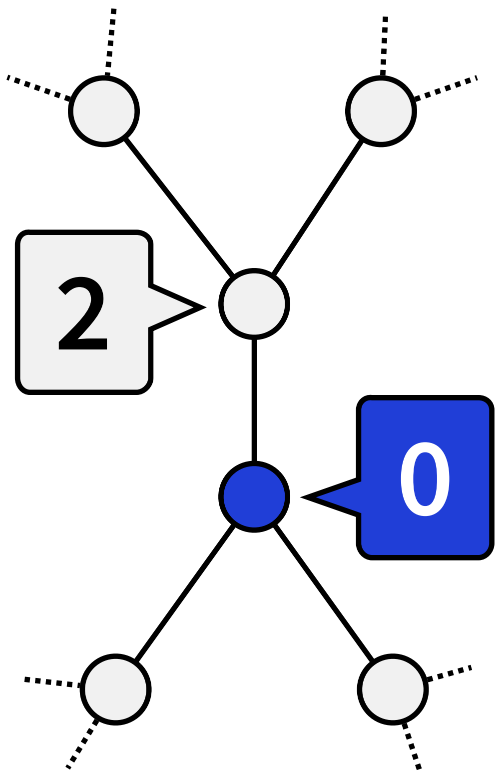
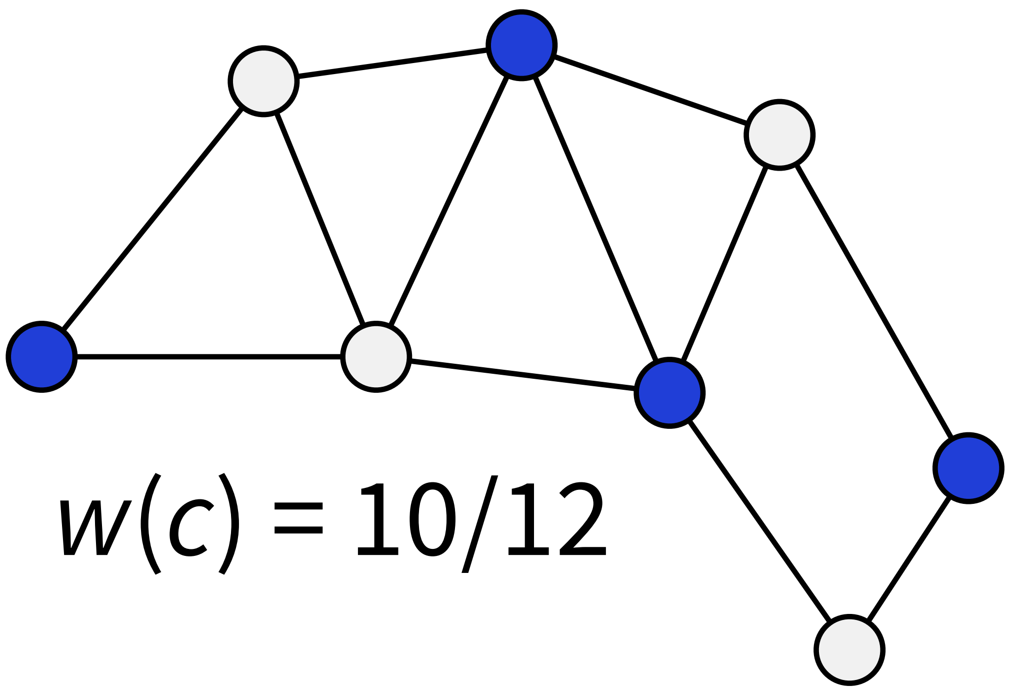
Large Cuts with Local Algorithms

Jukka Suomela
research.ics.aalto.fi/da

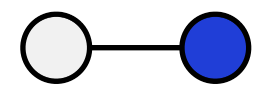
Helsinki Algorithms Seminar
28 March 2014



Cut

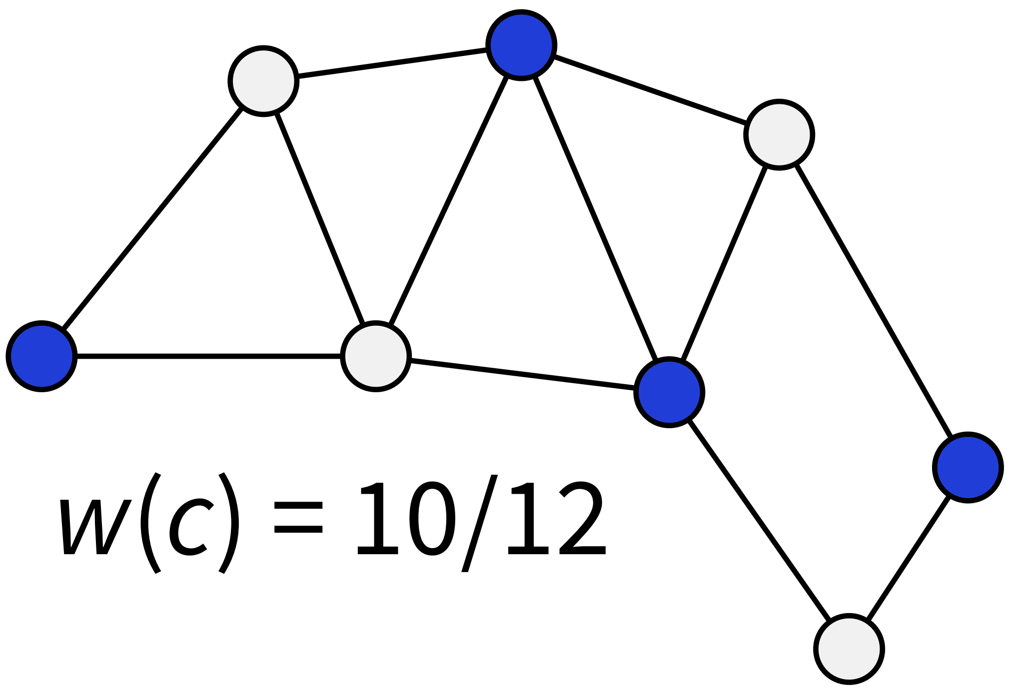


c: V → {○, ●}

Weight $w(c)$ = fraction of cut edges 

$0 \leq w(c) \leq 1$

Cut



c: V → {○, ●}

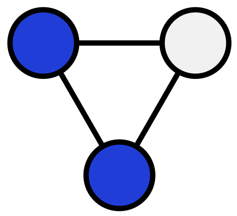
Weight $w(c)$ = fraction of cut edges ○—●

Uniform random cut: $E[w(c)] = 1/2$

Can we do better than 1/2?

Larger than 1/2?

- **Not much larger in general graphs**
  - complete graph: $w(c) \leq 1/2 + O(1/n)$
  - triangles bad
  - high degrees bad

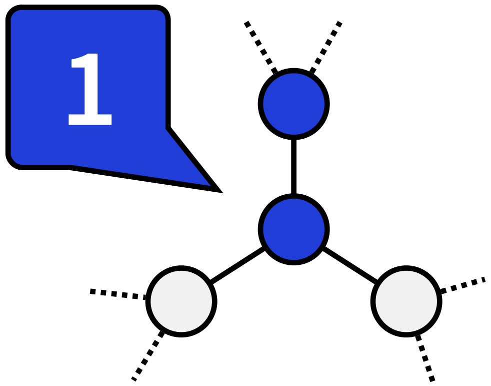


- **Focus: *d*-regular triangle-free graphs**

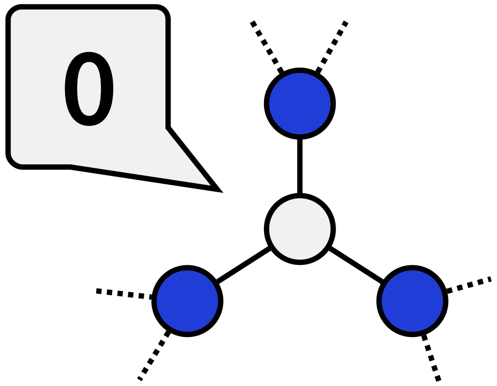
Cuts in *d*-regular triangle-free graphs

- **Shearer (1992):**
  $w(c) \geq 1/2 + 0.177/\sqrt{d}$ $d = 4$: $w(c) \geq 0.594$
- **Hirvonen, Rybicki, Schmid, S.:**
  $w(c) \geq 1/2 + 0.281/\sqrt{d}$ $d = 4$: $w(c) \geq 0.641$
- **Proof: simple randomised distributed algorithms!**

