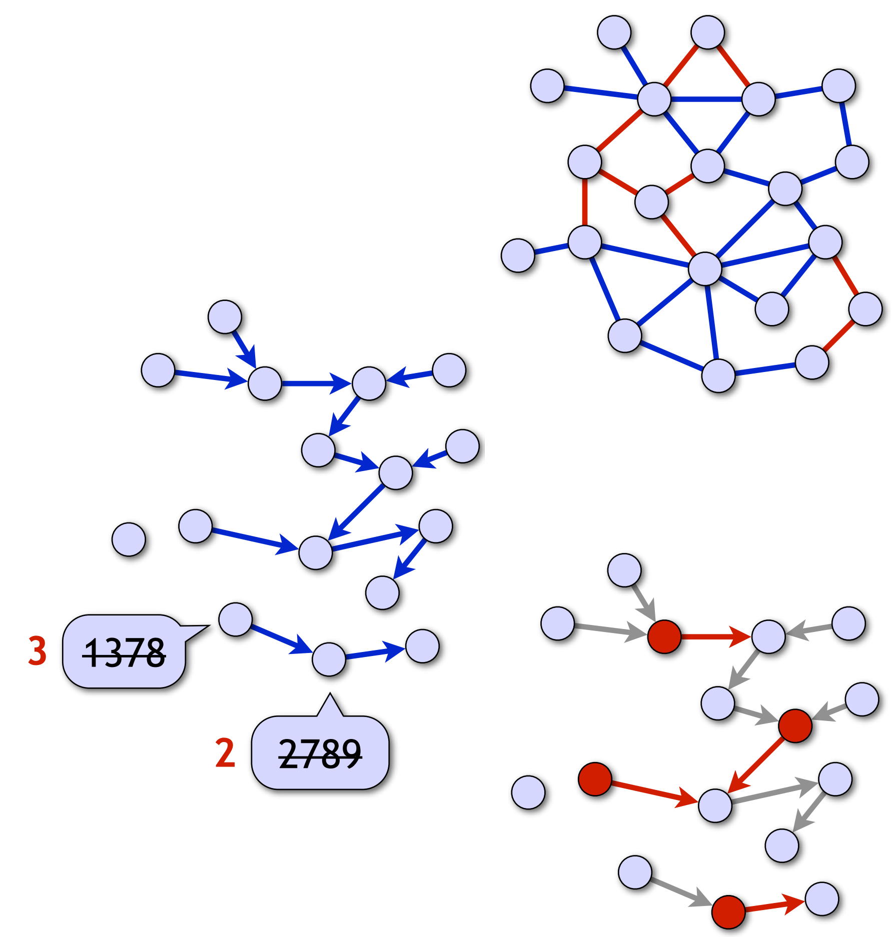
Fast distributed approximation algorithms for vertex cover and set cover in anonymous networks

Jukka Suomela

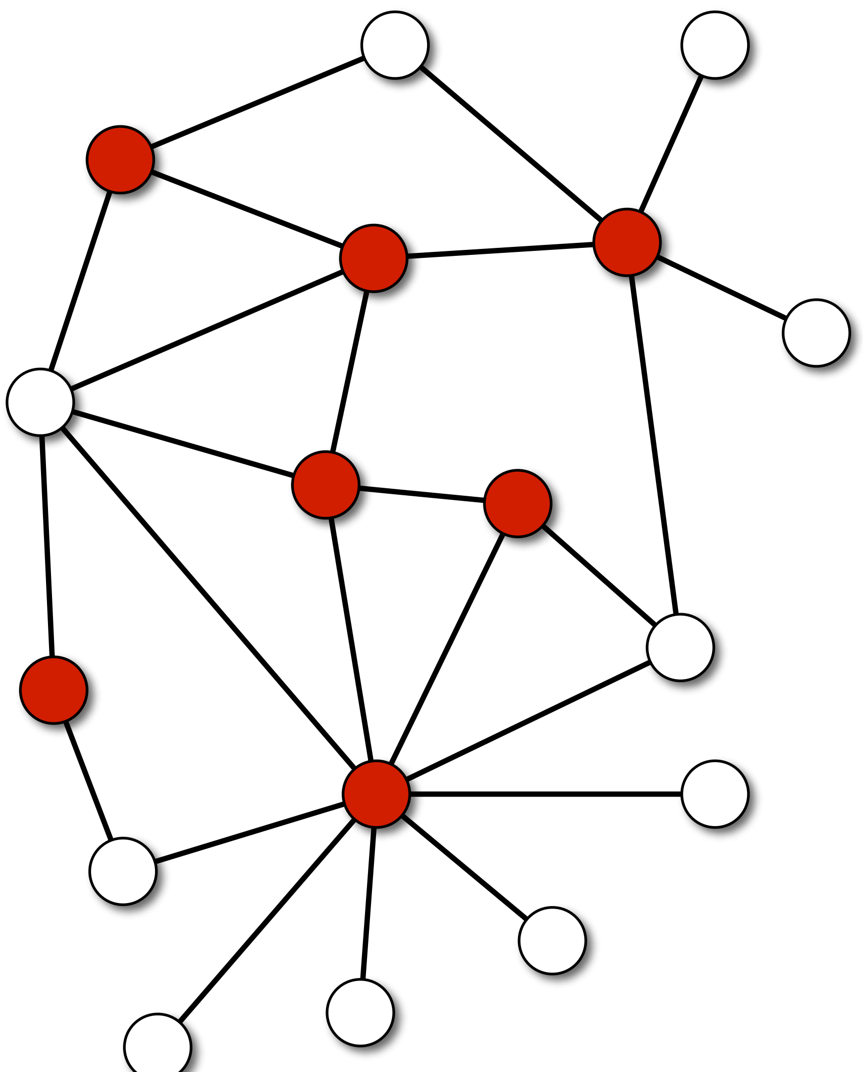
Helsinki Institute for Information Technology HIIT
University of Helsinki, Finland

Braunschweig,
29 November 2010



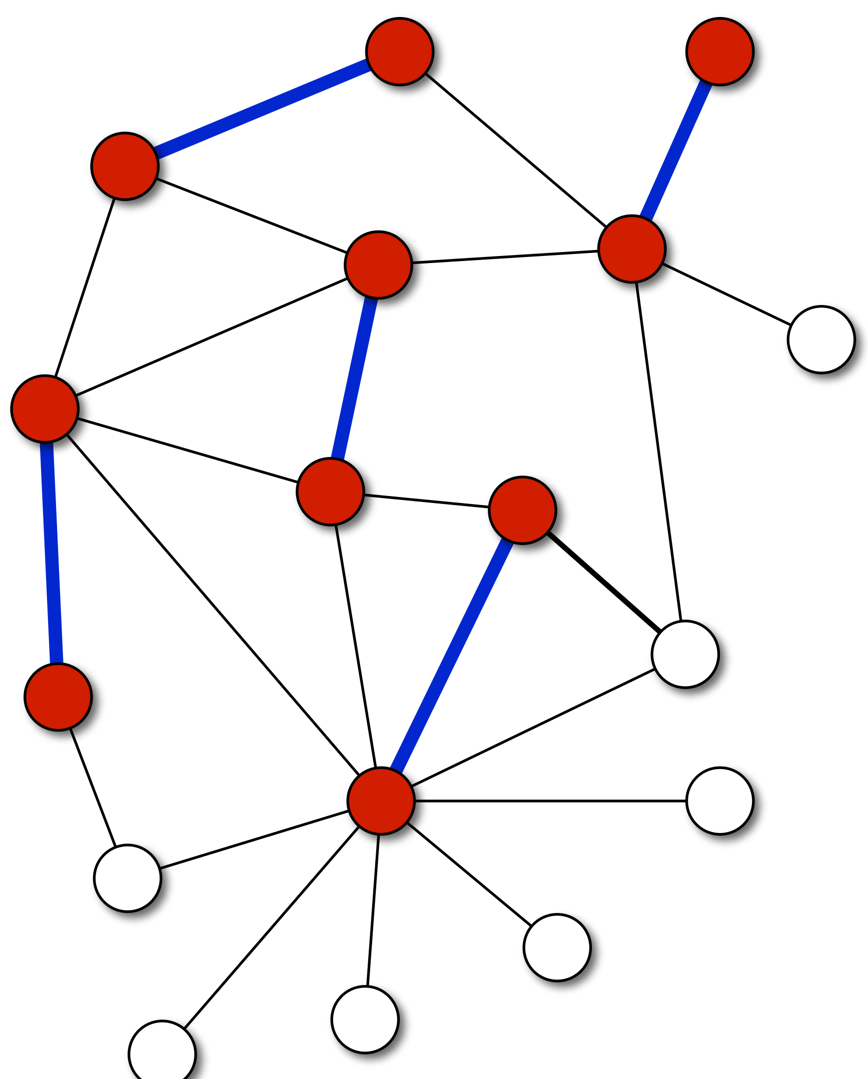
Joint work with Matti Åstrand

Vertex cover problem



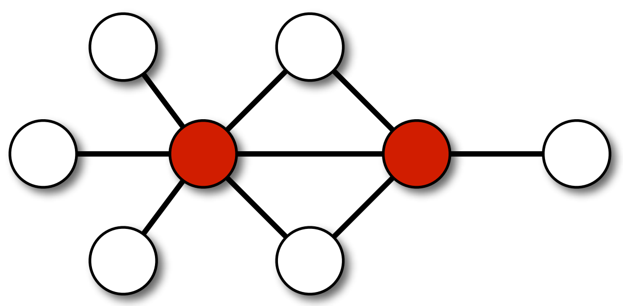
- **Vertex cover *C*** for a graph *G*:
  - Subset *C* of nodes that “covers” all edges of the graph
  - Each edge has at least one endpoint in *C*
- Can we find a ***small*** vertex cover?

Vertex cover problem



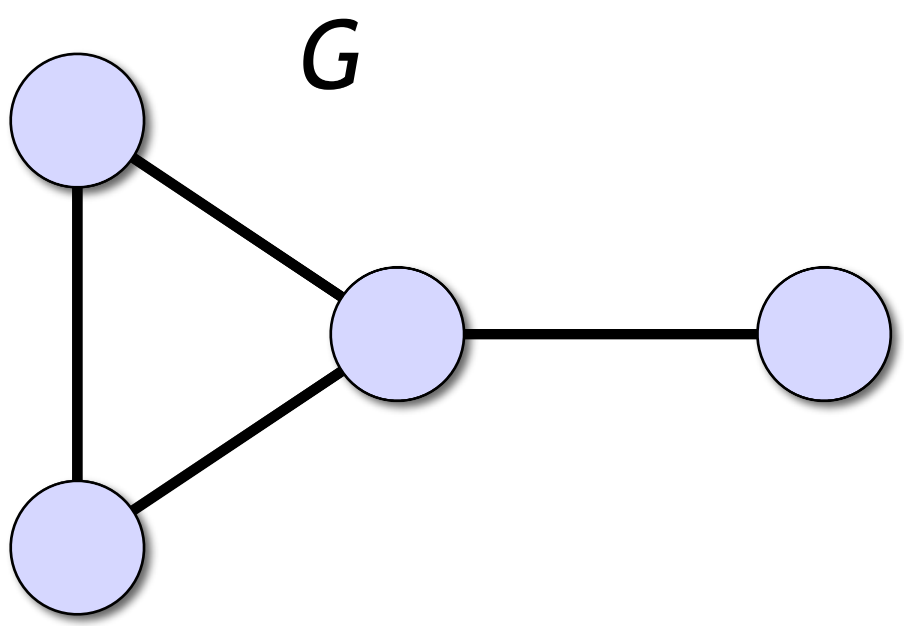
- Classical NP-hard optimisation problem
  - Simple 2-approximation algorithm: endpoints of a ***maximal matching***
  - No polynomial-time algorithm with approximation factor 1.999 known

Research question



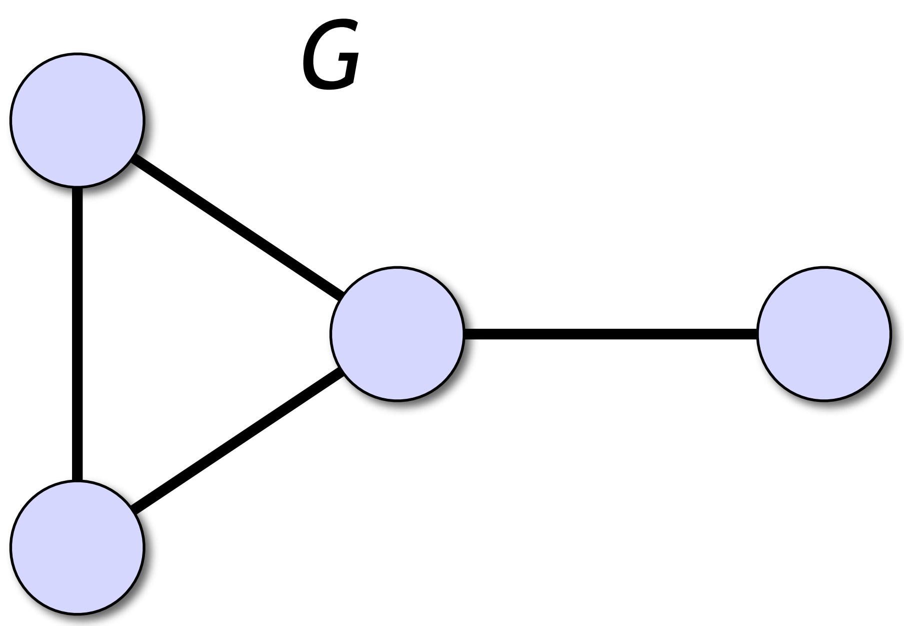
- **Distributed** approximation algorithms for vertex cover
  - Find a small vertex cover in any communication network
  - Best possible approximation ratio
  - As fast as possible: running time independent of *n*
  - Weakest possible models: no randomness, no unique node identifiers
- Let's first define the models...

