delta}$$

Parameterized problem family

$$W = \text{MO}^{\Delta-1} \mid \text{P}^{\Delta},$$

$$B = \text{M}[\text{PO}]^{\Delta-1} \mid \text{O}^{\Delta}$$

maximal matching

$$W_{\Delta}(x, y) = \left(\text{MO}^{d-1} \mid \text{P}^d \right) \text{O}^y \text{X}^x,$$

$$B_{\Delta}(x, y) = \left([\text{MX}][\text{POX}]^{d-1} \mid [\text{OX}]^d \right) [\text{POX}]^y [\text{MPOX}]^x,$$

$$d = \Delta - x - y$$

“weak” matching

Main lemma

- Given: \mathbf{A} solves $P(x, y)$ in T rounds
- We can construct: \mathbf{A}' solves $P(x + 1, y + x)$ in $T - 1$ rounds

$$W_{\Delta}(x, y) = \left(\text{MO}^{d-1} \mid \text{P}^d \right) \text{O}^y \text{X}^x,$$

$$B_{\Delta}(x, y) = \left([\text{MX}][\text{POX}]^{d-1} \mid [\text{OX}]^d \right) [\text{POX}]^y [\text{MPOX}]^x,$$

$$d = \Delta - x - y$$

Putting things together

Maximal matching in $o(\Delta)$ rounds

→ “ $\Delta^{1/2}$ matching” in $o(\Delta^{1/2})$ rounds

→ $P(\Delta^{1/2}, 0)$ in $o(\Delta^{1/2})$ rounds

→ $P(O(\Delta^{1/2}), o(\Delta))$ in 0 rounds

→ contradiction

What we really care about

k-matching:
select at most
k edges per node

Apply round
elimination
 $o(\Delta^{1/2})$ times

Putting things together

Proof technique does not work directly with unique IDs

- Basic version:
 - deterministic lower bound, *port-numbering model*
- Analyze what happens to local failure probability:
 - *randomized* lower bound, port-numbering model
- With randomness you can construct unique identifiers w.h.p.:
 - randomized lower bound, *LOCAL model*
- Fast deterministic → very fast randomized
 - stronger *deterministic* lower bound, LOCAL model

Main results

Maximal matching and **maximal independent set** cannot be solved in

- $o(\Delta + \log \log n / \log \log \log n)$ rounds with randomized algorithms
- $o(\Delta + \log n / \log \log n)$ rounds with deterministic algorithms

Lower bound for MM
implies a lower bound
for MIS

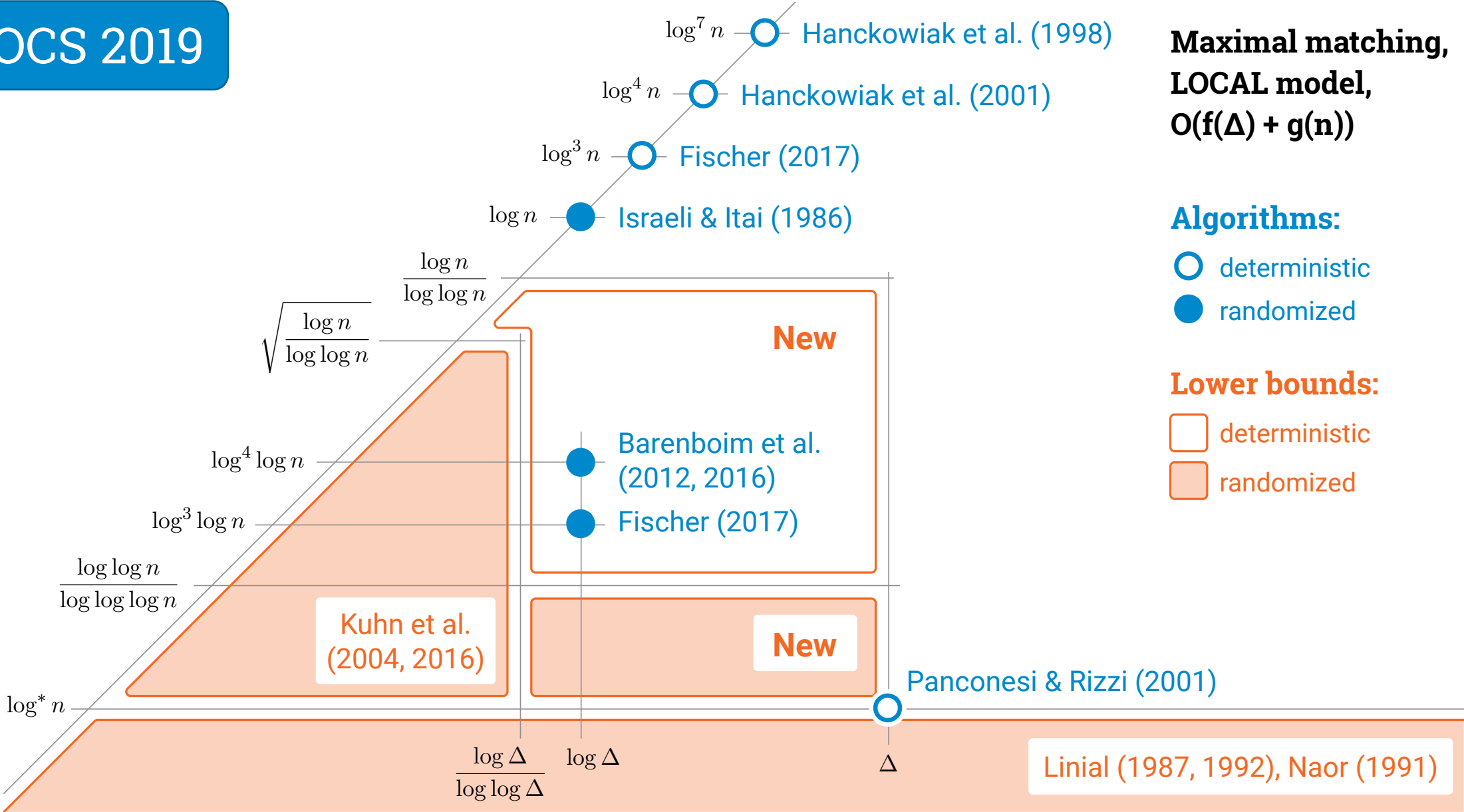
**Maximal matching,
LOCAL model,
 $O(f(\Delta) + g(n))$**