General idea

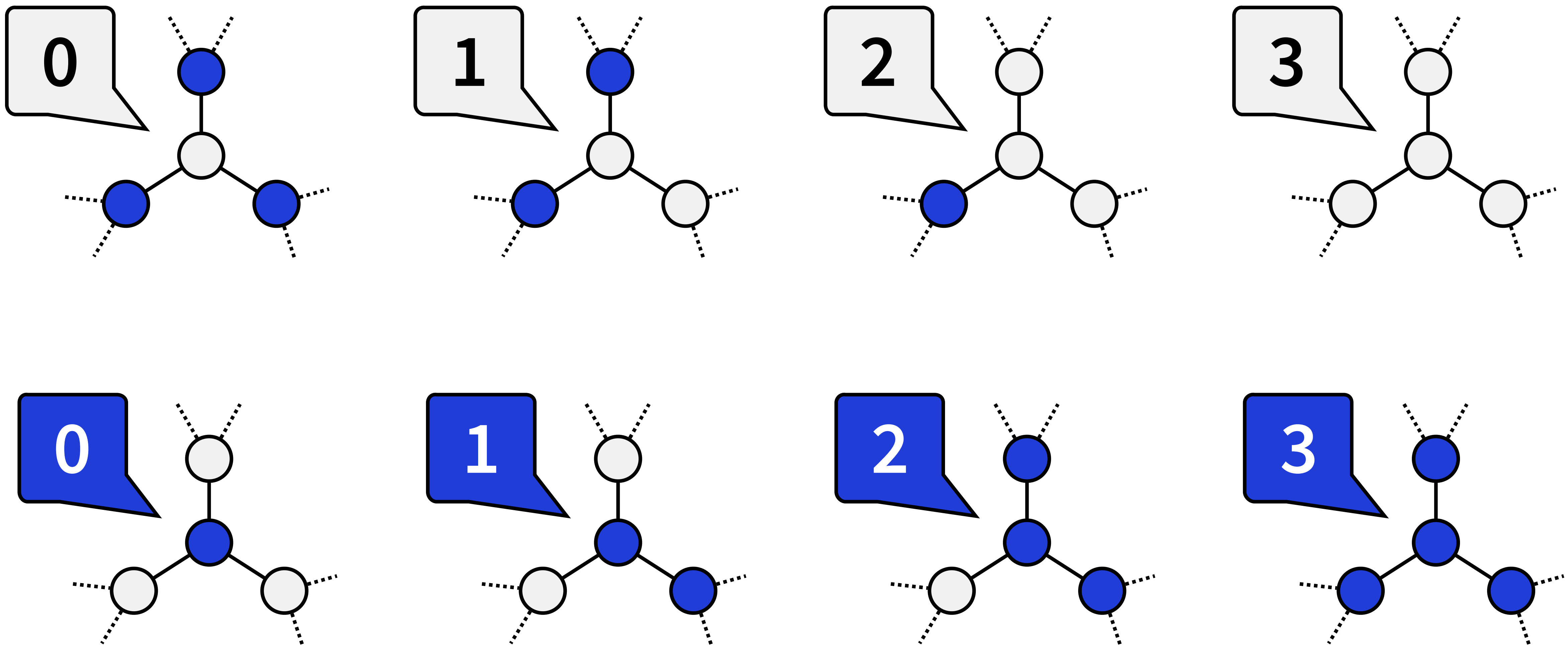


- **Pick a uniform random cut**

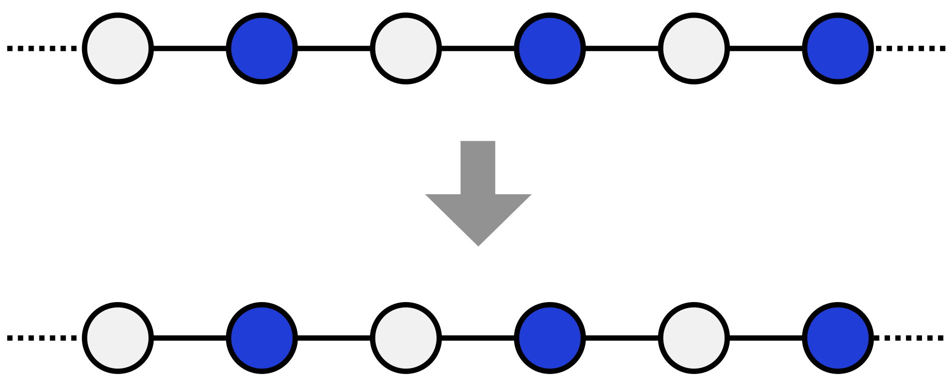


- **Fix it locally**
- **Each node only looks at:**
  - its own colour
  - how many neighbours have the same colour