Distributed algorithms



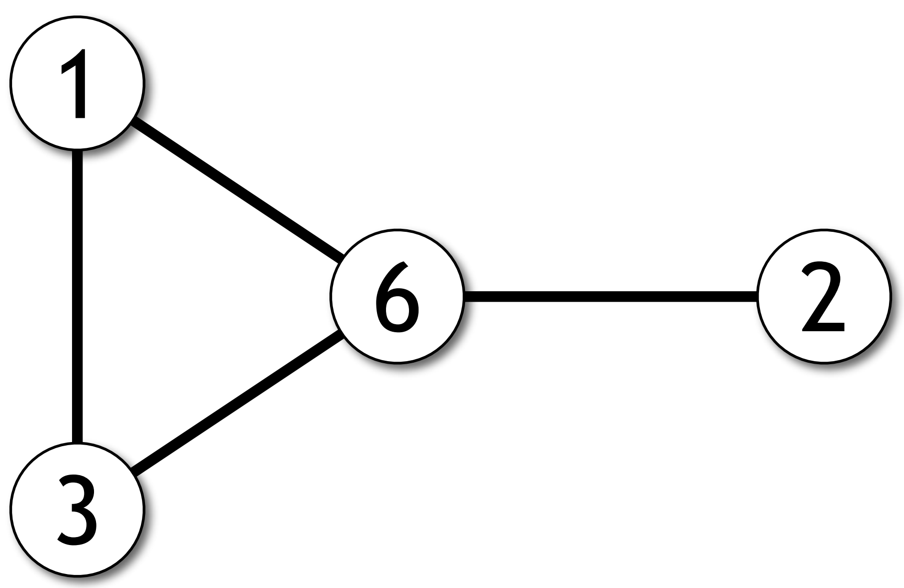
- Communication graph *G*
- Node = computer
- Edge = communication link
- Computers exchange messages and finally decide whether they are in vertex cover *C*
  - "Local output", 0 or 1

Distributed algorithms



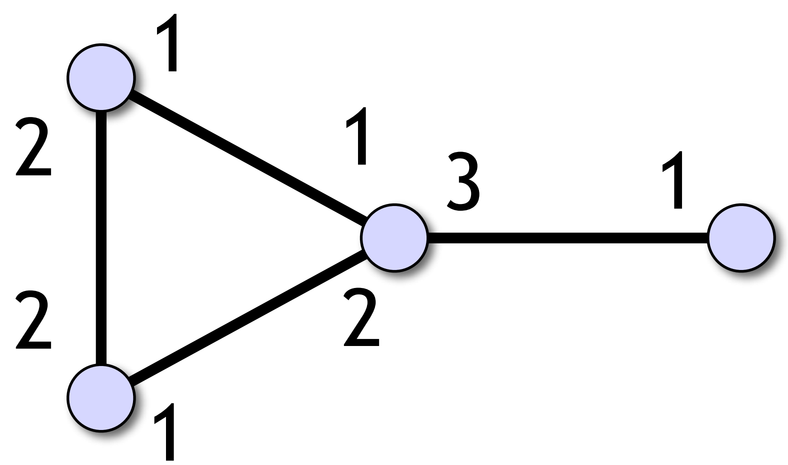
- All nodes are identical, run the **same algorithm**
- ***We*** can choose the algorithm
- An ***adversary*** chooses the structure of *G*
- Our algorithm must produce a valid vertex cover in any graph *G*

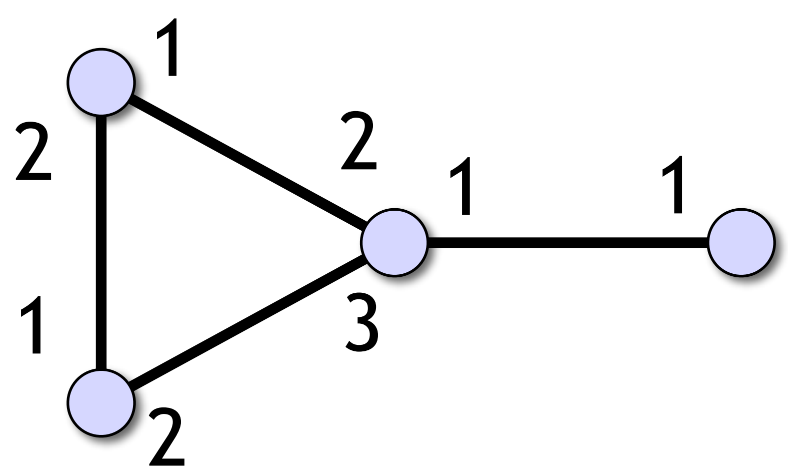
Model 1: Unique identifiers



- The "standard model"
- Node identifiers are a subset of 1, 2, ..., poly(n)
- Subset chosen by adversary

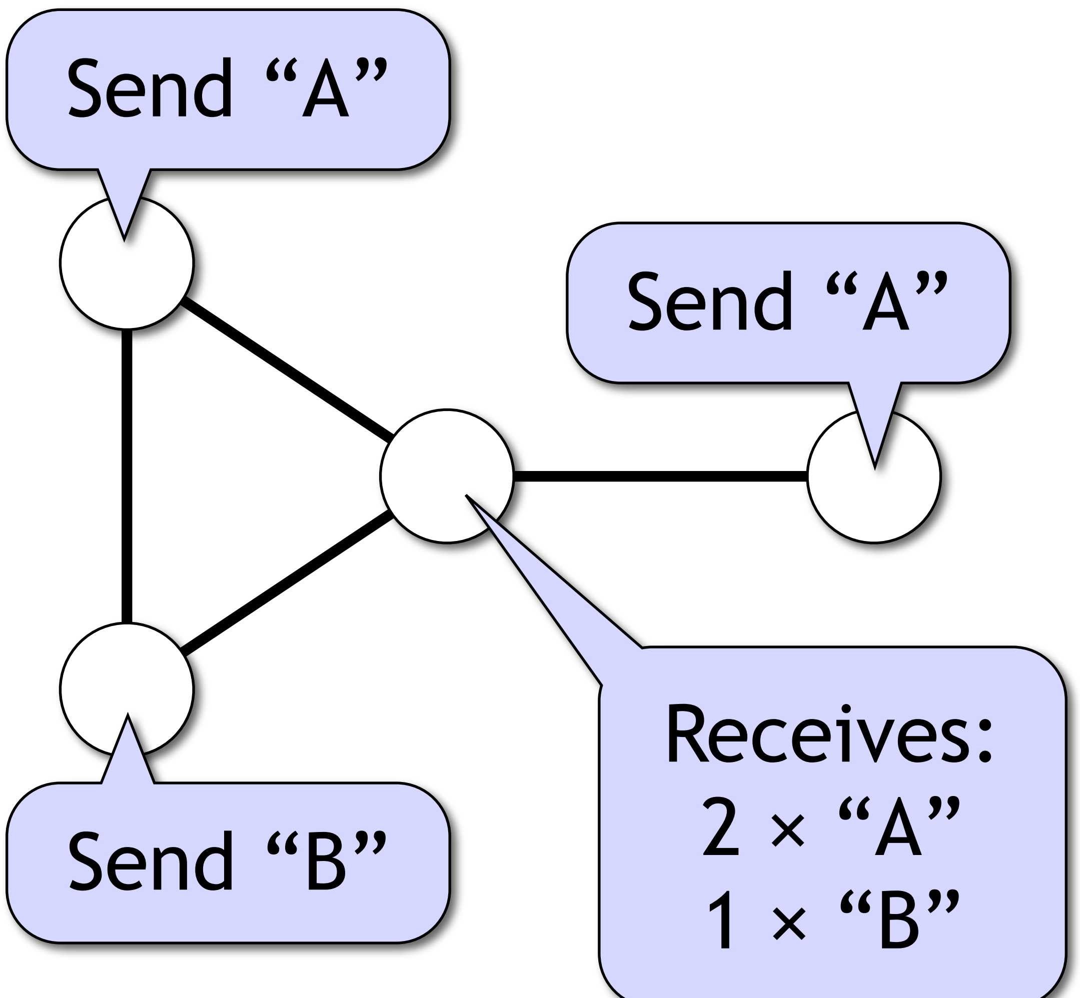
Model 2: Port-numbering model





- No unique identifiers
- A node of degree *d* can refer to its neighbours by integers 1, 2, ..., *d*
- Port-numbering chosen by adversary

Model 3: Broadcast model



- No identifiers, no port numbers
- A node has to send the **same message** to each neighbour
- A node does not know which message was received from which neighbour (*multiset*)

Deterministic distributed algorithms for vertex cover

- Guaranteed approximation ratios?
  - E.g., 2-approximation of minimum vertex cover = at most 2 times as large as the smallest vertex cover
- Fast?
  - Time = number of communication rounds
  - n = number of nodes
  - Δ = maximum degree
- In weak models of distributed computing?

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$						1
$f(\Delta)$ + polylog(n)						
$f(\Delta) + O(\log^* n)$						
$f(\Delta)$						
	Broadcast model		Port numbering		Unique identifiers	

Trivial algorithm

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$						1
$f(\Delta)$ + polylog(n)						**2**
$f(\Delta) + O(\log^* n)$						**2**
$f(\Delta)$						
	Broadcast model		Port numbering		Unique identifiers	

Maximal matching

(Panconesi & Rizzi 2001)

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$				**2**		1
$f(\Delta)$ + polylog(n)				**2**		2
$f(\Delta) + O(\log^* n)$						2
$f(\Delta)$						
	Broadcast model		Port numbering		Unique identifiers	

Near-maximal edge packing

(Khuller et al. 1994)

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$				2		1
$f(\Delta)$ + polylog(n)				2		2
$f(\Delta) + O(\log^* n)$				**2 + ε**		2
$f(\Delta)$				**2 + ε**		**2 + ε**
	Broadcast model		Port numbering		Unique identifiers	

Deterministic LP rounding

(Kuhn et al. 2006)

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$				2		1
$f(\Delta)$ + polylog(n)				2		2
$f(\Delta) + O(\log^* n)$				2 + ε		2
$f(\Delta)$	**2**		**2**	2 + ε	**2**	2 + ε
	Broadcast model		Port numbering		Unique identifiers	

Czygrinow et al. 2008

Lenzen & Wattenhofer 2008

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$	**2**		**2**	2		1
$f(\Delta)$ + polylog(n)	**2**		**2**	2		2
$f(\Delta) + O(\log^* n)$	**2**		**2**	2 + ε		2
$f(\Delta)$	**2**		**2**	2 + ε	2	2 + ε
	Broadcast model		Port numbering		Unique identifiers	

Trivial (cycles)

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$	2		2	2		1
$f(\Delta)$ + polylog(n)	2		2	2		2
$f(\Delta) + O(\log^* n)$	2		2	2 + ε		2
$f(\Delta)$	2		2	2 + ε	2	2 + ε
	Broadcast model		Port numbering		Unique identifiers	

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$	2	**?**				1
$f(\Delta)$ + polylog(n)	2	**?**				
$f(\Delta) + O(\log^* n)$	2	**?**	2	**2 + ε**		
$f(\Delta)$	2	**?**	2	**2 + ε**	2	**2 + ε**
	Broadcast model		Port numbering		Unique identifiers	

Anything here?

Could we have 2?