Algorithms:

- deterministic
- randomized

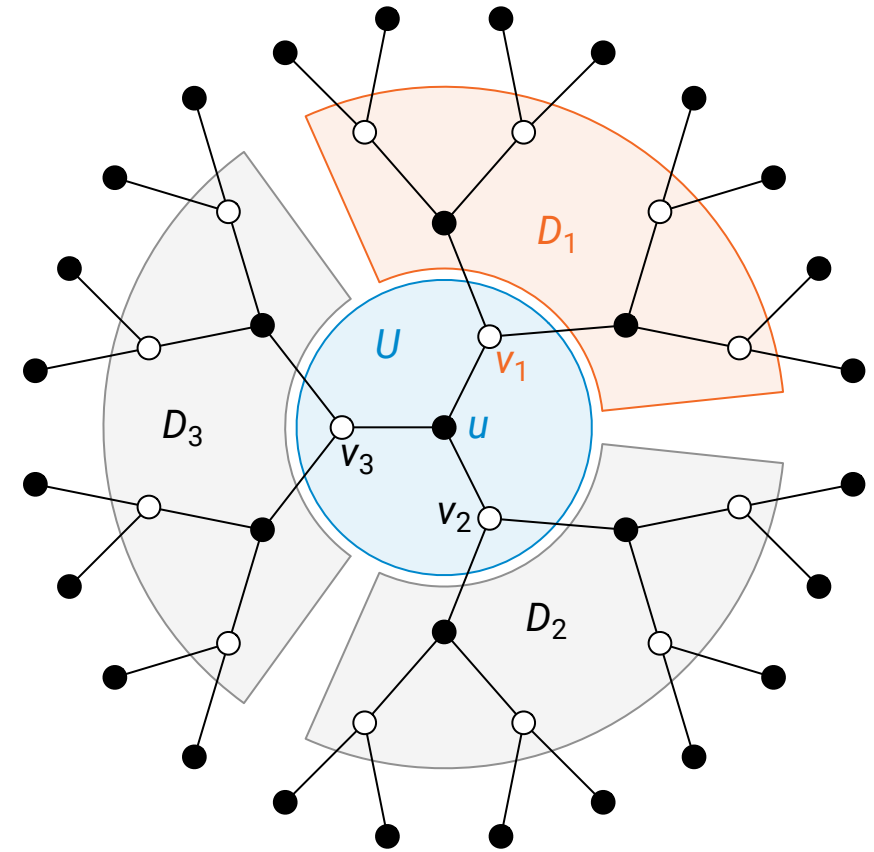
Lower bounds:

- deterministic
- randomized



Summary

- Round elimination technique
- Locality lower bounds for a wide range of problems:
 - symmetry breaking in cycles
 - symmetry breaking in regular trees
 - algorithmic Lovász local lemma
 - **maximal matching**, maximal independent set ...
- And for a wide range of localities:
 - $\Omega(\log^* n)$, $\Omega(\log \log n)$, $\Omega(\log n)$, $\Omega(\log^* \Delta)$, $\Omega(\Delta)$...



Open questions

- Lower bounds for **volume complexity**?
 - volume lower bounds for **sinkless orientation**?
- Lower bounds for problems related to **graph coloring**?
 - when is **partial/defective coloring** “easy” and when is it “hard”?
 - nontrivial lower bounds for **$(\Delta+1)$ -coloring**?
- Exactly when do we get **fixed points** and why?