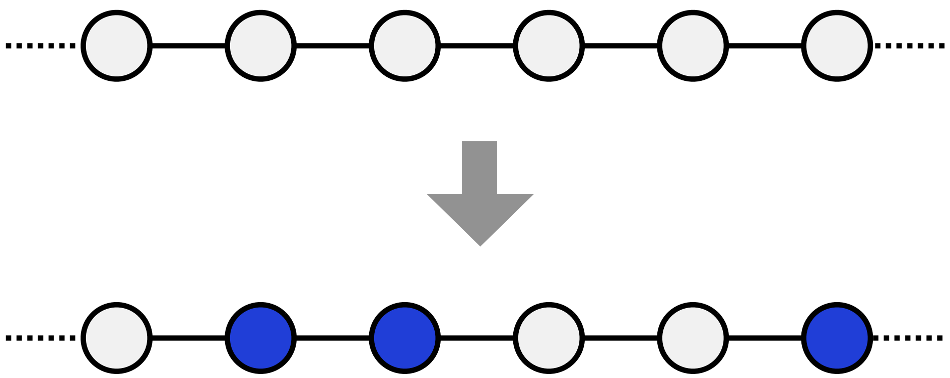
Only $2d+2$ cases



Shearer's algorithm

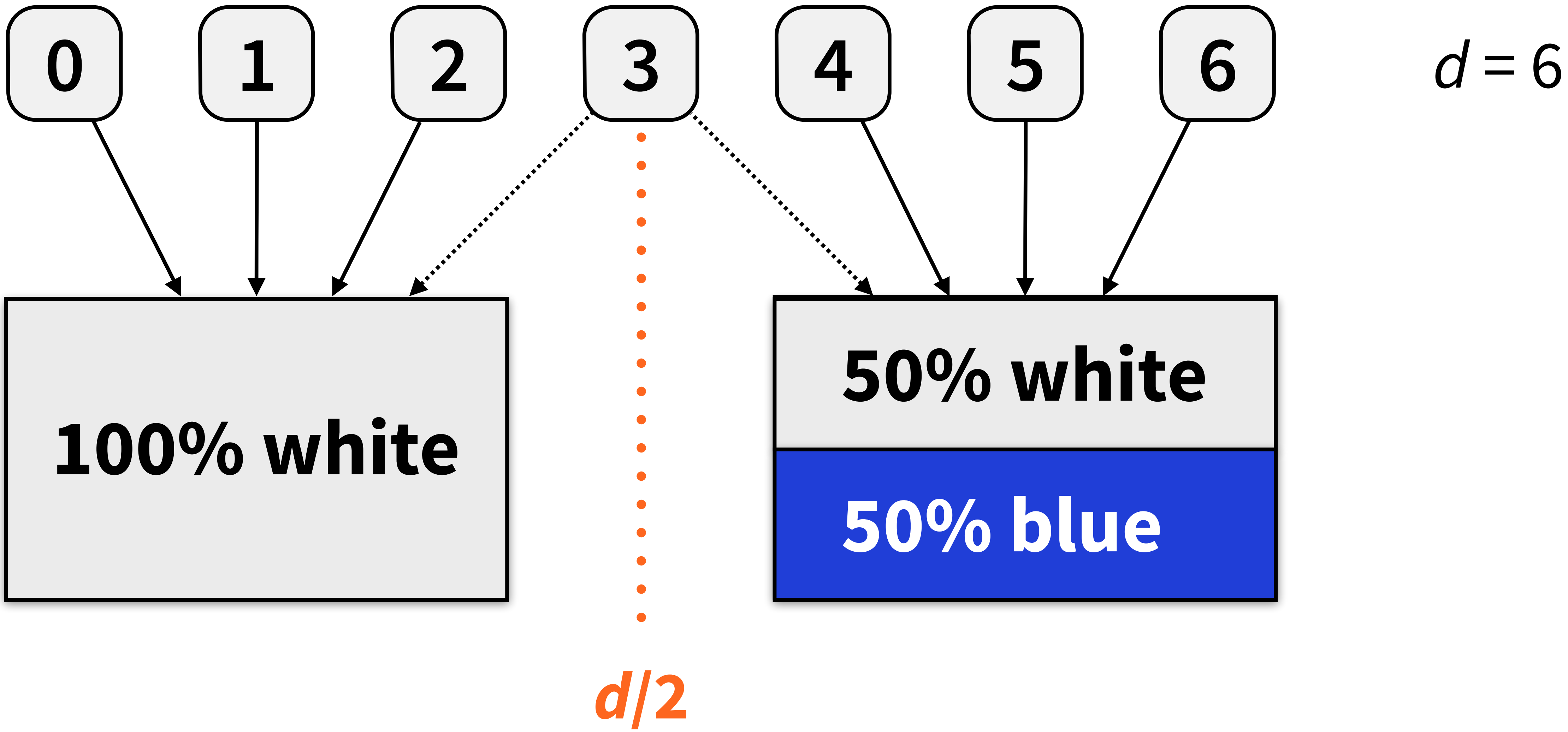


we were lucky,
do nothing

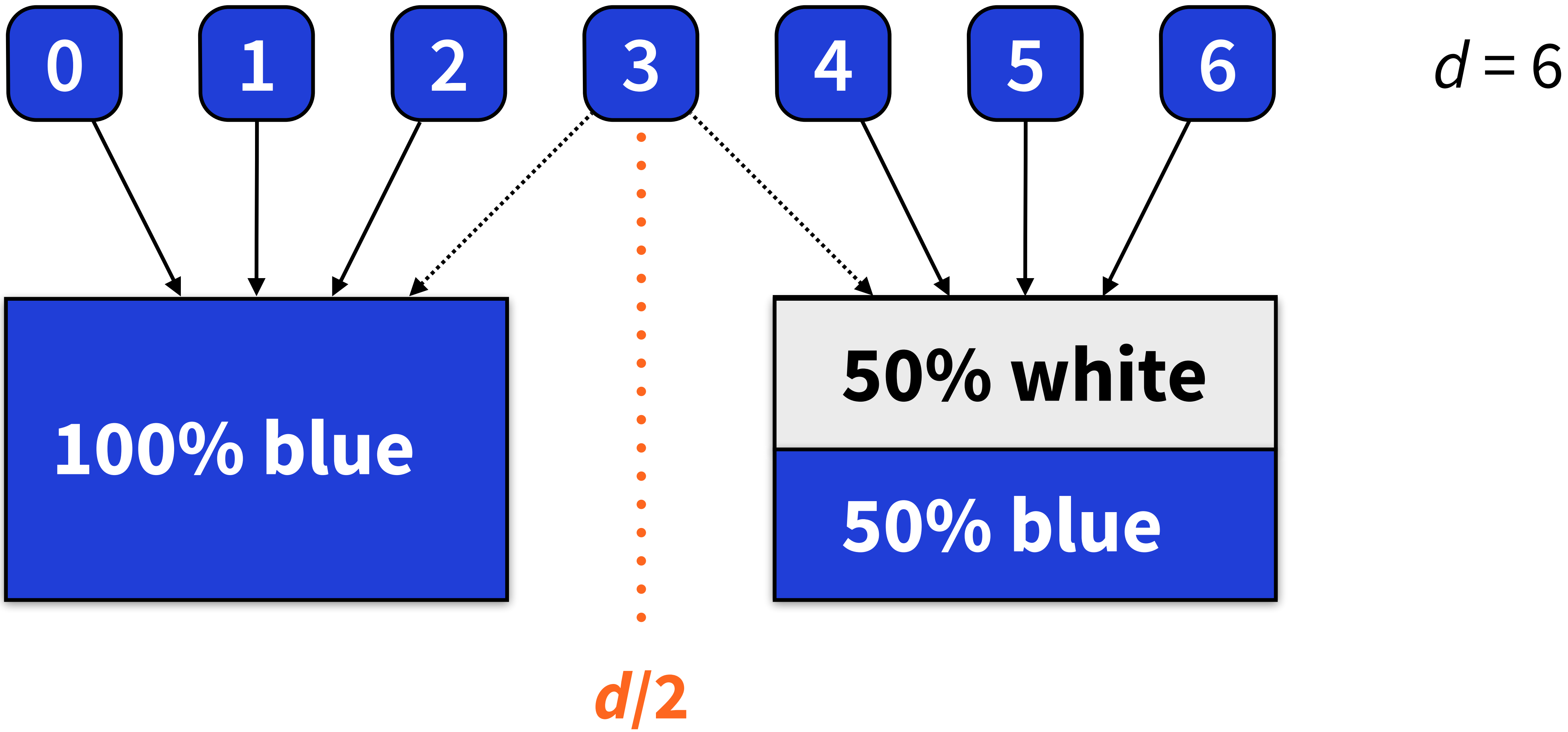


looks bad,
*try again **<u>once</u>***

Shearer's algorithm



Shearer’s algorithm



Better *and* simpler!

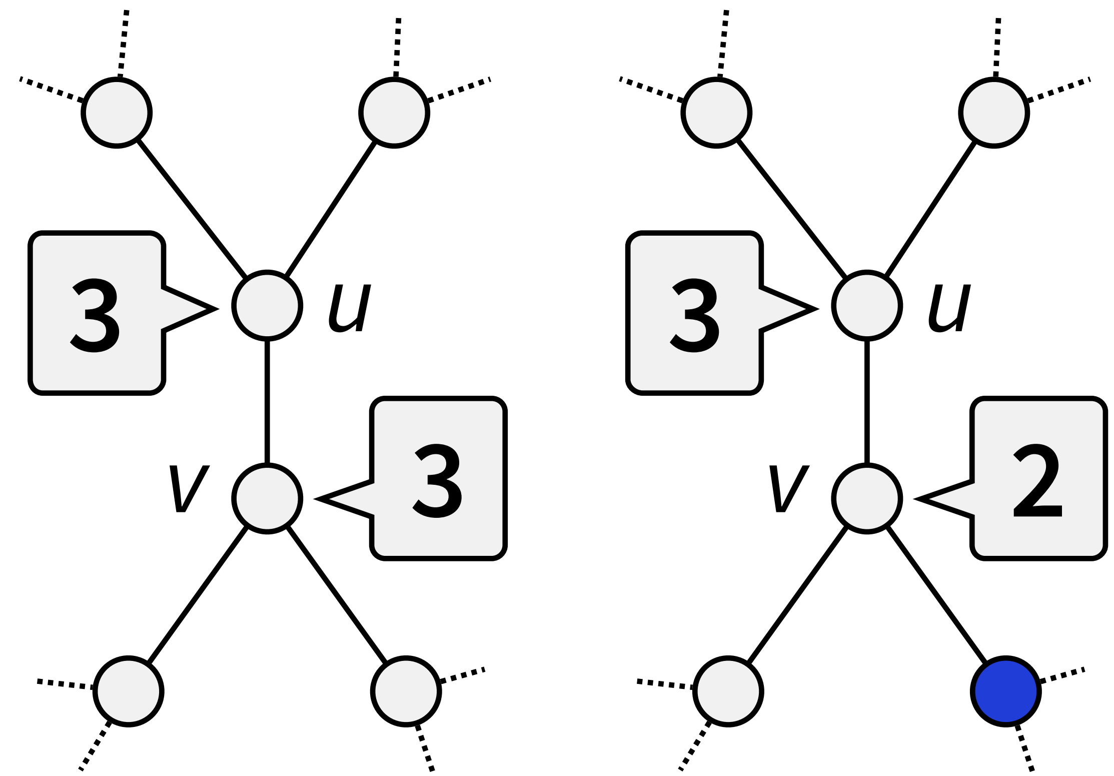
- **Shearer: *randomised* fixing of random cuts**
- **New: *deterministic* fixing of random cuts**
- **There are only 2^{2d+2} possible algorithms!**

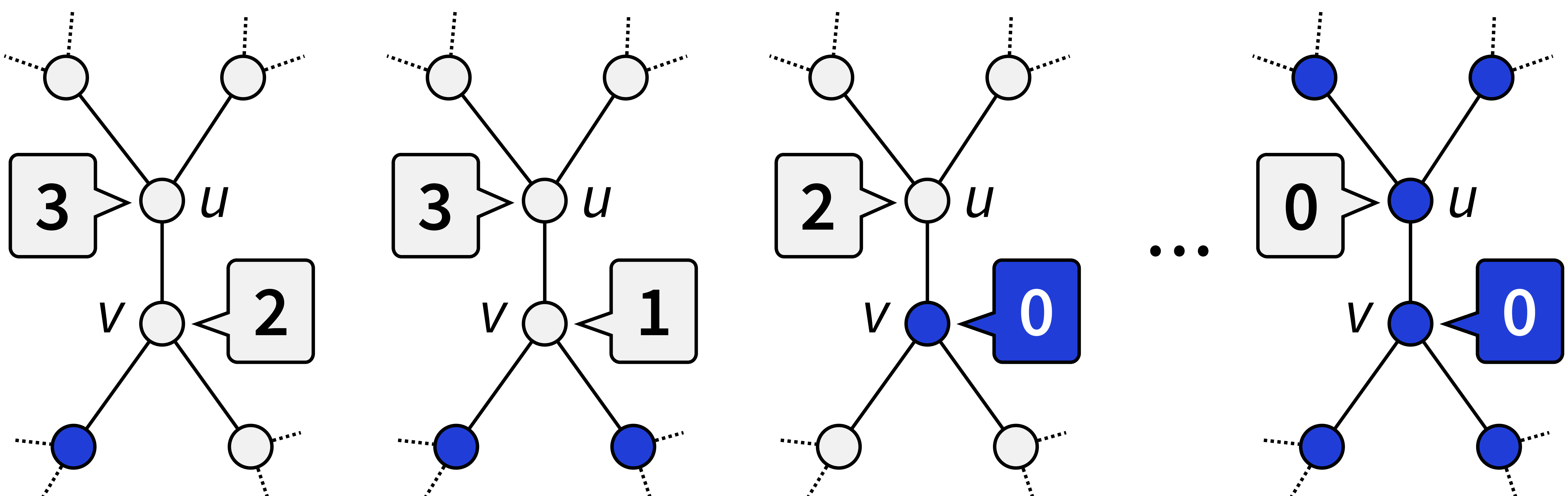
A: { 0 1 … d **0** **1** … **d** } → { ○, ● }

Evaluation of algorithm candidates

- **Fix an algorithm *A***
- **Any *d*-regular triangle-free graph *G*, any edge *e* = {*u*, *v*} in *G***
- **Pr[e is a cut edge] ?**
  - independent of *G* and *e*, only depends on *A*