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$	2	**?**	2	2		1
$f(\Delta)$ + polylog(n)	2	**?**	2	2		
$f(\Delta) + O(\log^* n)$	2	**?**	2	**2**		
$f(\Delta)$	2	**?**	2	**2**	2	**2**
	Broadcast model		Port numbering		Unique identifiers	

DISC 2009

Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$	2	**2**				1
$f(\Delta)$ + polylog(n)	2	**2**				
$f(\Delta) + O(\log^* n)$	2	**2**	2	**2**		
$f(\Delta)$	2	**2**	2	**2**	2	**2**
	Broadcast model		Port numbering		Unique identifiers	

Latest results

+ faster and more general solution here

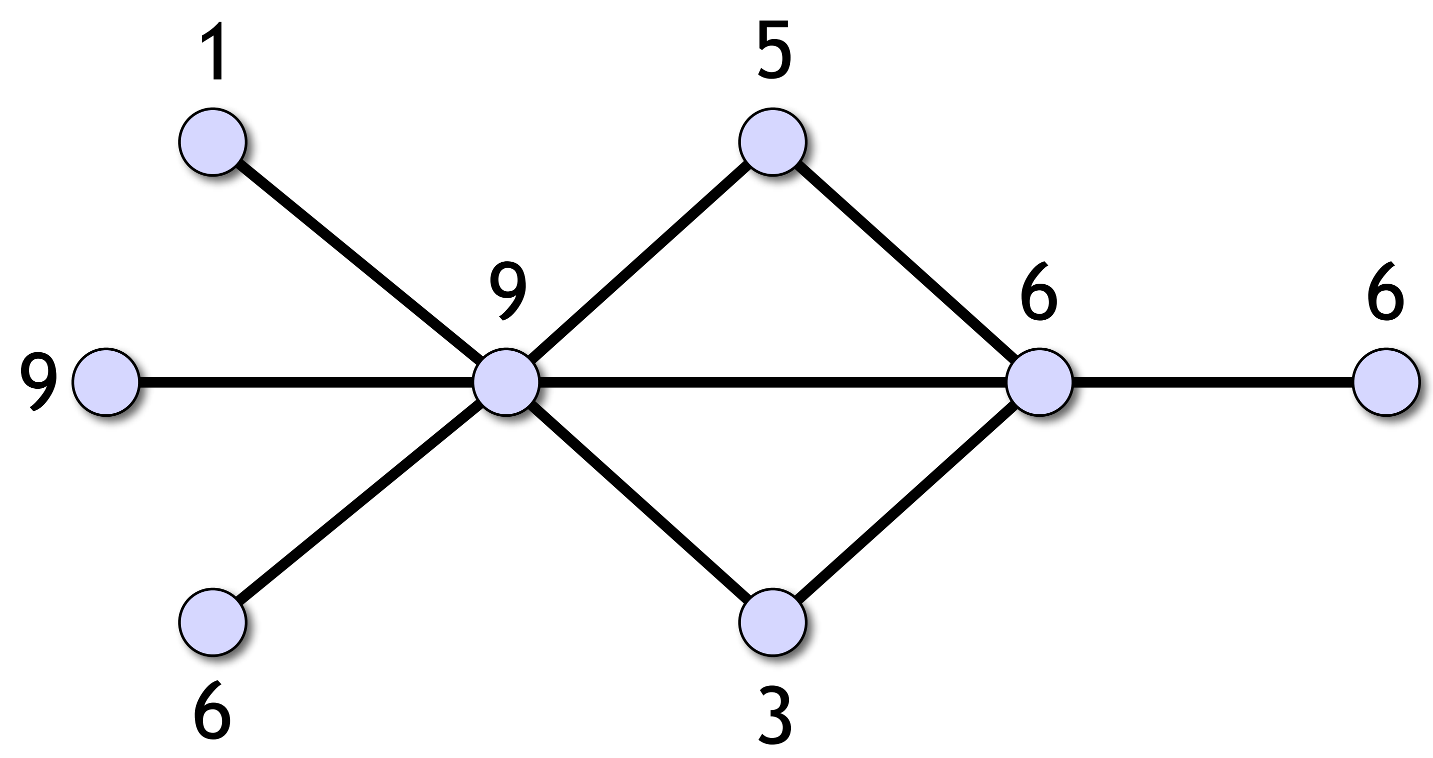
Deterministic distributed algorithms for vertex cover: approximation ratios

Time	lower	upper	lower	upper	lower	upper
$O(n)$	2	**2**	2	2		1
$f(\Delta)$ + polylog(n)	2	**2**	2	2		
$f(\Delta) + O(\log^* n)$	2	**2**	2	**2**		
$f(\Delta)$	2	**2**	2	**2**	2	**2**
	Broadcast model		Port numbering		Unique identifiers	

Let's study this case first...

Vertex cover in the port-numbering model

- Convenient to study a more general problem: minimum-weight vertex cover
  - More general problems are sometimes easier to solve?



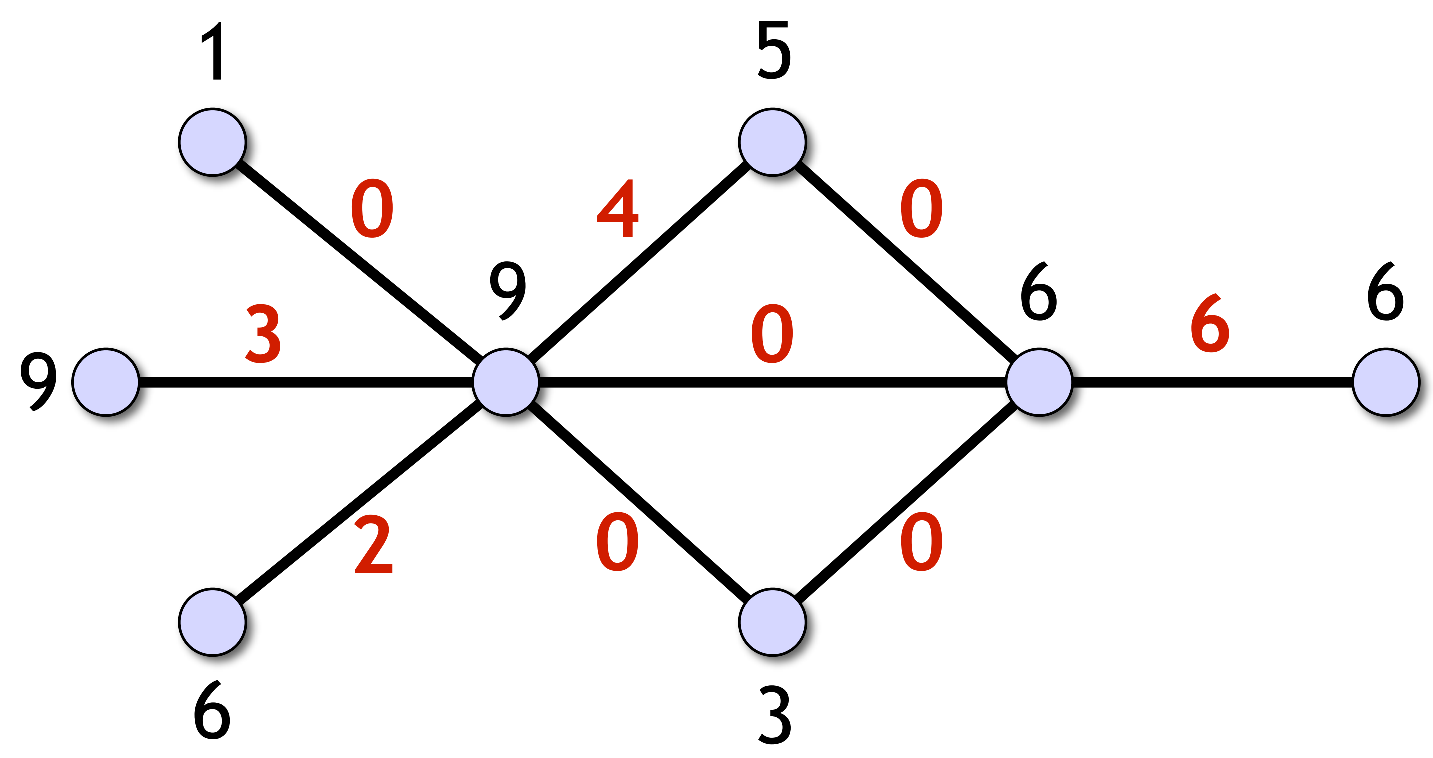
Notation:
$w(v)$ = weight of v

Edge packings and vertex covers

- **Edge packing**: weight $y(e) \geq 0$ for each edge e
  - Packing constraint: $y[v] \leq w(v)$ for each node v, where $y[v]$ = total weight of edges incident to v

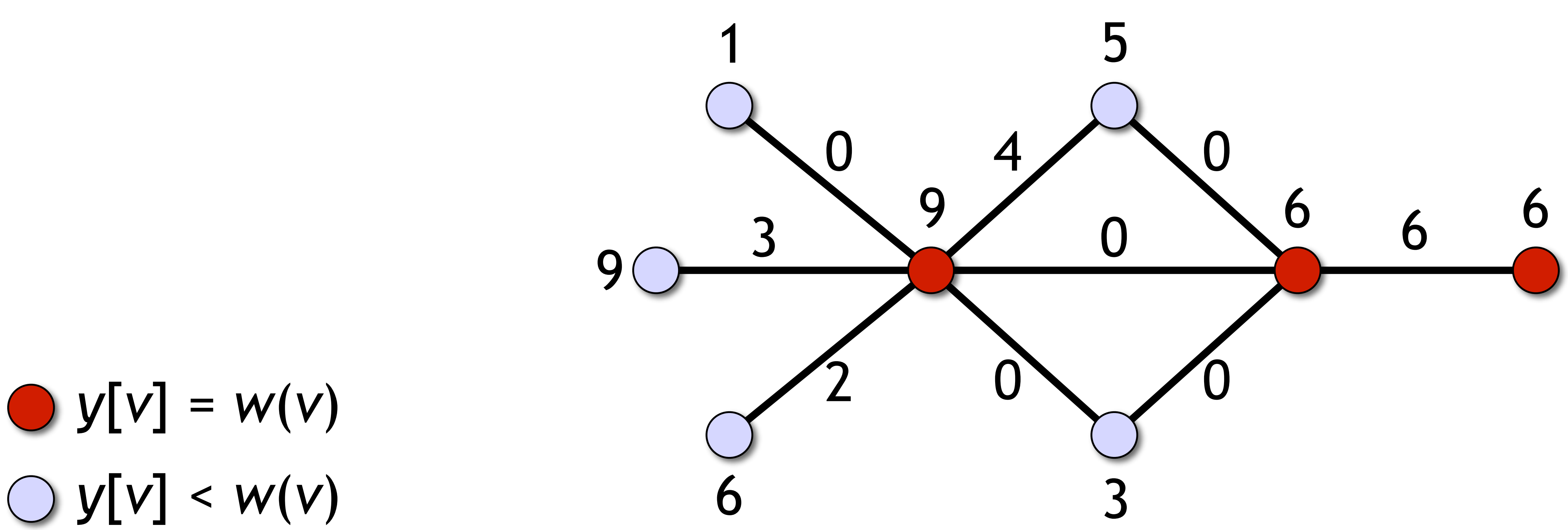
edge packing
≈ fractional matching

(LP relaxation)



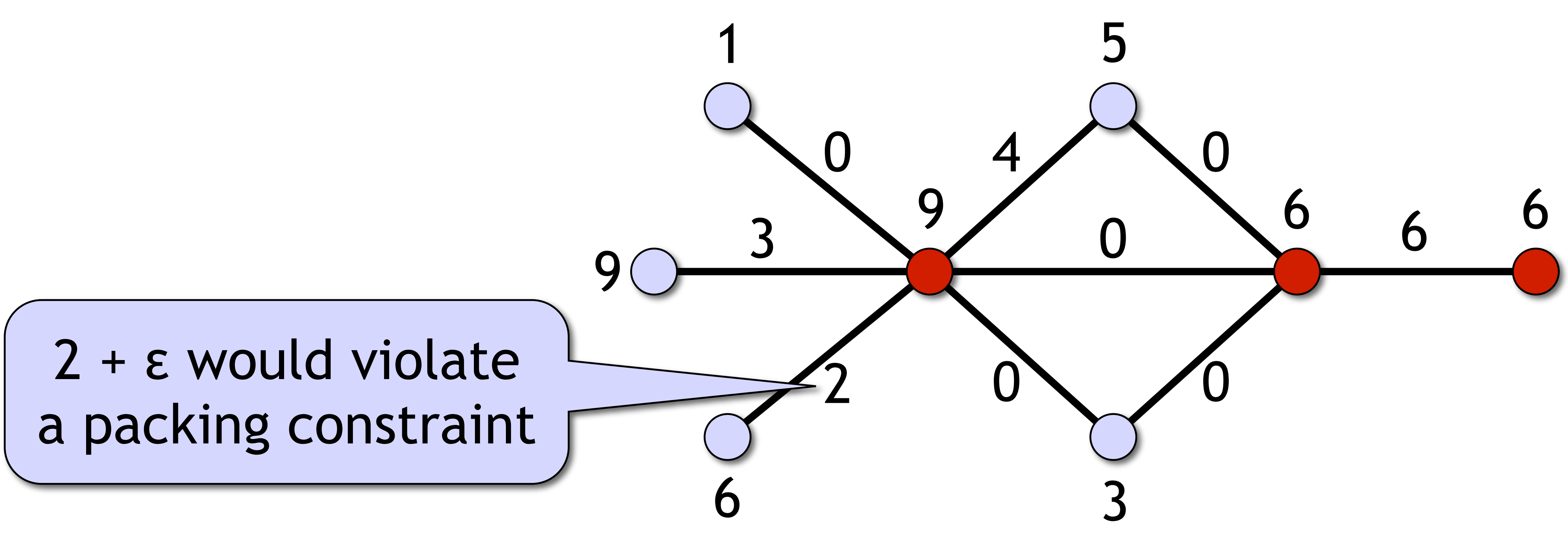
Edge packings and vertex covers

- Node v is **saturated** if $y[v] = w(v)$
  - Total weight of edges incident to v is ***equal*** to $w(v)$, i.e., the packing constraint holds with equality