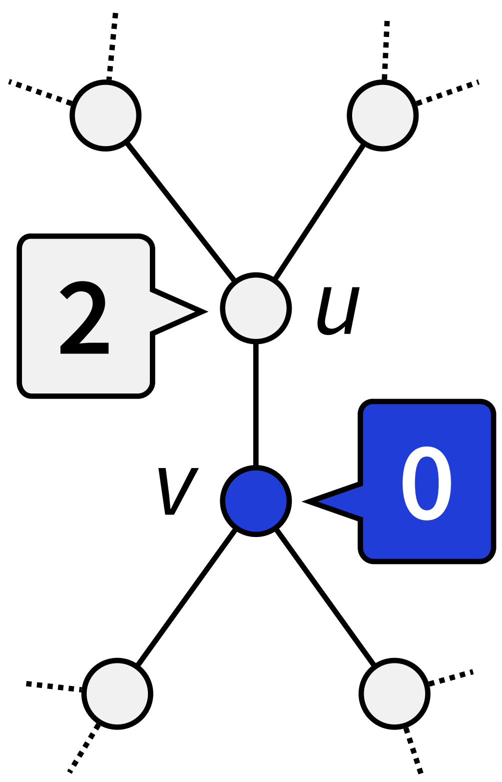
2^{2d} cases



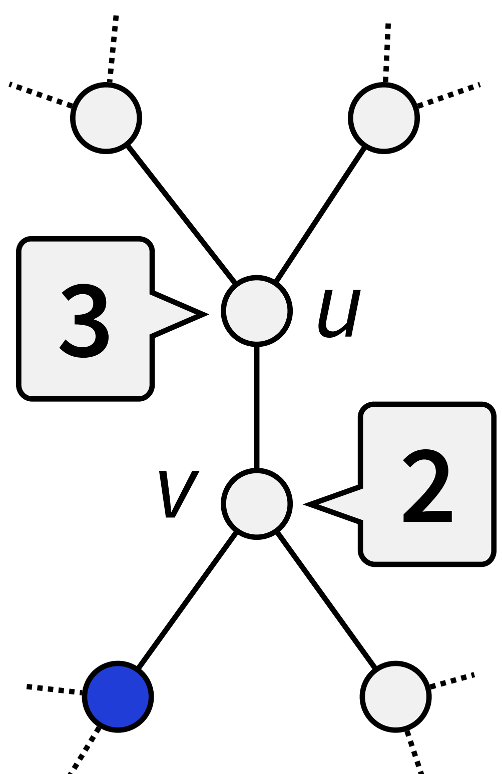


2^{2d} cases

$\{u, v\}$ is a cut edge
iff $A($ 3 $) \neq A($ 0 $)$

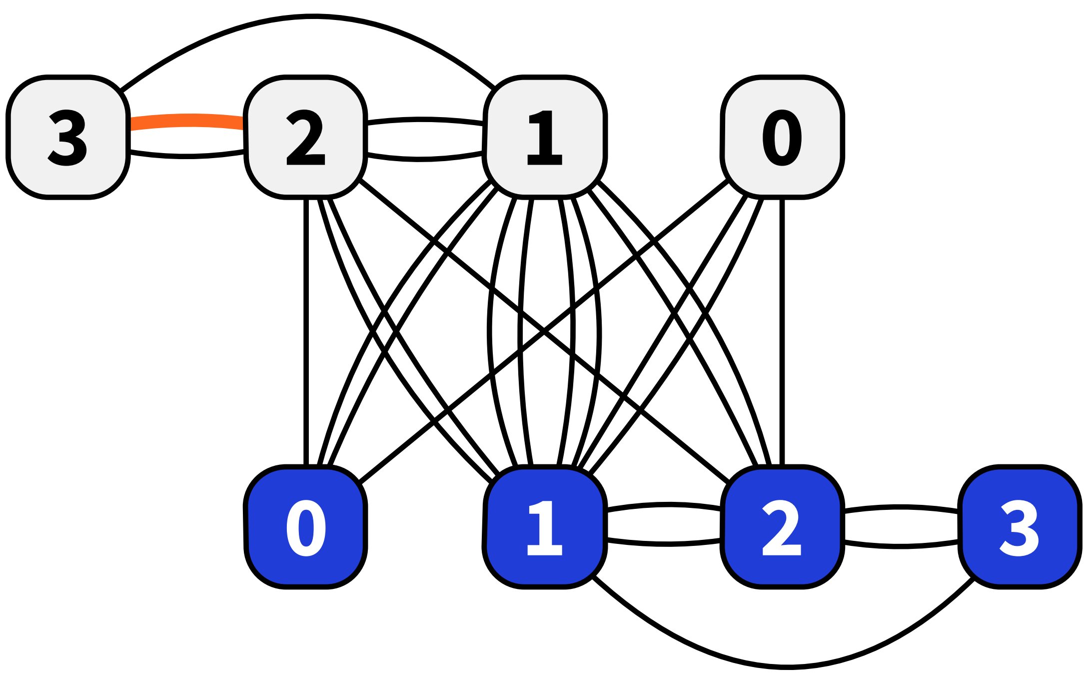


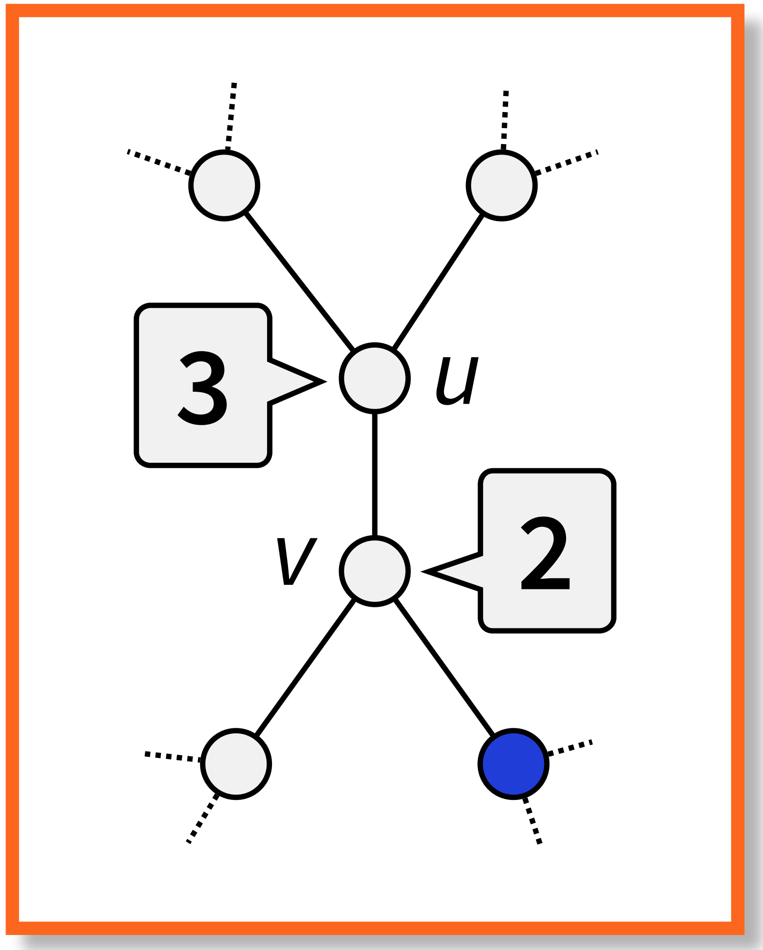
2^{2d} cases



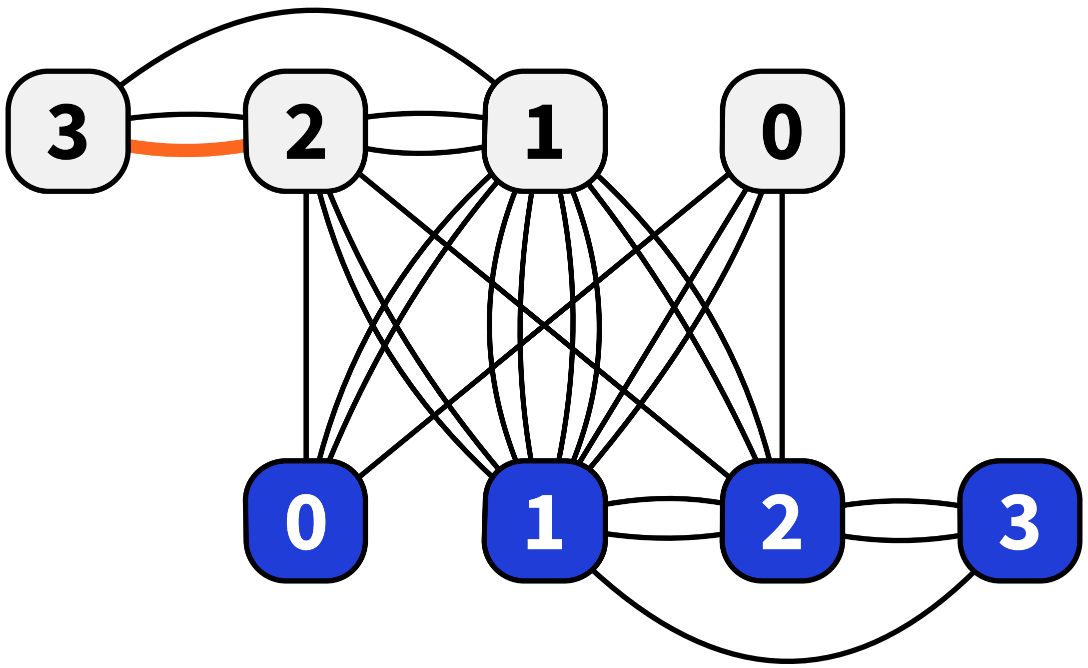
$\{u, v\}$ is a cut edge
iff $A(3) \neq A(2)$