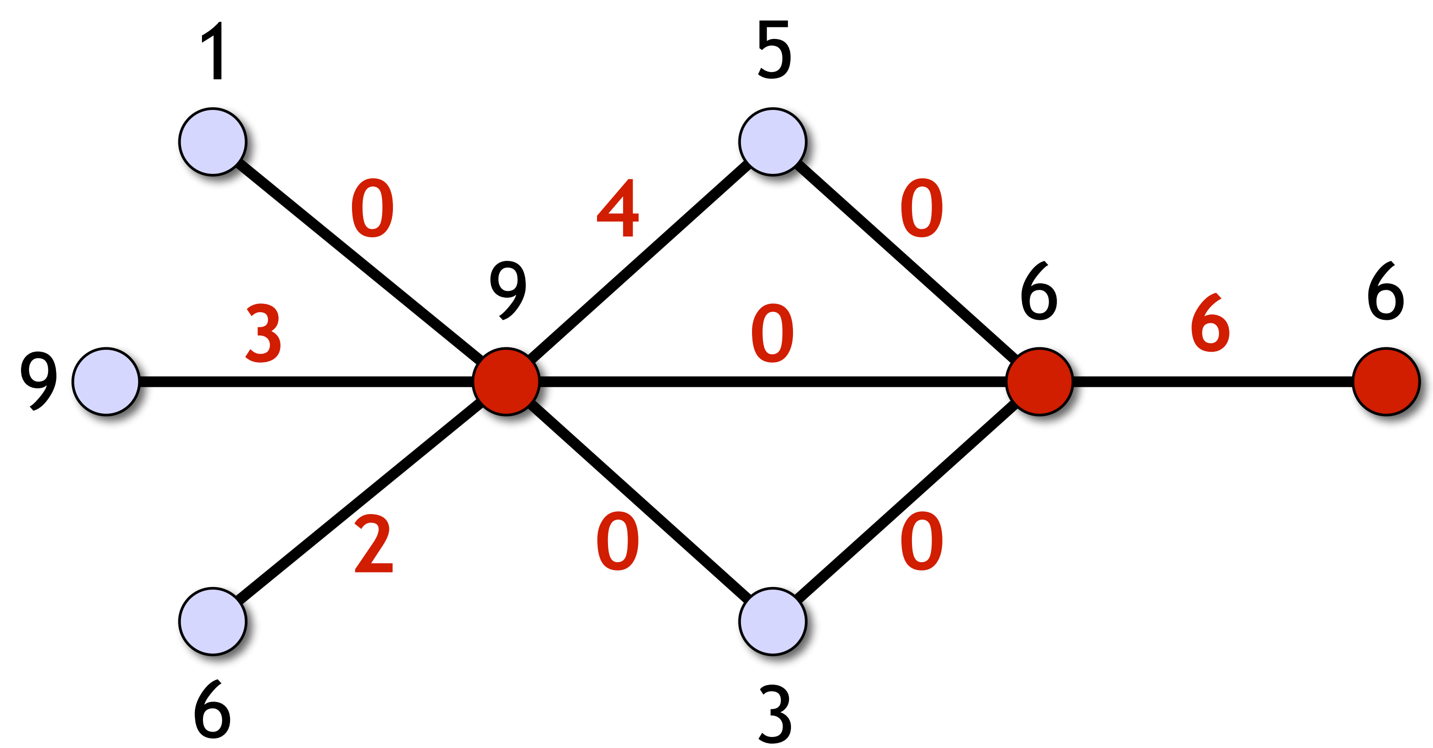
Edge packings and vertex covers

- Edge e is **saturated** if at least one endpoint of e is saturated
  - Equivalently: edge weight $y(e)$ can't be increased



Edge packings and vertex covers

- **Maximal edge packing**: all edges saturated
  $\Longleftrightarrow$ none of the edge weights $y(e)$ can be increased
  $\Longleftrightarrow$ saturated nodes form a vertex cover

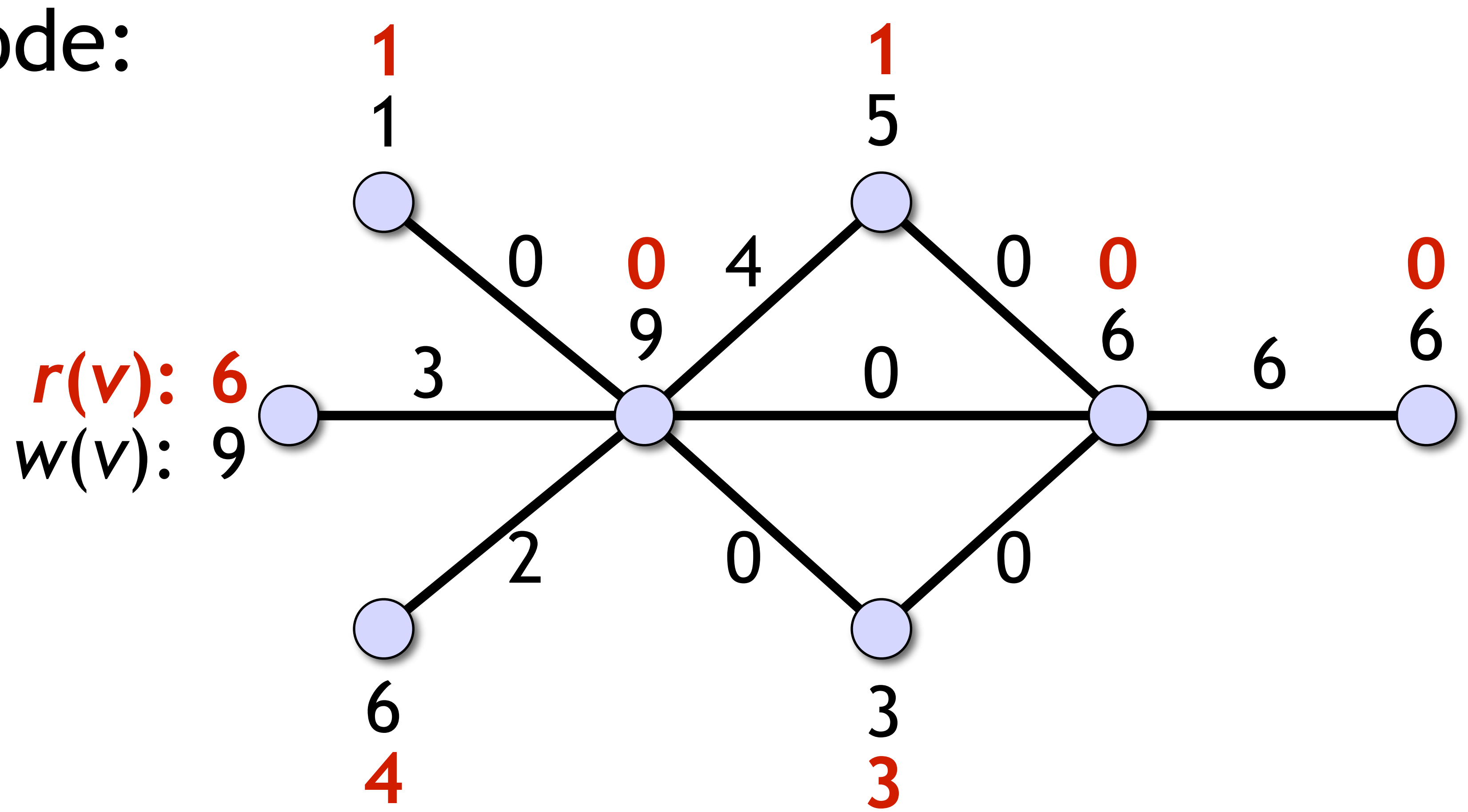


Edge packings and vertex covers

- **Maximal edge packing**: all edges saturated
  $\Longleftrightarrow$ saturated nodes form a vertex cover
  - ... and saturated nodes are **2-approximation** of minimum-weight vertex cover (Bar-Yehuda & Even 1981)
- How to find a maximal edge packing...?
  - Phase I: "*greedy but safe*", cf. Khuller et al. (1994), Papadimitriou & Yannakakis (1993)
  - Phase II: if phase I fails to saturate an edge $e = \{u,v\}$, we can *break symmetry* between u and v; exploit it!

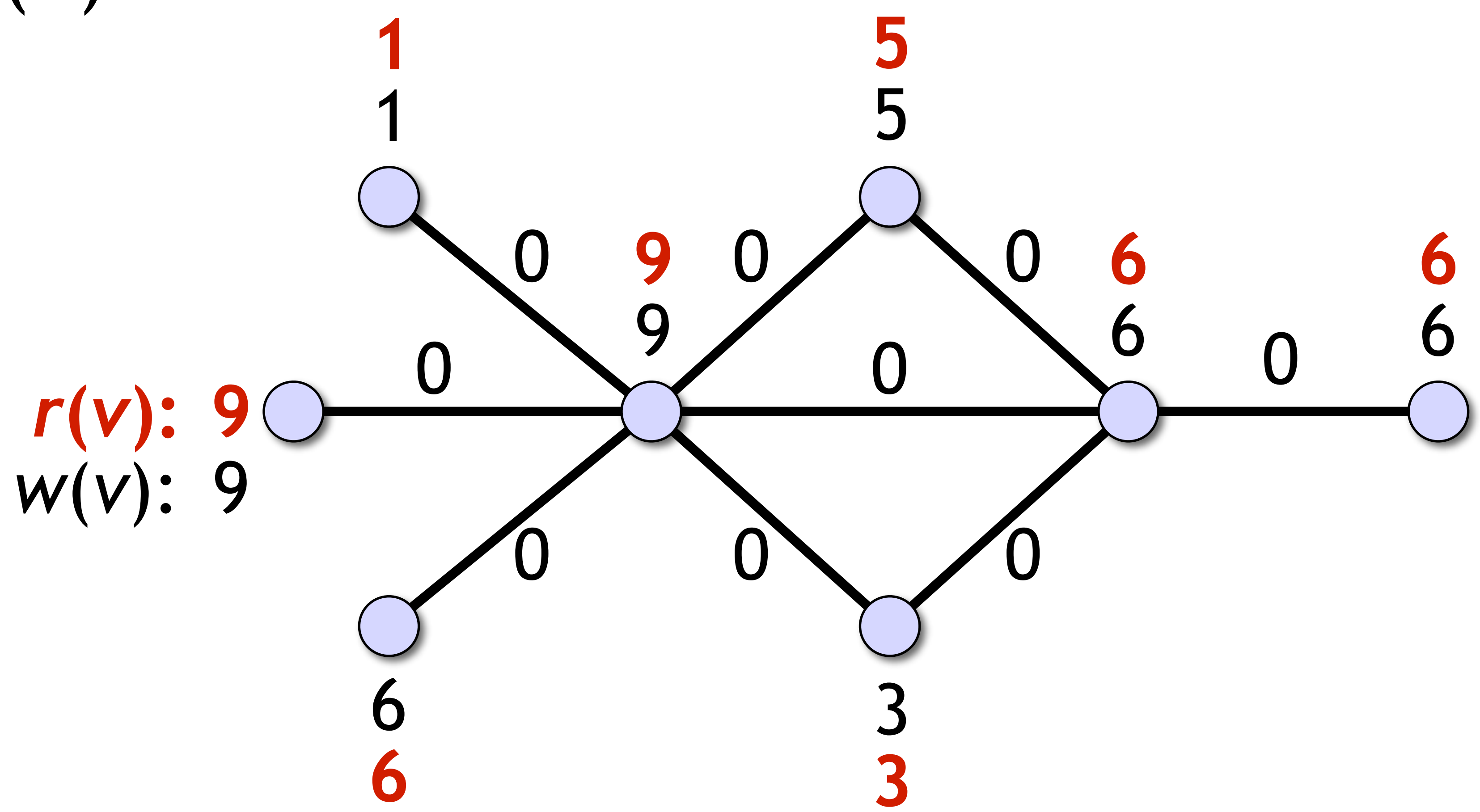
Finding a maximal edge packing: phase I

- $y[v]$ = total weight of edges incident to node v
- **Residual capacity** of node v: $r(v) = w(v) - y[v]$
- Saturated node: $r(v) = 0$