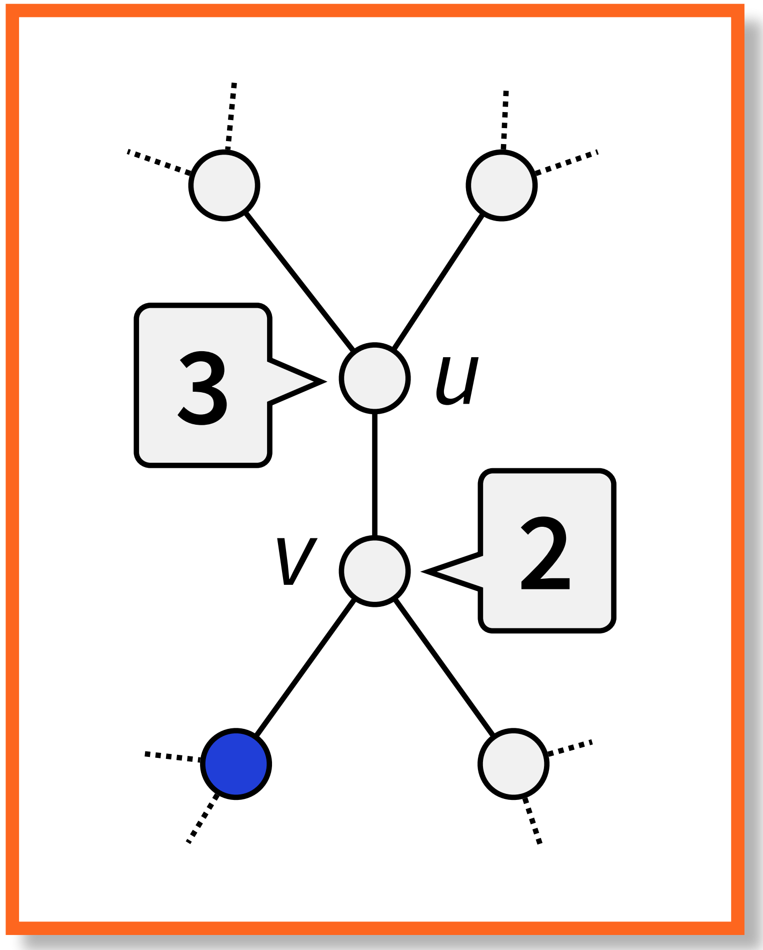
2^{2d} cases



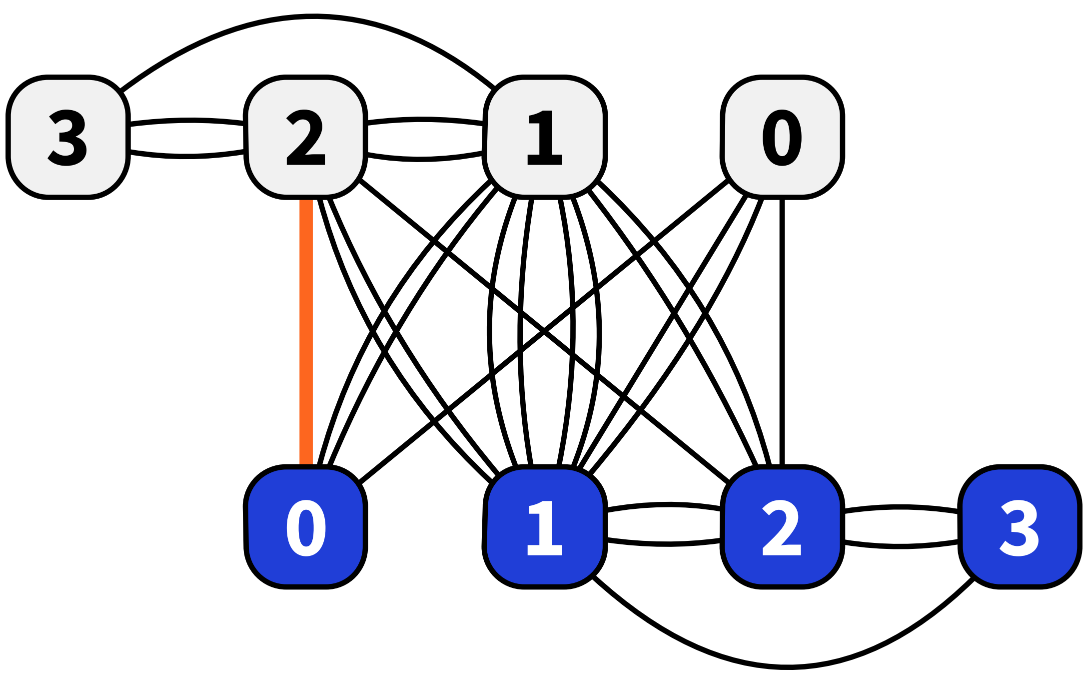


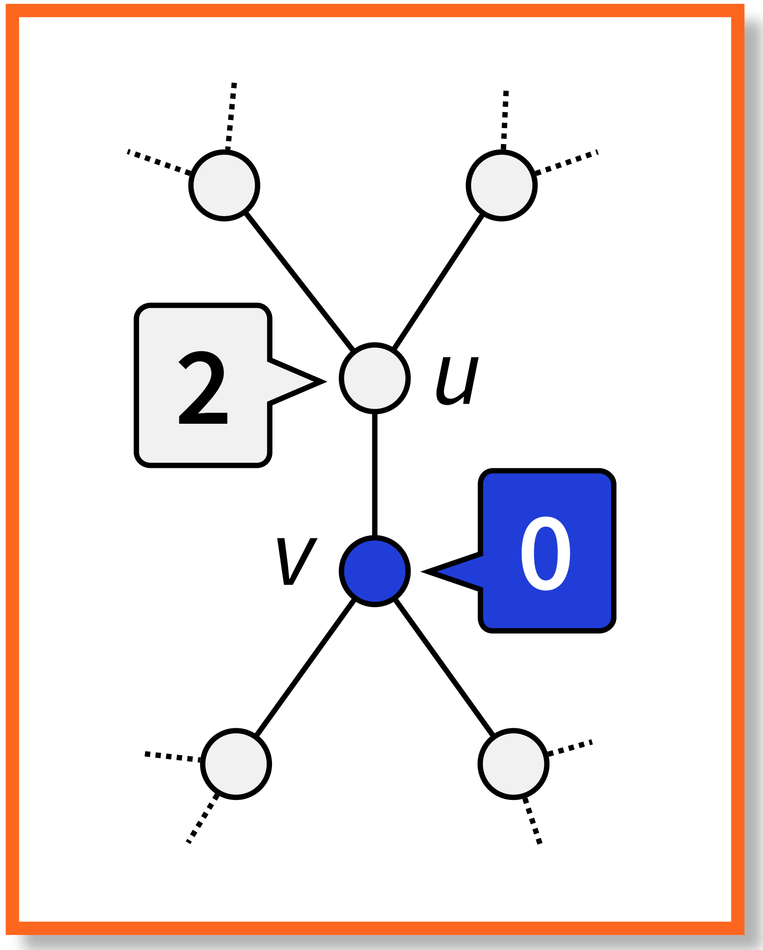
2^{2d} cases



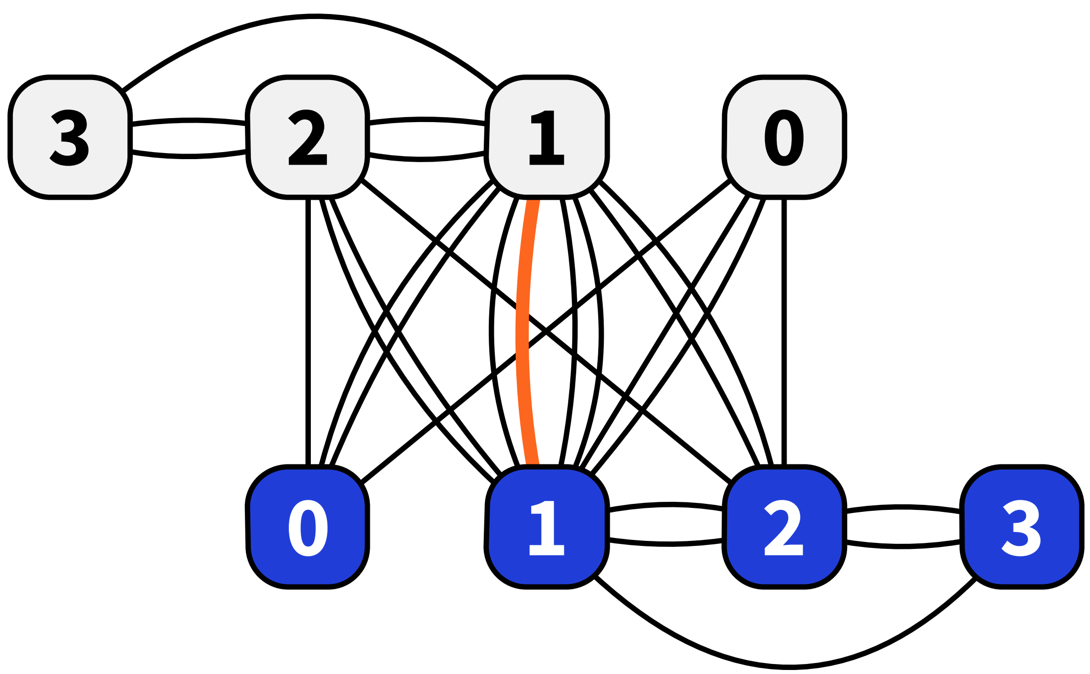


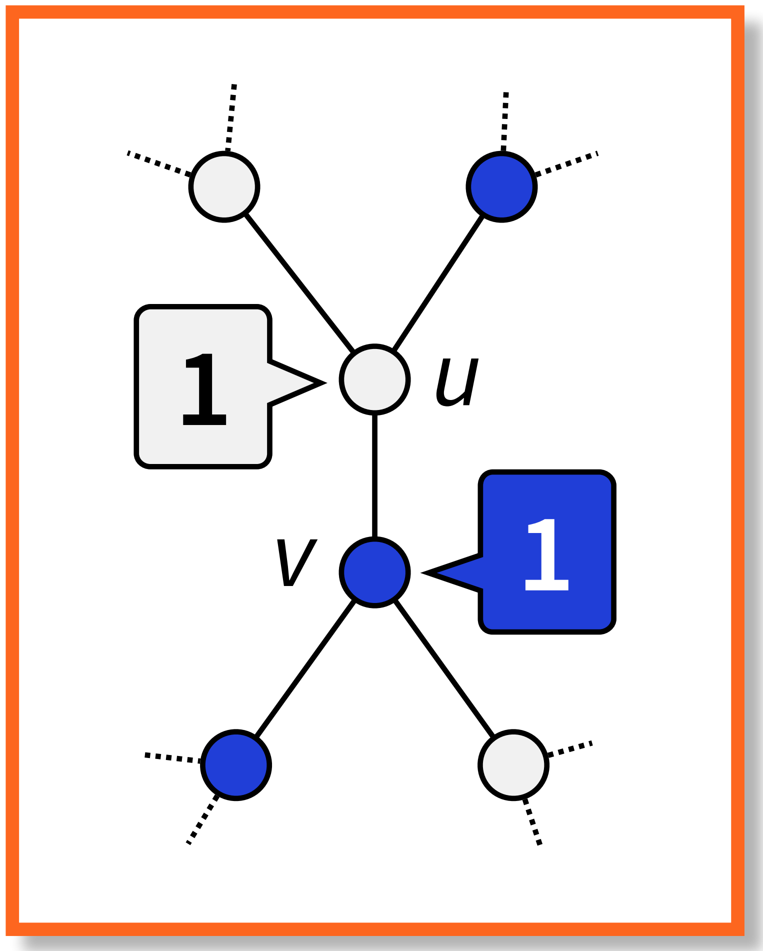
2^{2d} cases