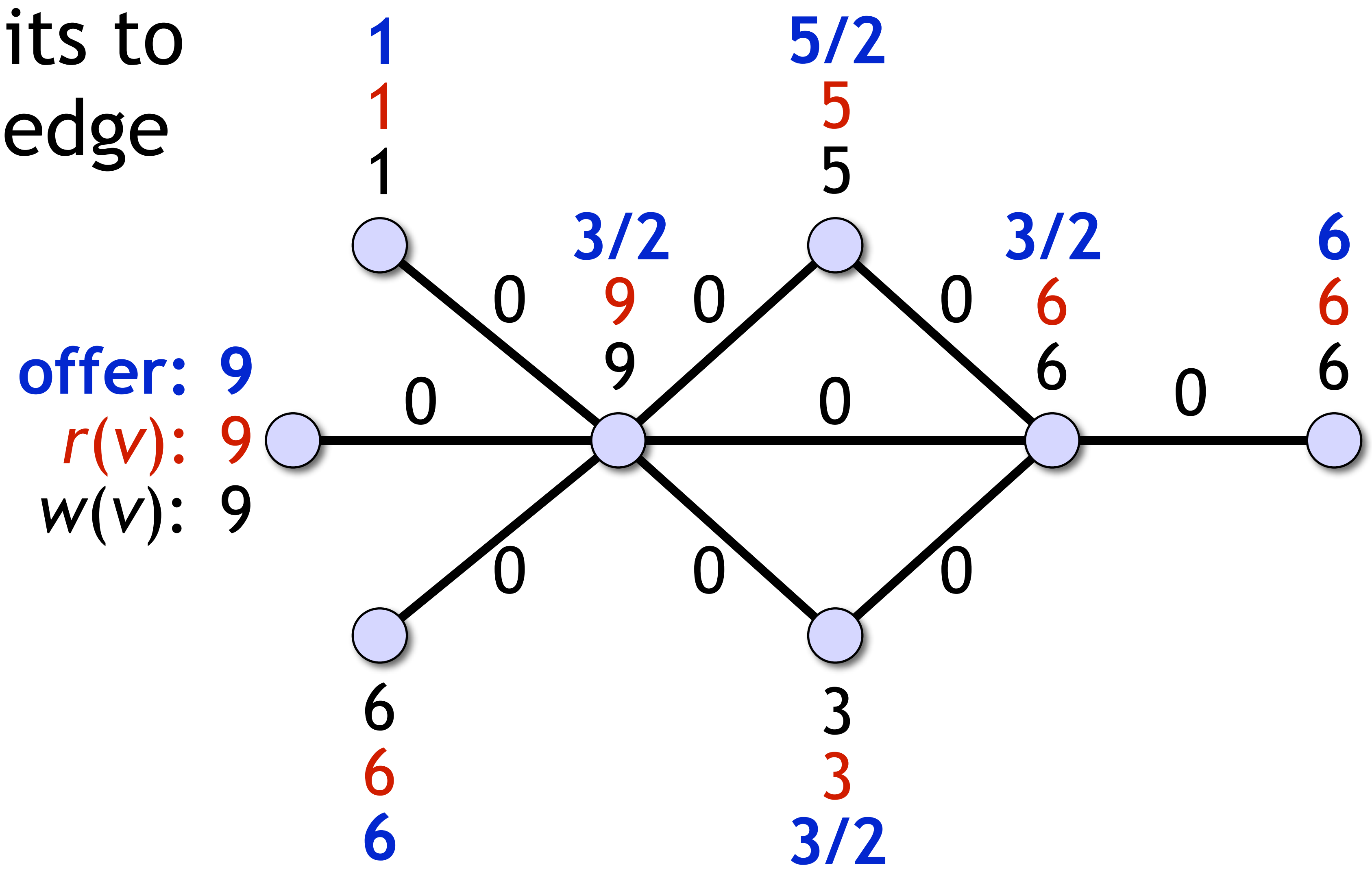
Finding a maximal edge packing: phase I

Start with a trivial edge packing $y(e) = 0$



Finding a maximal edge packing: phase I

Each node *v* **offers** *r*(*v*)/deg(*v*) units to each incident edge

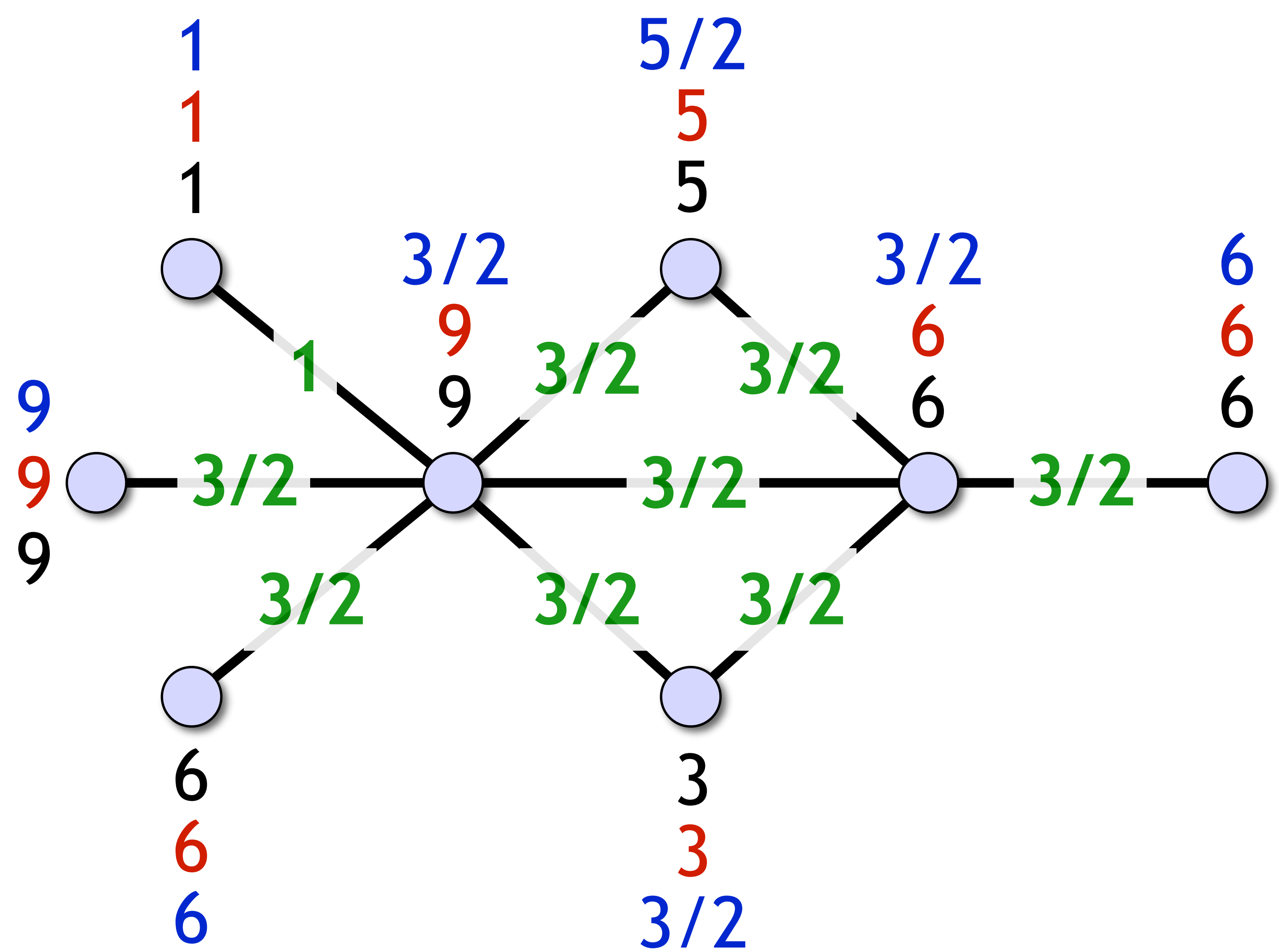


Finding a maximal edge packing: phase I

Each edge **accepts** the smallest of the 2 offers it received

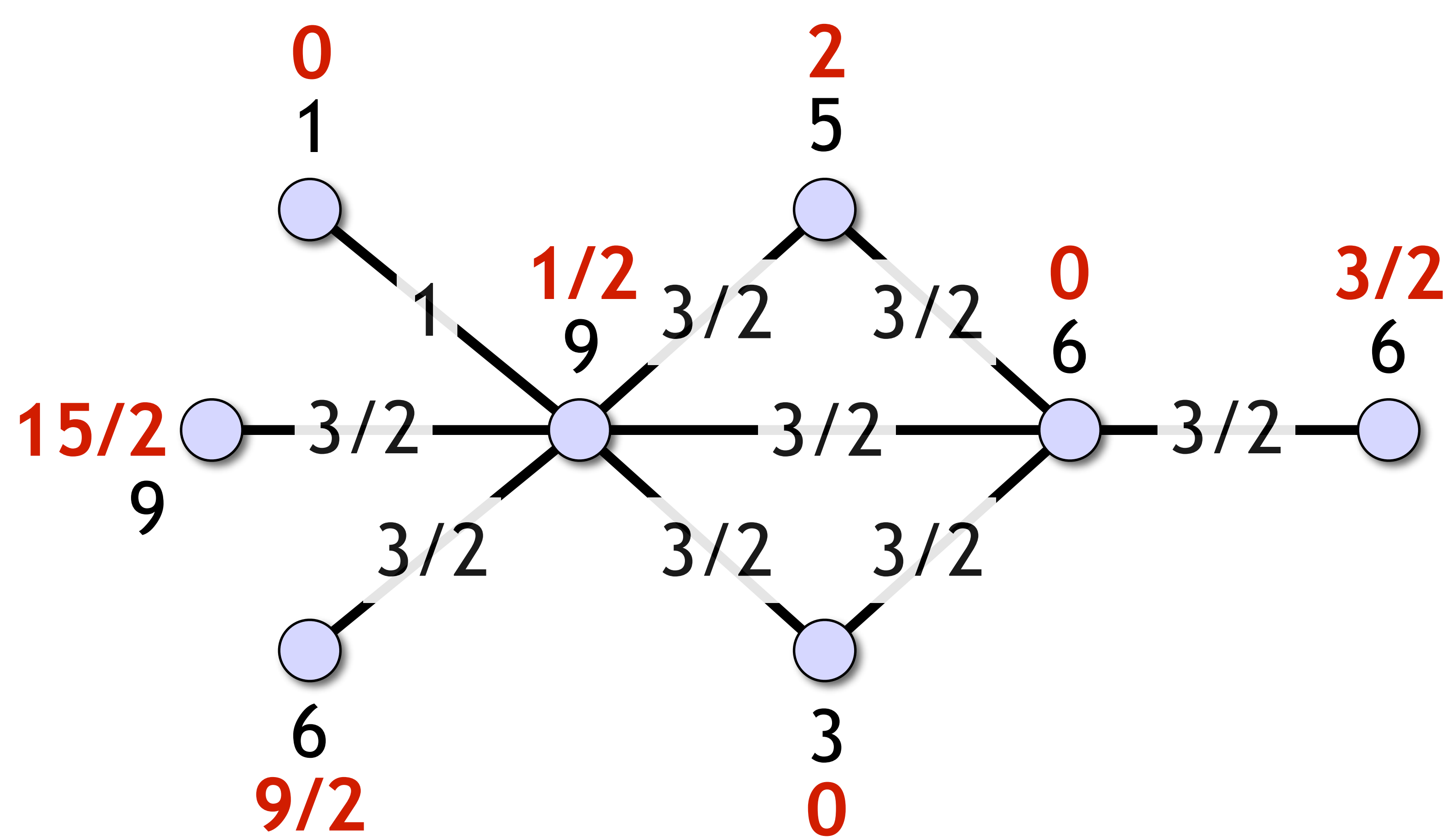
Increase $y(e)$ by this amount

- Safe, can’t violate packing constraints



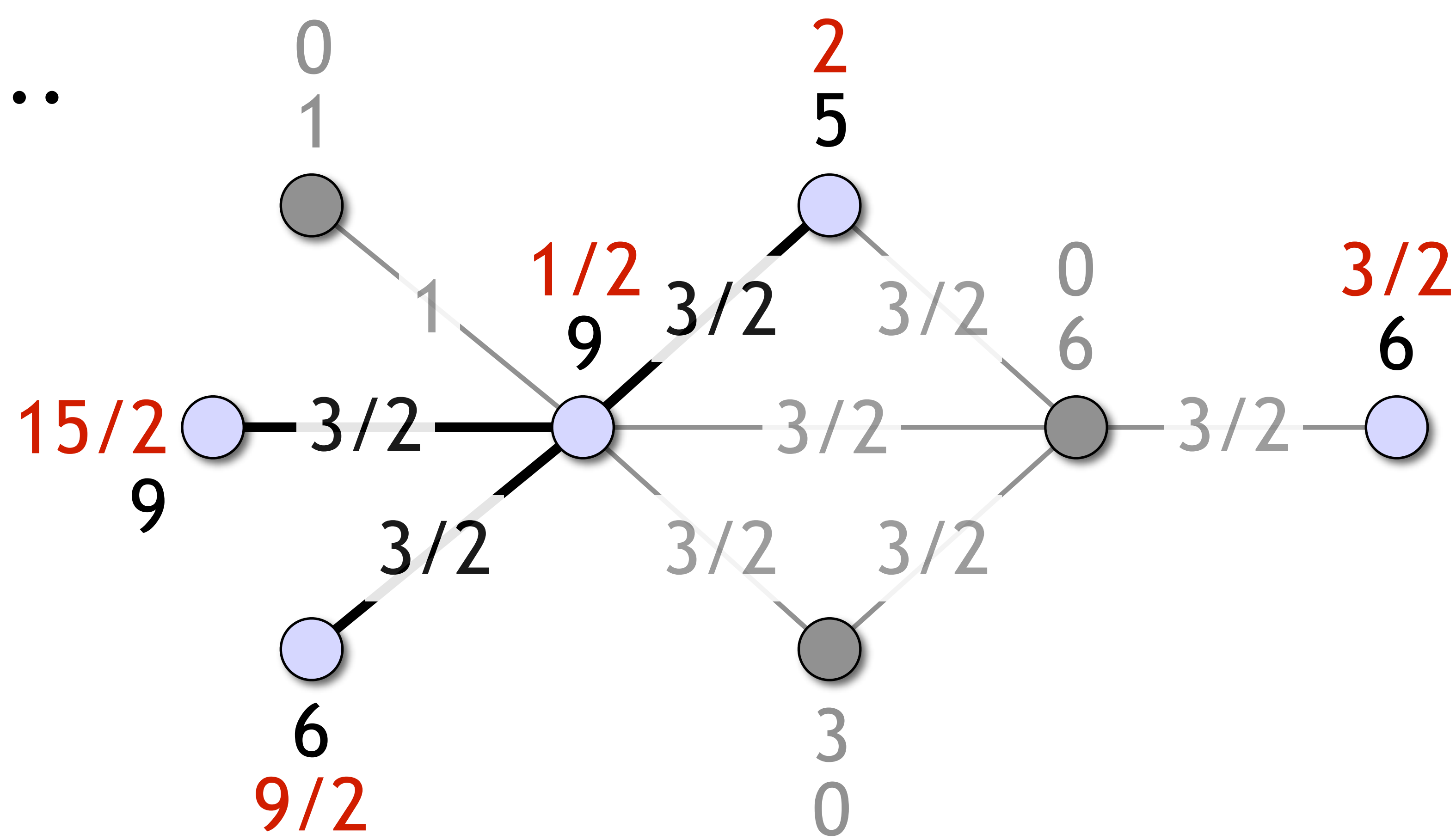
Finding a maximal edge packing: phase I

Update **residuals**...