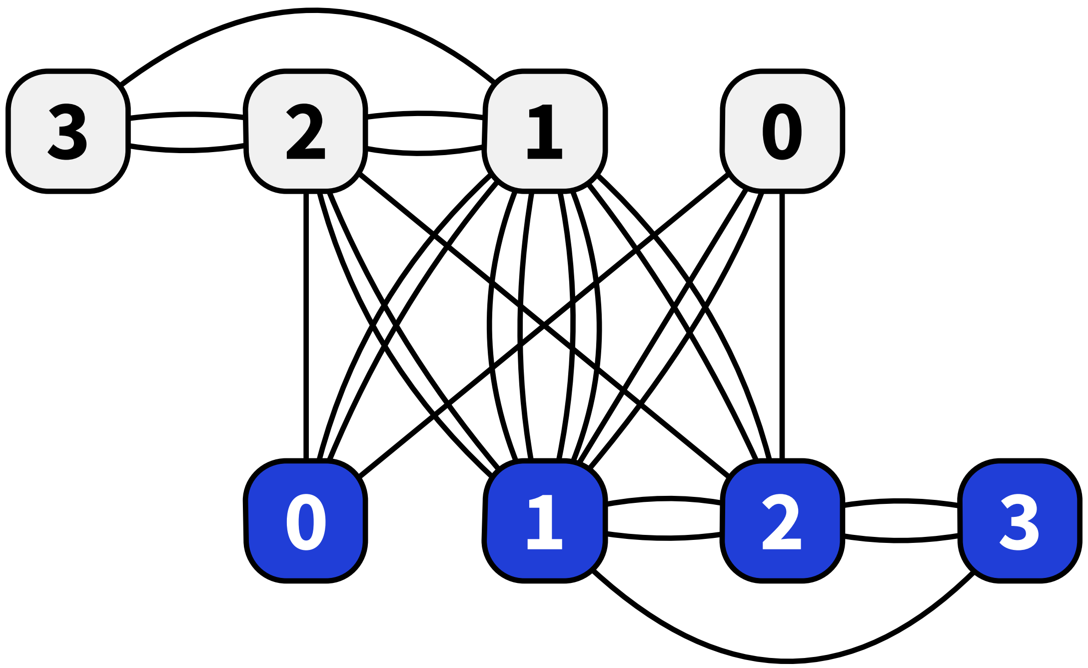


2^{2d} cases



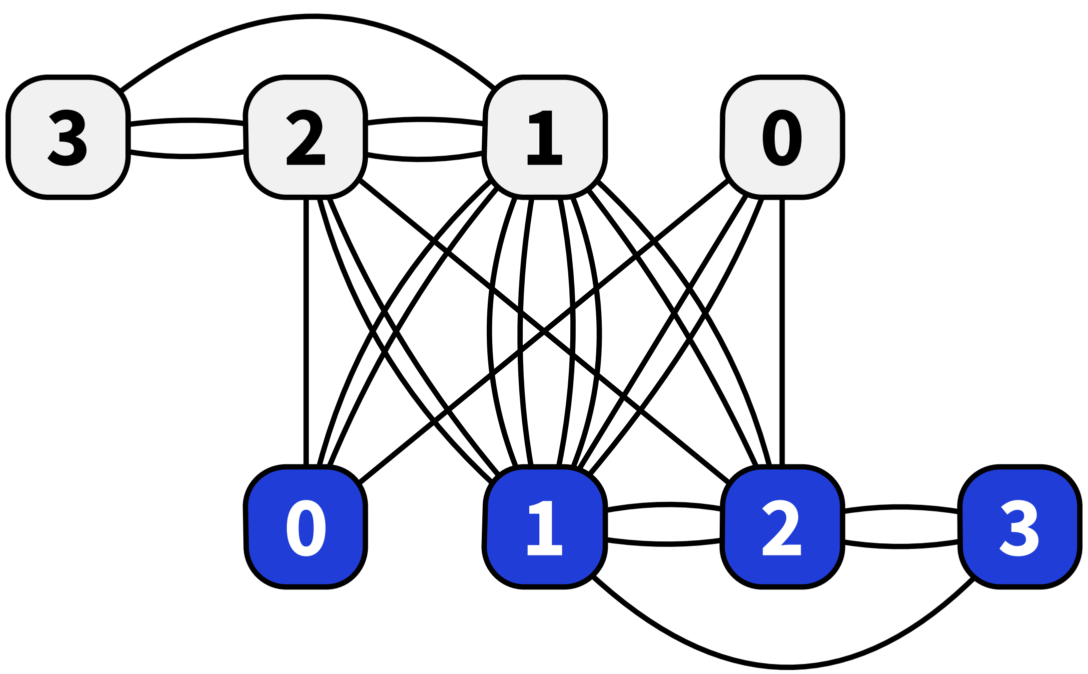


Neighbourhood graph

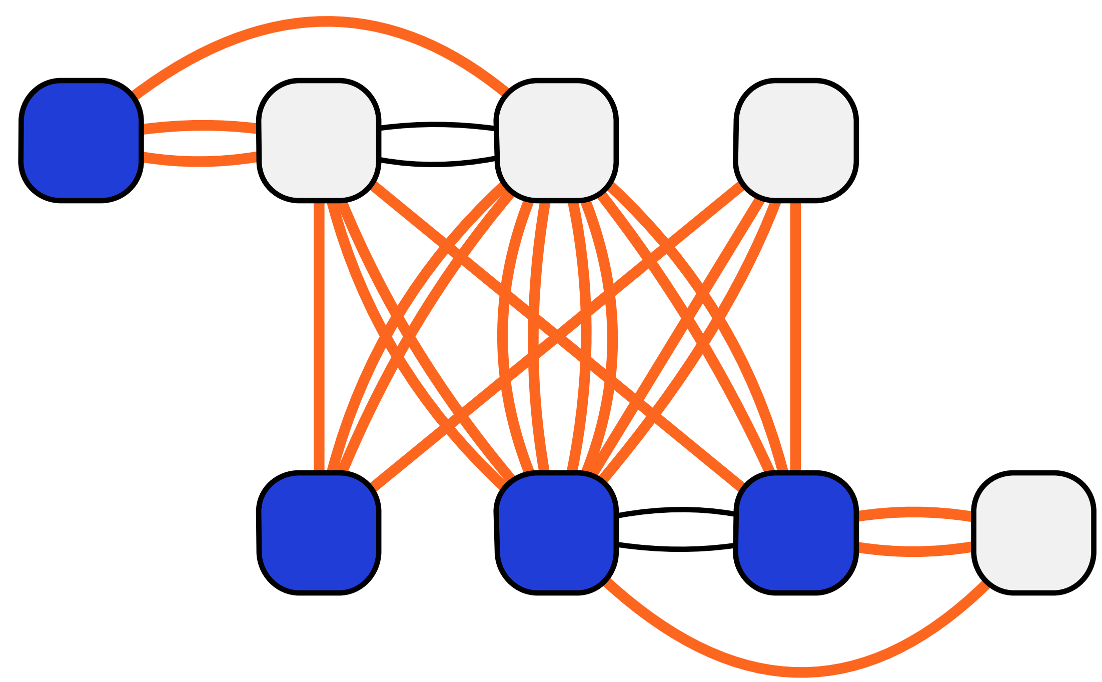


neighbourhoods

Largest cut = optimal algorithm

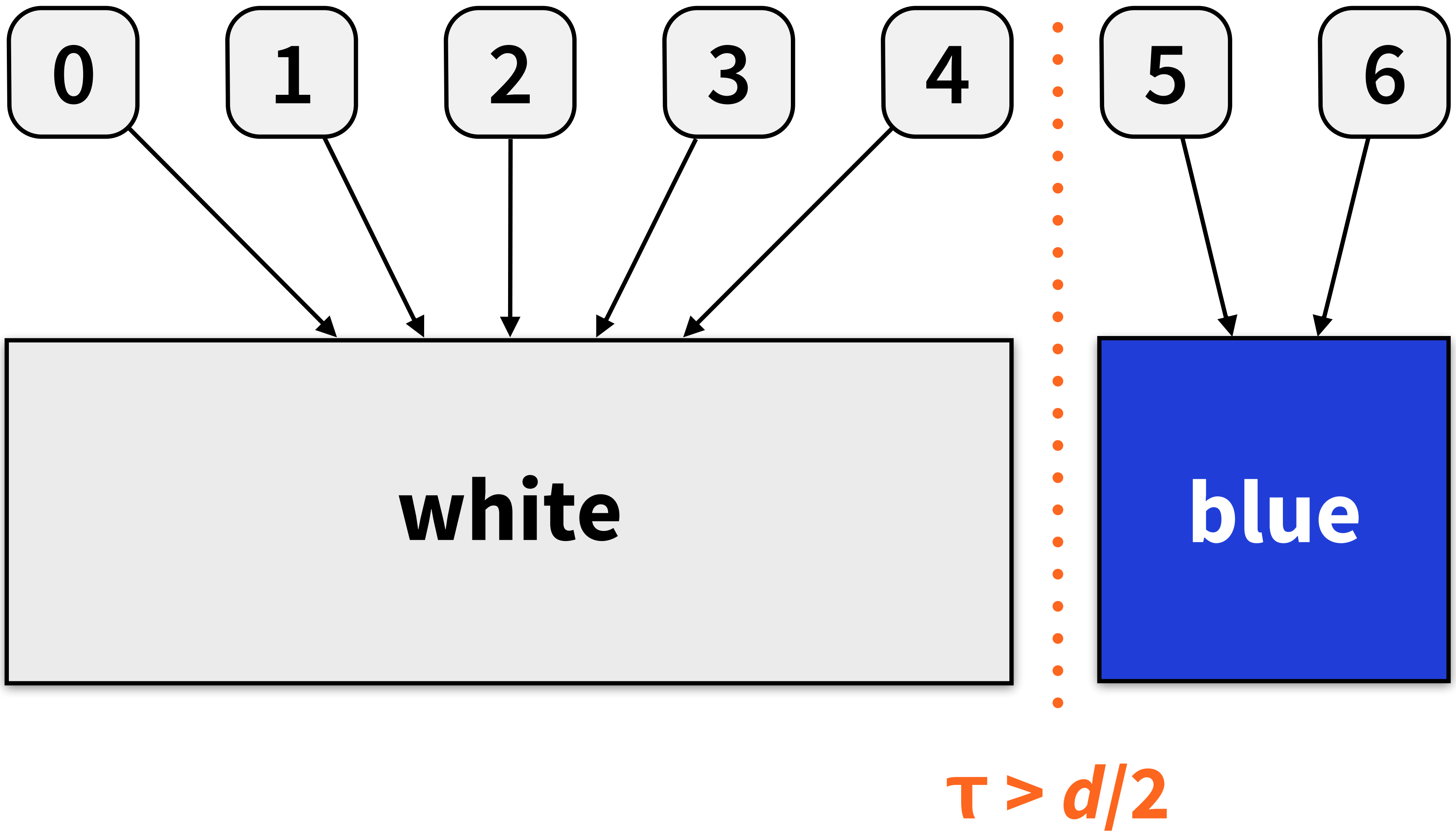


neighbourhoods