Finding a maximal edge packing: phase I

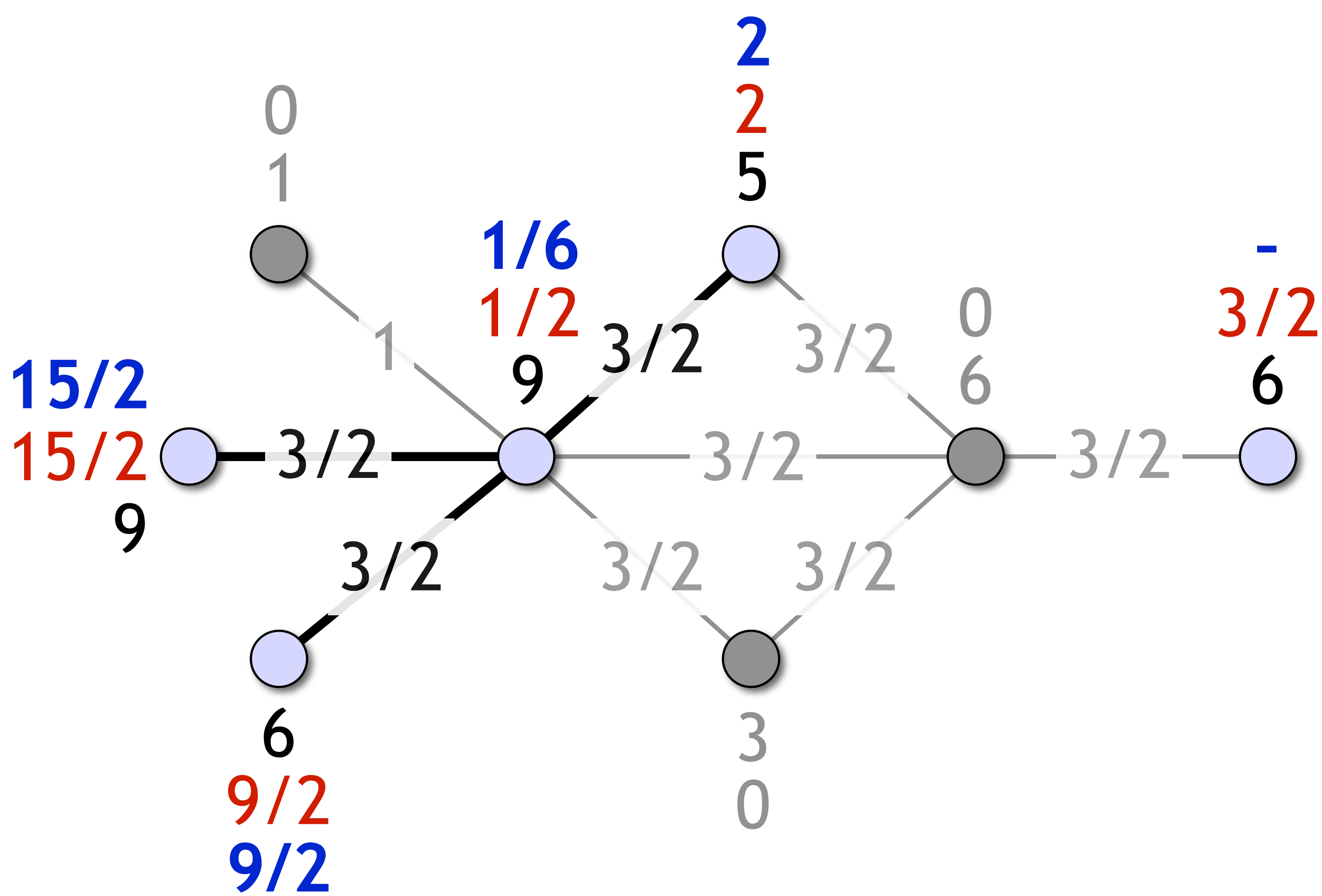
Update residuals, discard saturated nodes and edges...



Finding a maximal edge packing: phase I

Update residuals, discard saturated nodes and edges, repeat...

Offers...

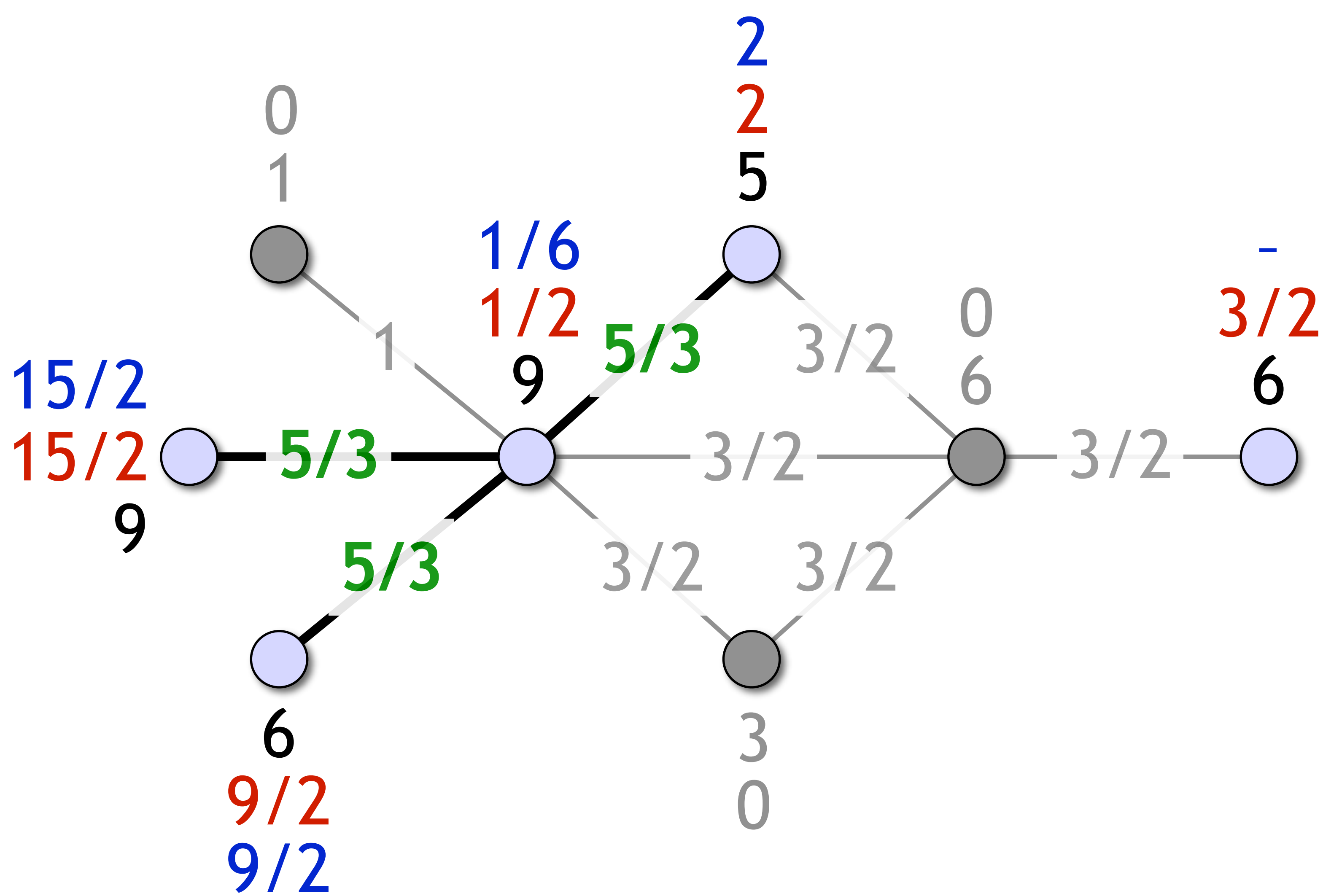


Finding a maximal edge packing: phase I

Update residuals, discard saturated nodes and edges, repeat...

Offers...

Increase weights...



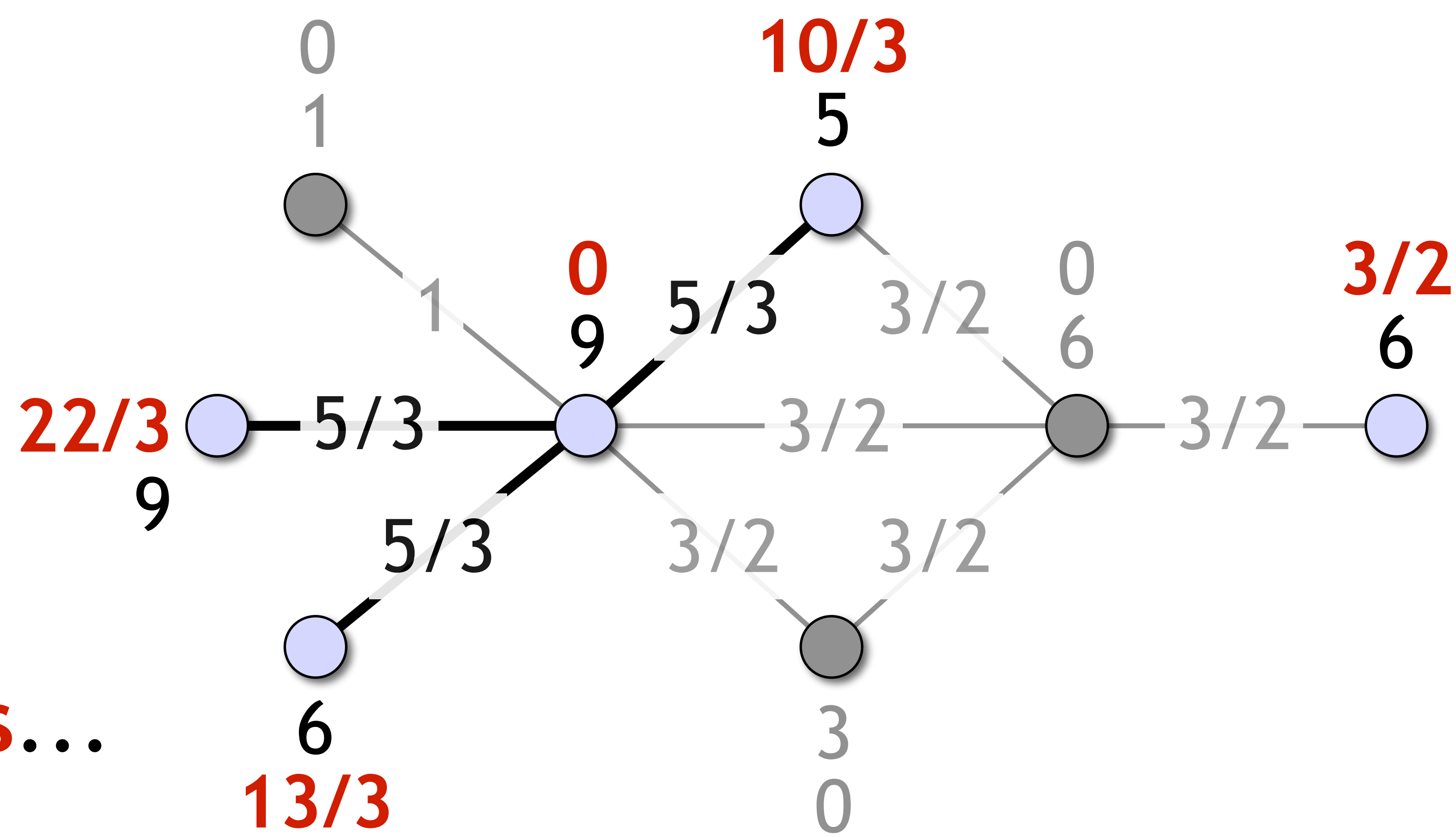
Finding a maximal edge packing: phase I

Update residuals, discard saturated nodes and edges, repeat...

Offers...

Increase weights...

Update residuals...