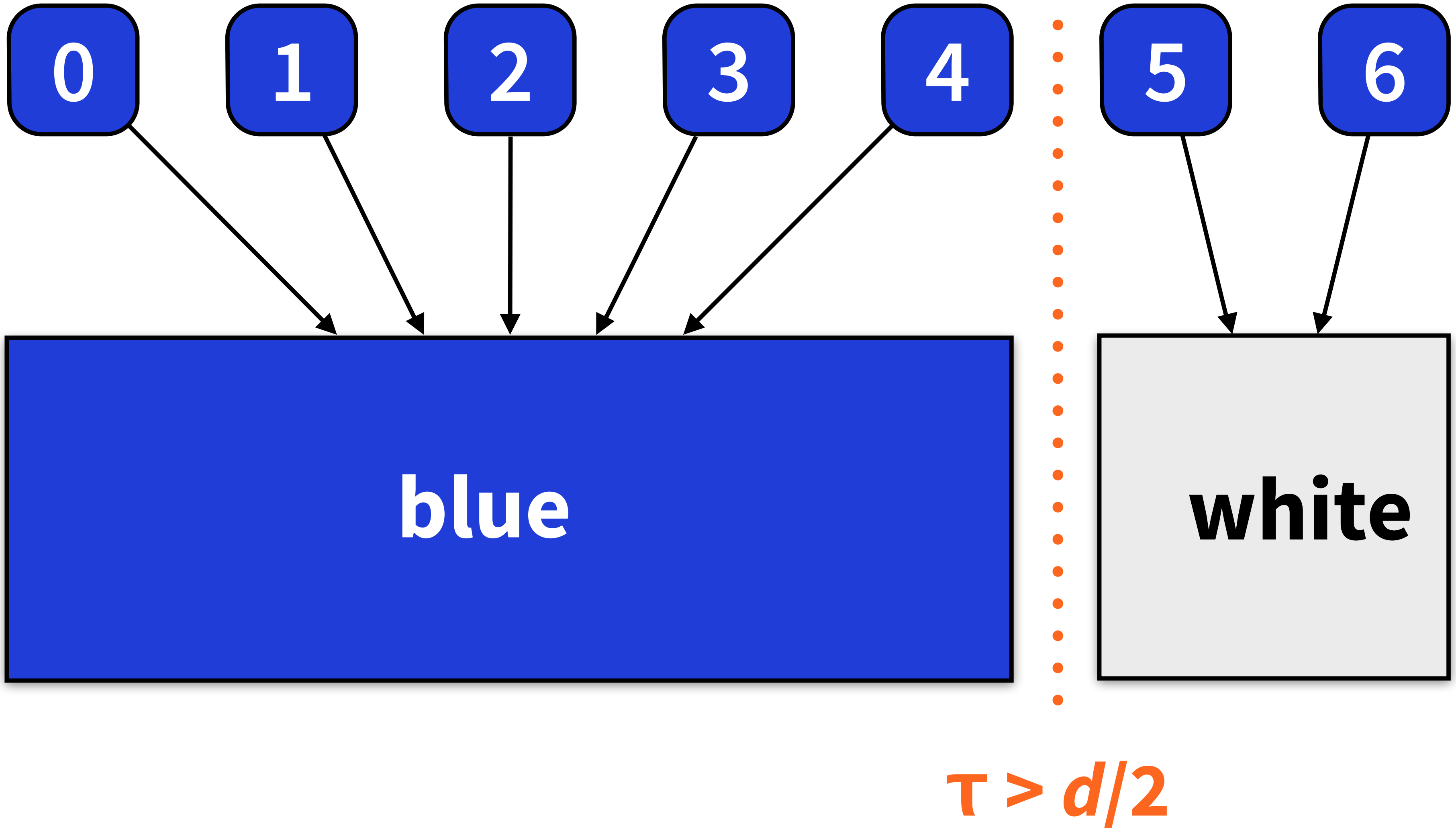
output for each case

Small *d*: optimal algorithms



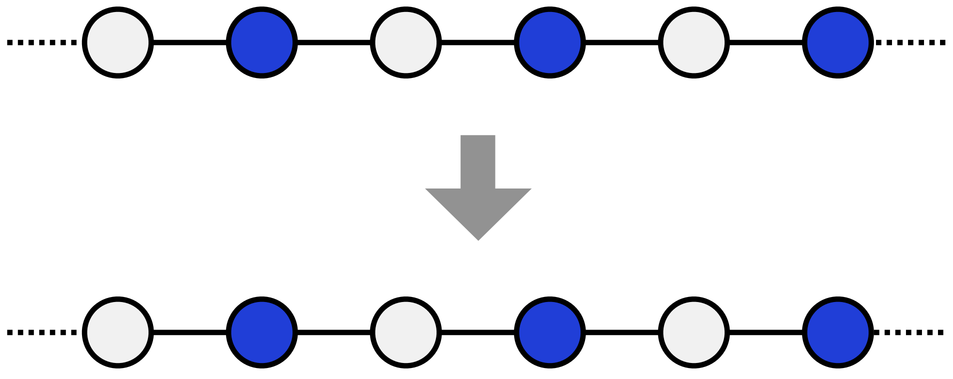
$d = 6$

Small *d*: optimal algorithms

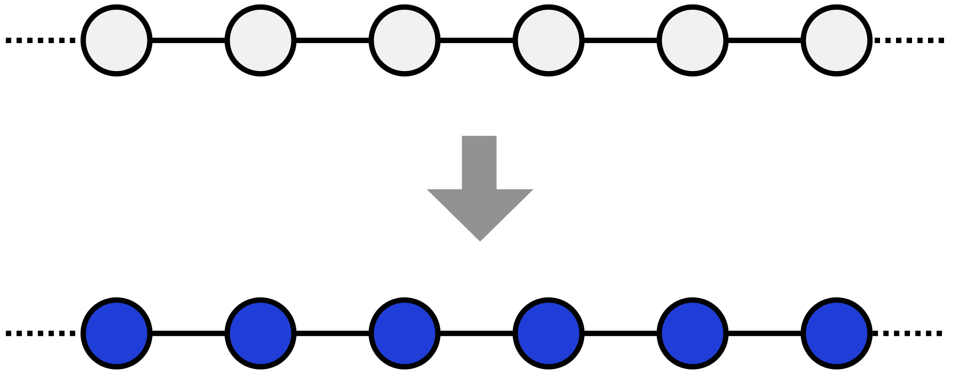


$d = 6$

Does it make any sense??