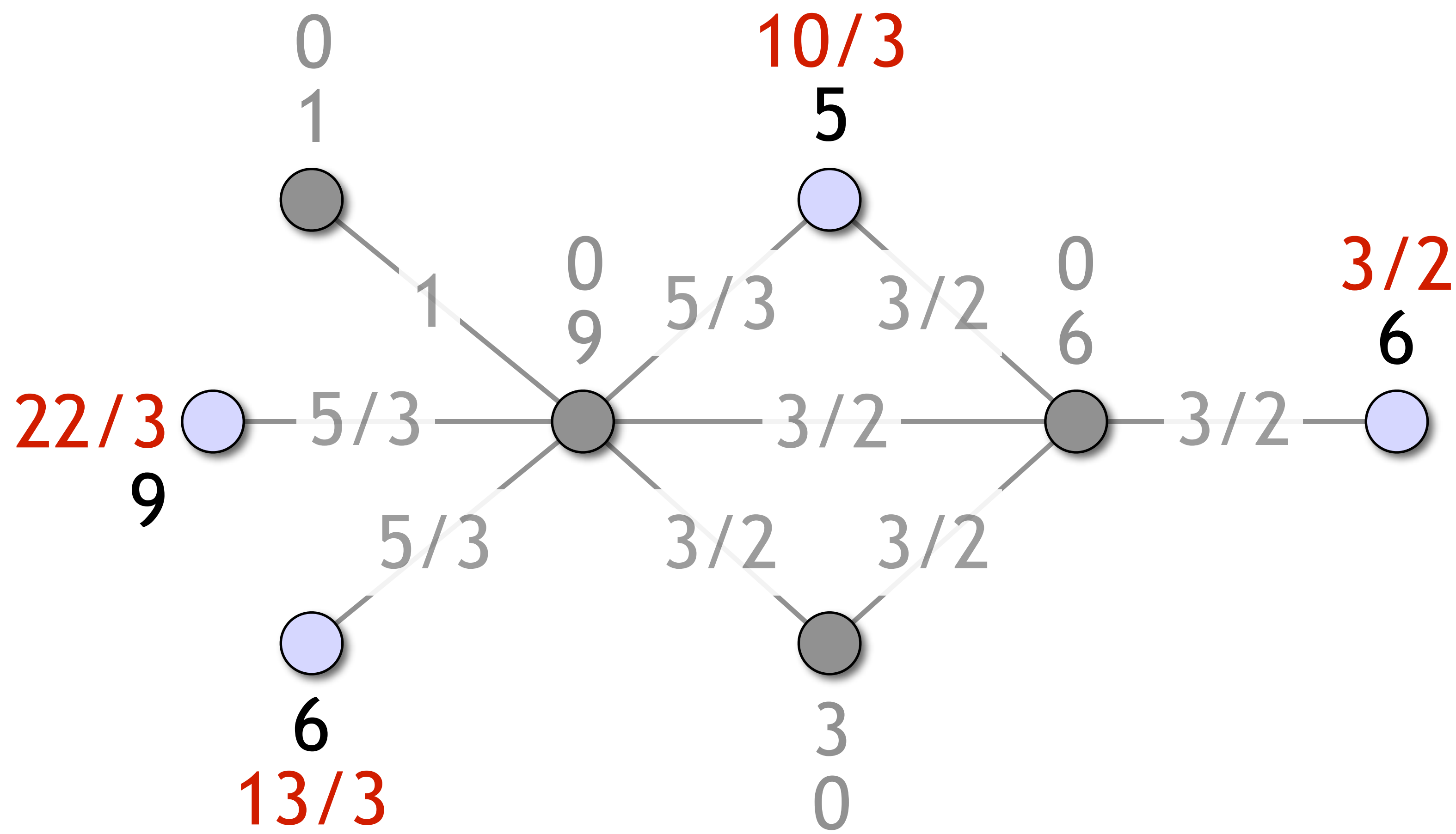
Finding a maximal edge packing: phase I

Update residuals, discard saturated nodes and edges, repeat...

Offers...

Increase weights...

Update residuals and graph, etc.

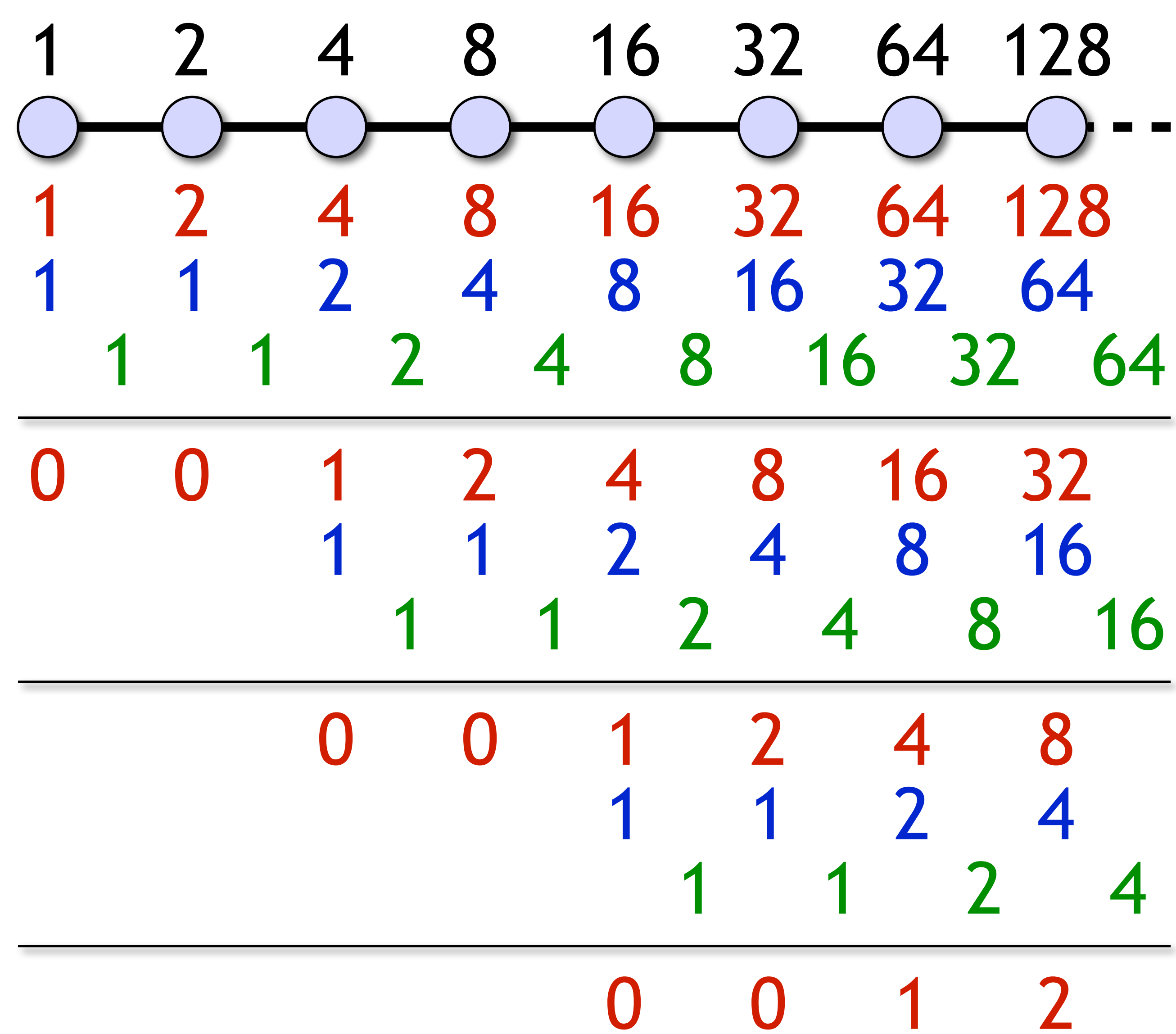


Finding a maximal edge packing: phase I

We are making some progress towards finding a maximal edge packing...

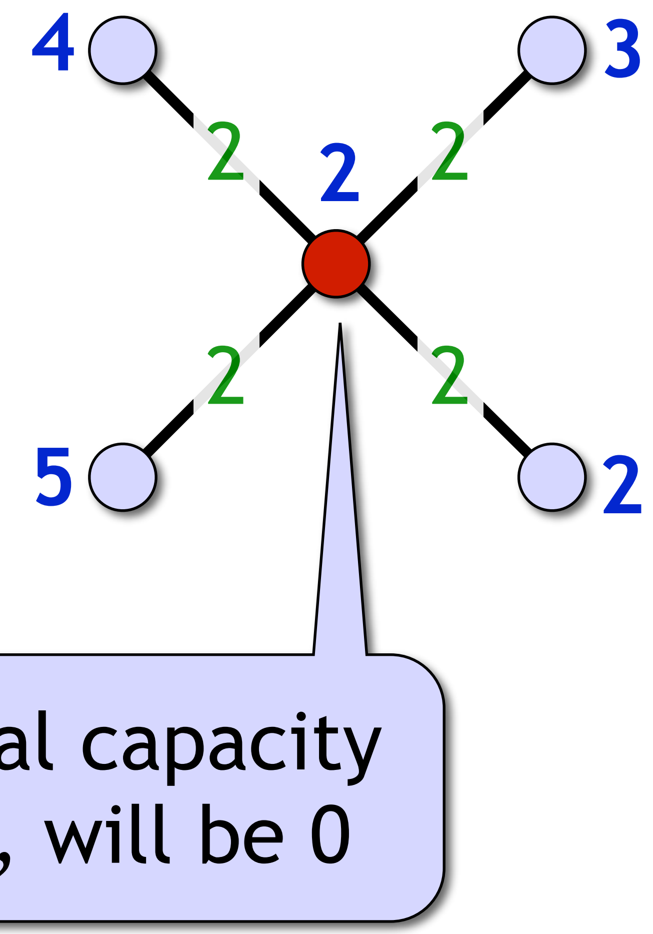
But this is **too slow**!

How to make it faster?



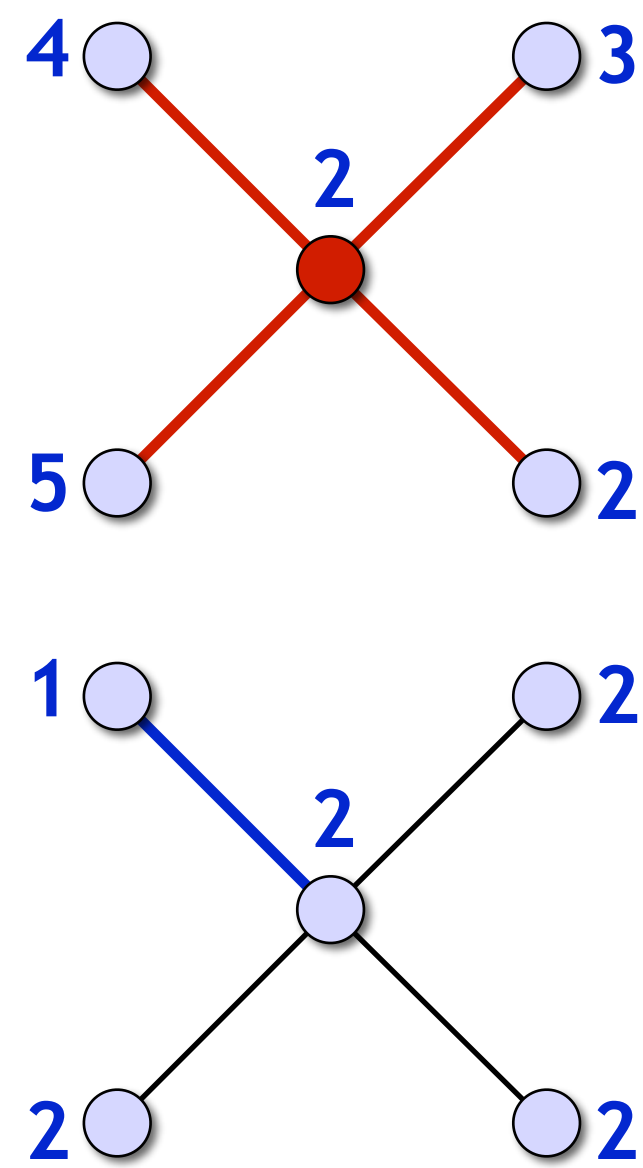
Finding a maximal edge packing: colouring trick

- Offer is a local minimum:
  - Node will be **saturated**
  - And all edges incident to it will be saturated as well



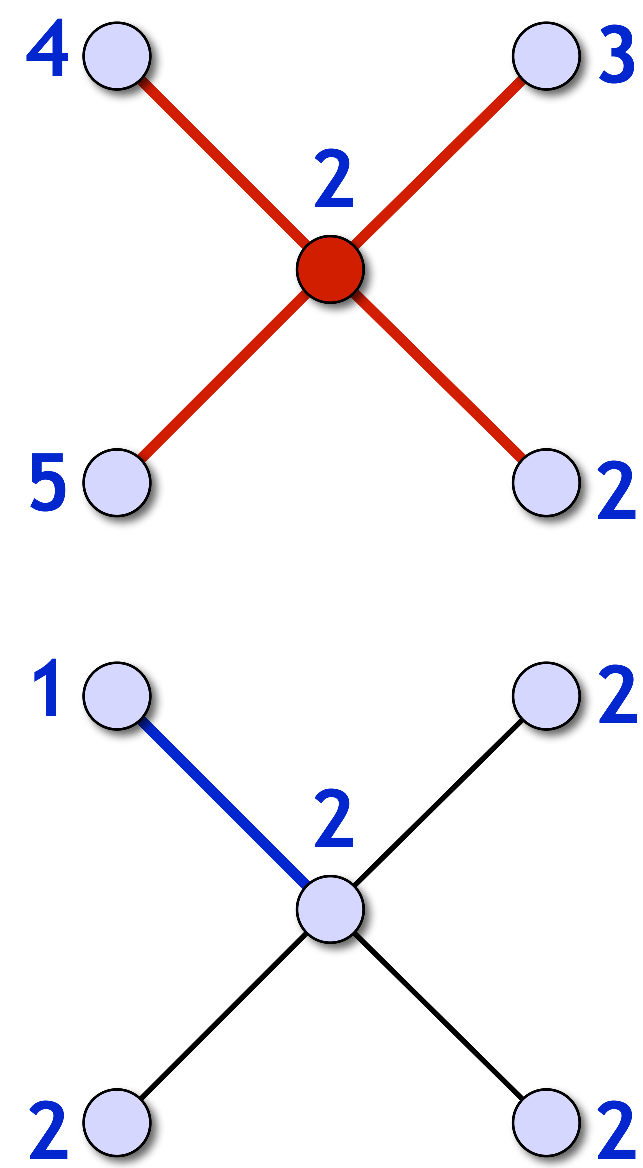
Finding a maximal edge packing: colouring trick

- Offer is a local minimum:
  - Node will be **saturated**
- Otherwise there is a neighbour with a different offer:
  - Interpret the offer sequences as colours
  - Nodes *u* and *v* have different colours: {*u*, *v*} is **multicoloured**



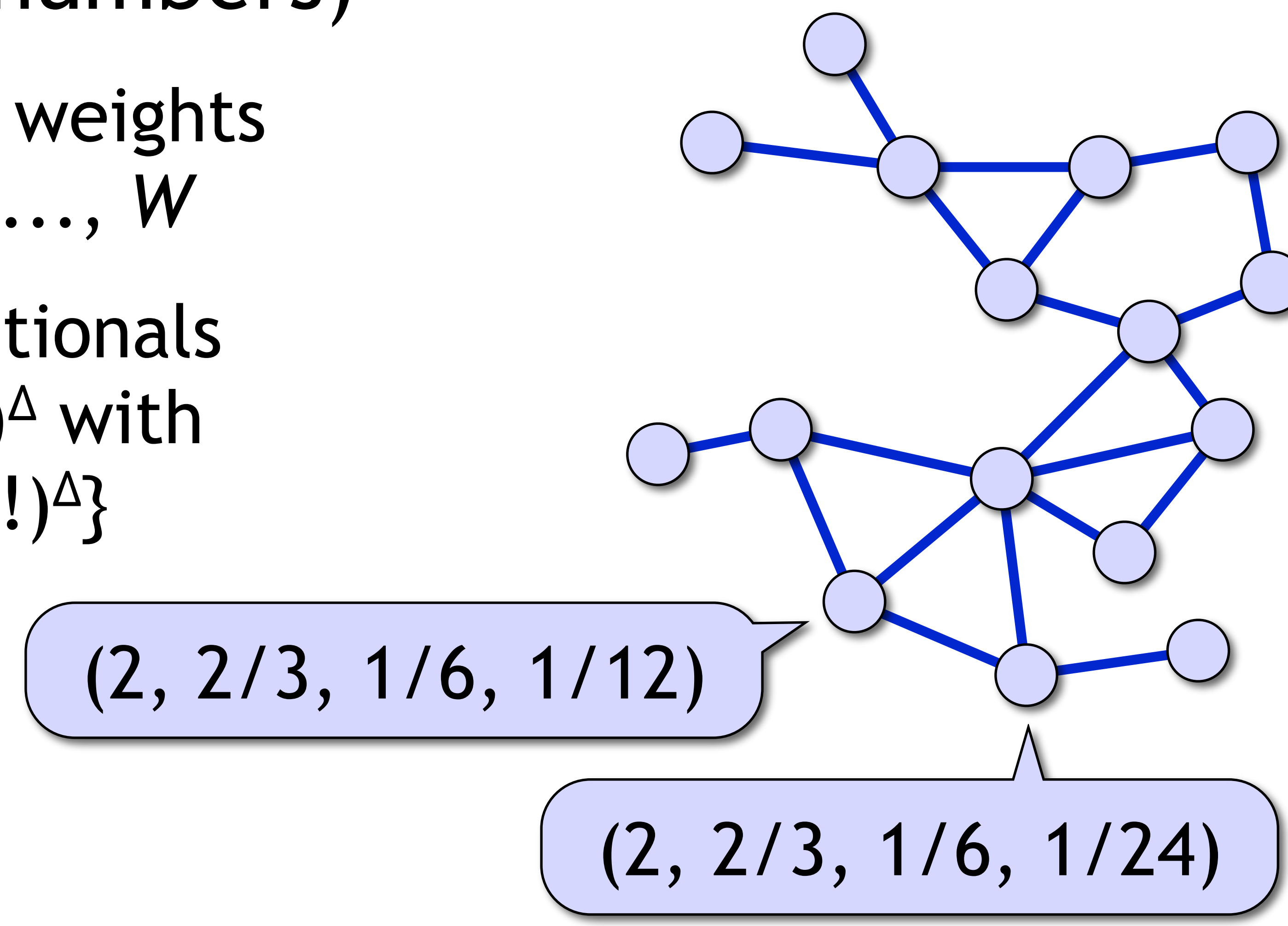
Finding a maximal edge packing: colouring trick

- Progress guaranteed:
  - On each iteration, for each node, at least one incident edge becomes **saturated** or **multicoloured**
  - Such edges are be discarded in phase I; maximum degree Δ decreases by at least one
  - Hence in Δ rounds all edges are saturated or multicoloured



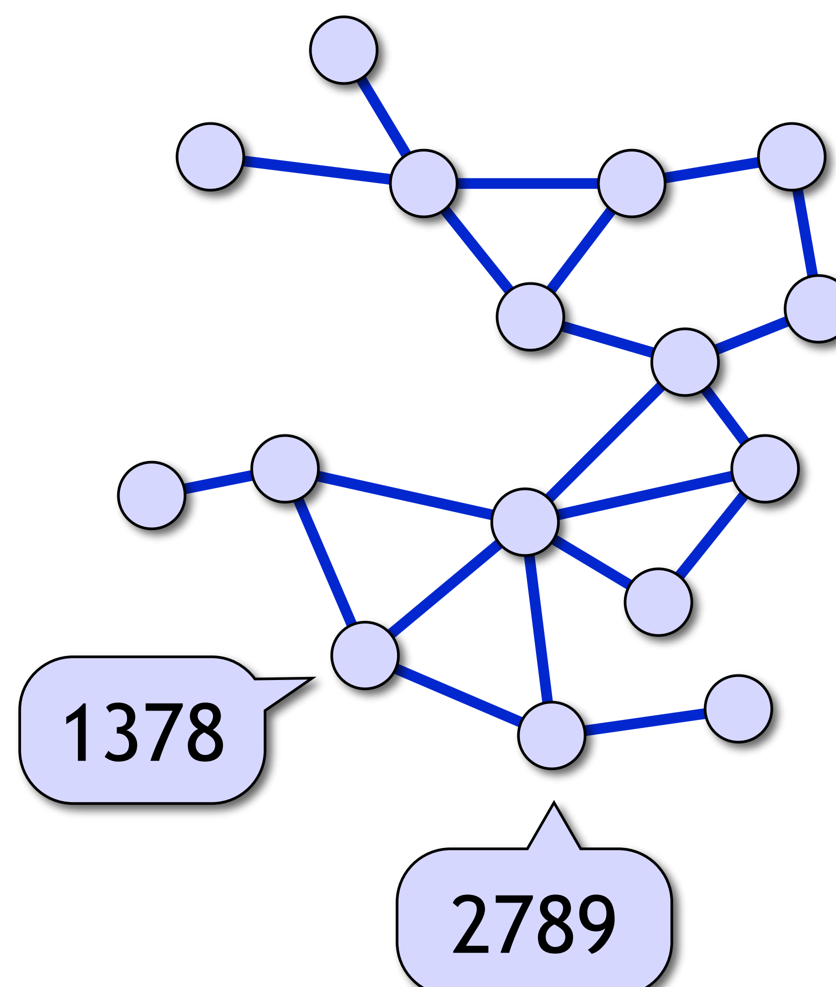
Finding a maximal edge packing: colouring trick

- Colours are sequences of Δ offers (rational numbers)
  - Assume that node weights are integers 1, 2, ..., W
  - Then offers are rationals of the form $q/(\Delta!)^\Delta$ with $q \in \{1, 2, \ldots, W(\Delta!)^\Delta\}$



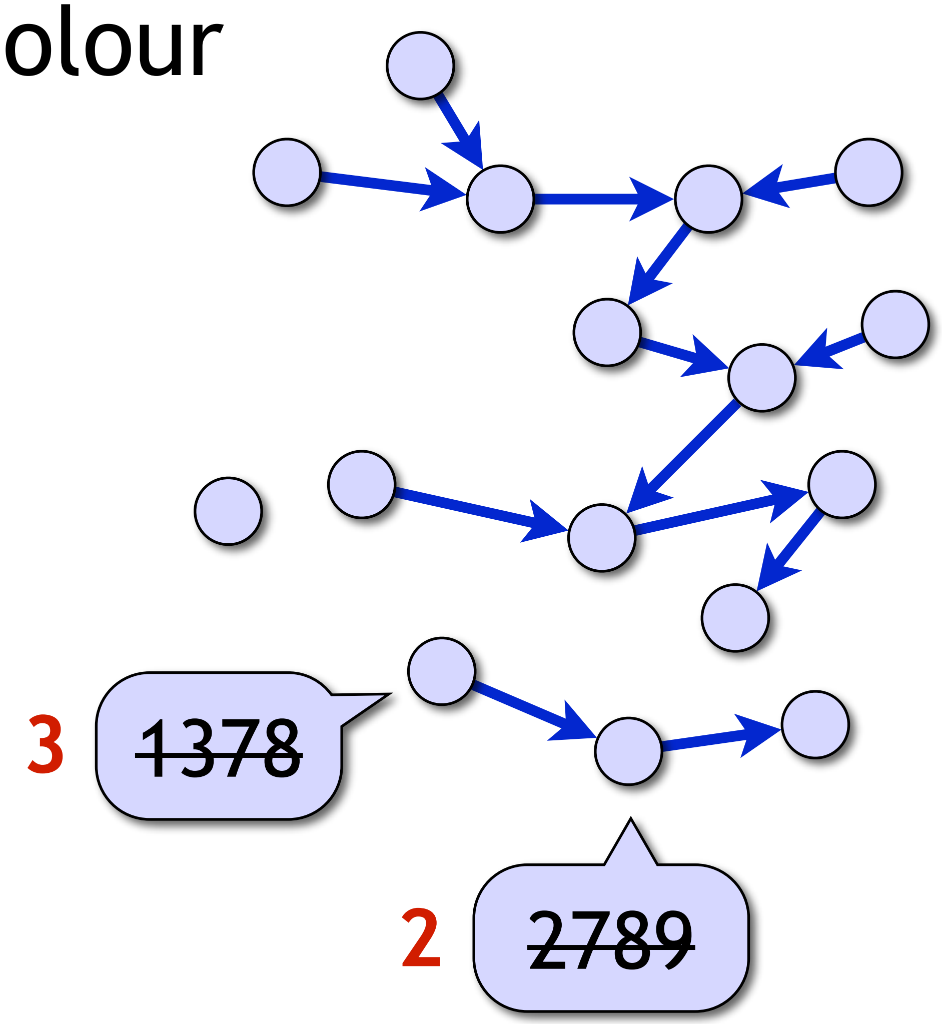
Finding a maximal edge packing: colouring trick

- Colours are sequences of Δ offers (rational numbers)
  - Assume that node weights are integers 1, 2, ..., W
  - Then offers are rationals of the form $q/(\Delta!)^\Delta$ with $q \in \{1, 2, \ldots, W(\Delta!)^\Delta\}$
  - $k = (W(\Delta!)^\Delta)^\Delta$ possible colours, replace with integers 1, 2, ..., k



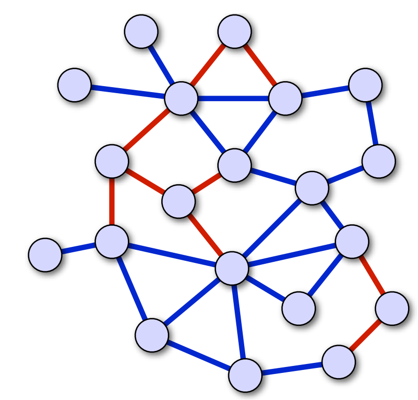
Finding a maximal edge packing: phase II

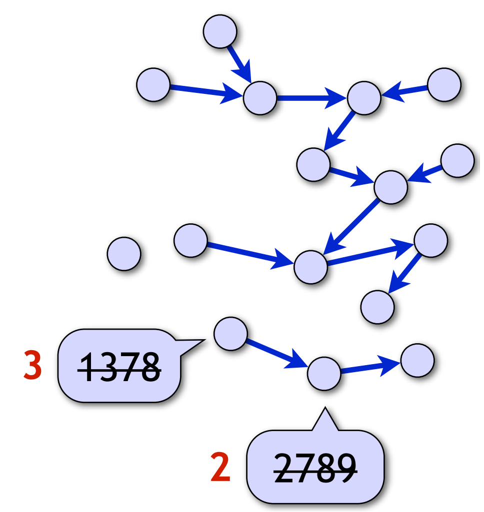
- Proper *k*-colouring of the unsaturated subgraph
- Orient from lower to higher colour
- Partition in Δ forests
  - Use Cole–Vishkin (1986) style colour reduction algorithm
- Use colour classes to saturate edges
- $O(\Delta + \log^* W)$ rounds

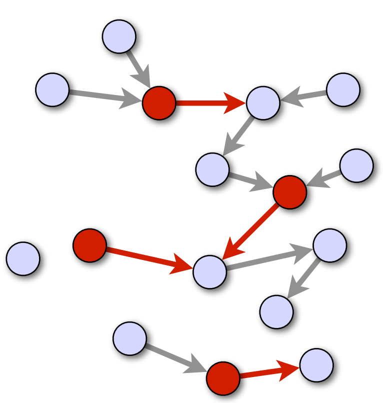


Finding a maximal edge packing: summary

- Maximal edge packing and 2-approximation of vertex cover in time $O(\Delta + \log^* W)$
  - W = maximum node weight
- Unweighted graphs: running time simply $O(\Delta)$, independent of n
- Everything can be implemented in the **port-numbering model**







Vertex cover and set cover in anonymous networks: summary

- 2-approximation of vertex cover in time $O(\Delta)$ in the **port-numbering model**
  - Idea: consider a more general problem, minimum-***weight*** vertex cover
- 2-approximation of vertex cover in time poly(Δ) in the **broadcast model**?
  - Idea: consider a more general problem, minimum-weight ***set cover***!

Take-home messages

- Algorithms that we saw today are **strictly local**
  - Running time independent of the number of nodes
  - Output of a node depends only on its local neighbourhood
  - Very efficient, can be used in arbitrarily large networks
  - Deterministic, highly fault-tolerant
- There are non-trivial graph problems that can be solved with strictly local algorithms!
  - More: www.cs.helsinki.fi/jukka.suomela/local-survey