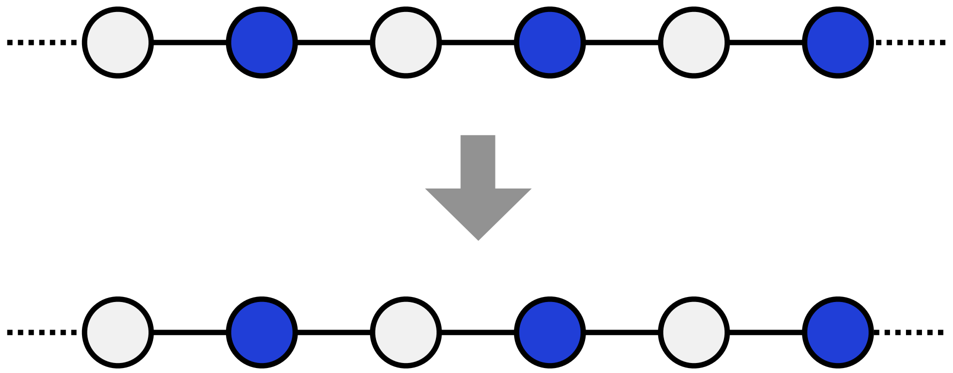


we were lucky,
do nothing

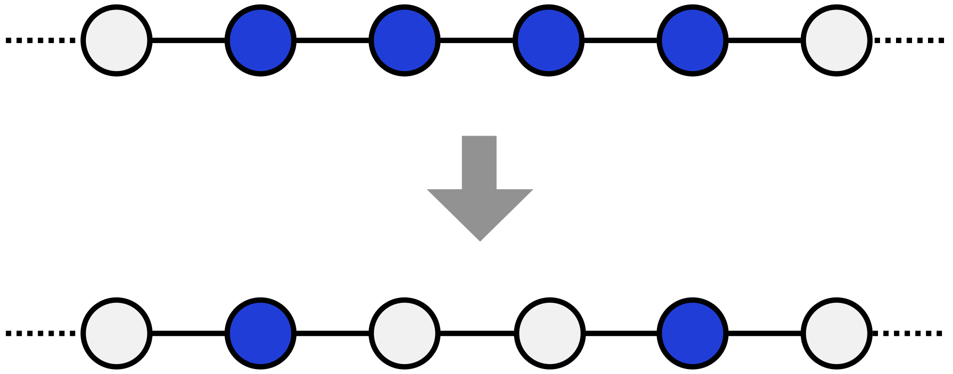


looks bad,
flip

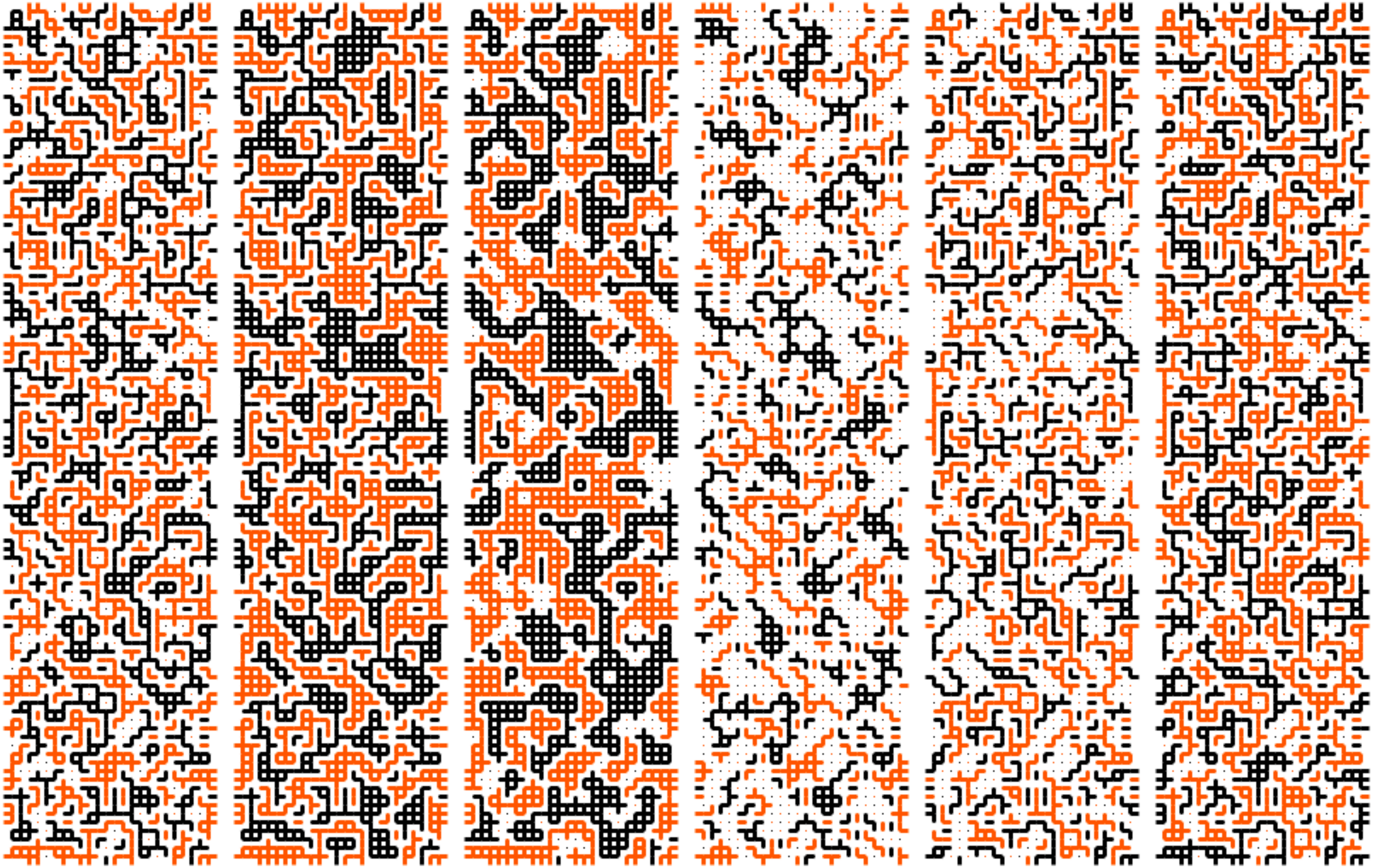
More typical scenario



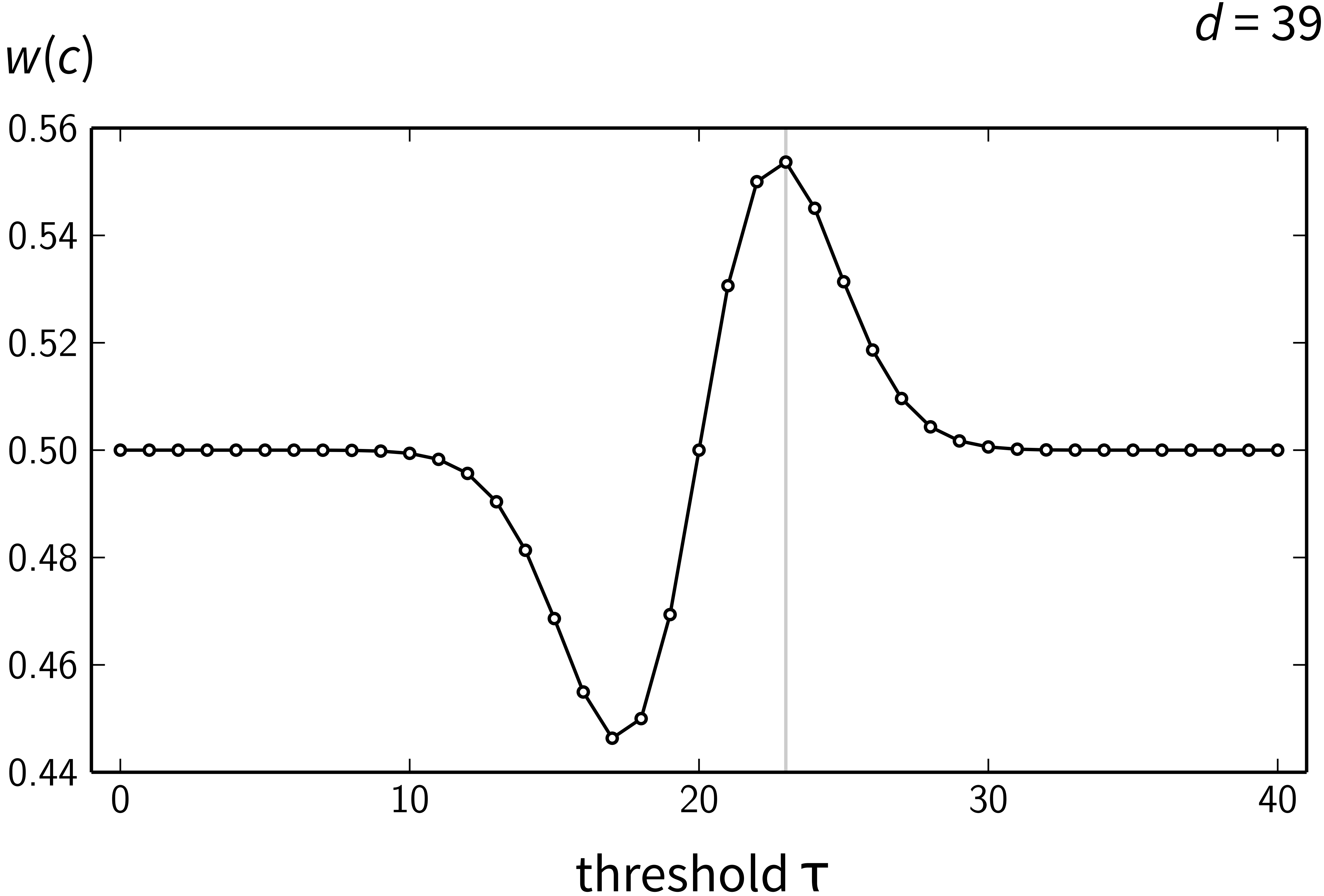
we were lucky,
do nothing



looks bad,
flip



$\tau = 0$	$\tau = 1$	$\tau = 2$	**$\tau = 3$**	$\tau = 4$	$\tau = 5$
0.500		0.359	**0.641**		0.500


d = 39
w(c)
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0
10
20
30
40
threshold τ

Cuts in *d*-regular triangle-free graphs

- **Shearer (1992):**
  $w(c) \geq 1/2 + 0.177/\sqrt{d}$ $d = 4$: $w(c) \geq 0.594$

- **Hirvonen, Rybicki, Schmid, S.:**
  $w(c) \geq 1/2 + 0.281/\sqrt{d}$ $d = 4$: $w(c) \geq 0.641$

- **Demo:**
  users.ics.aalto.fi/suomela/local-maxcut-demo