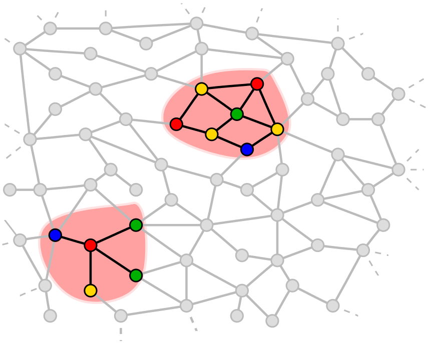


Exact bounds for distributed graph colouring

Joel Rybicki
Helsinki Institute of Information Technology HIIT
Department of Computer Science, University of Helsinki

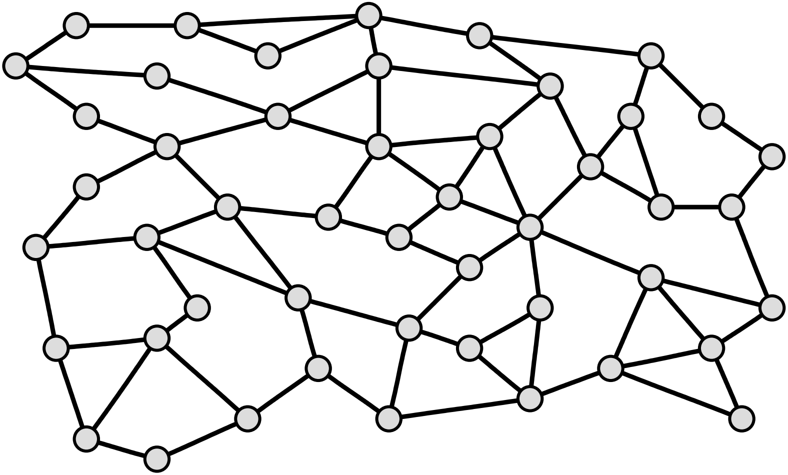
Federated Computer Science Event, Helsinki

29 May, 2012

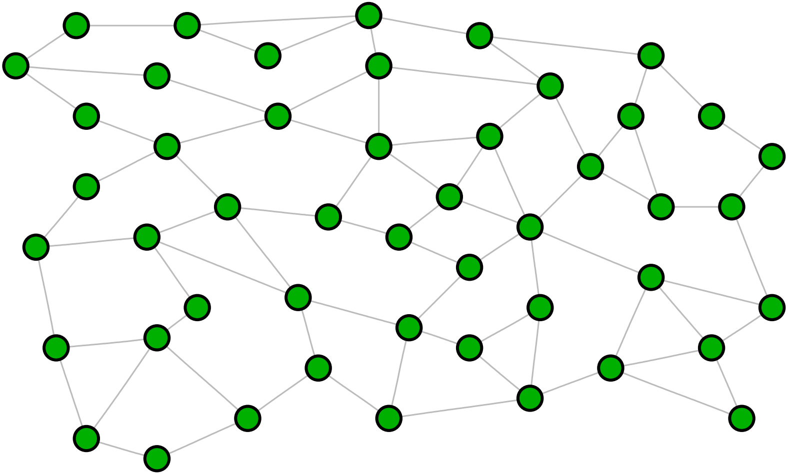
Outline

1. What is distributed computing?
2. The graph colouring problem
3. The model of distributed computing
4. Attaining exact bounds: Techniques and results

The distributed setting

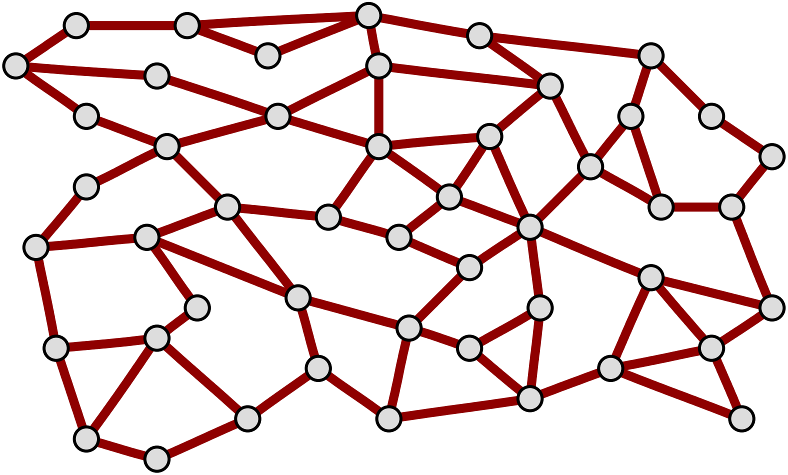


The distributed setting



Nodes = processors

The distributed setting



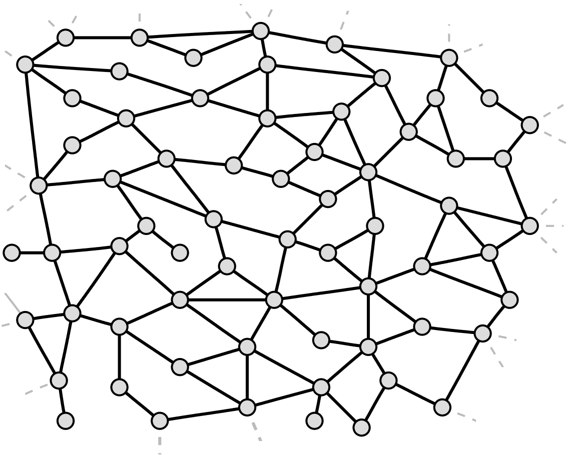
Nodes = processors, **edges = communication links**

Theory of distributed computing

Theory of distributed computing studies algorithmic problems in large networks.

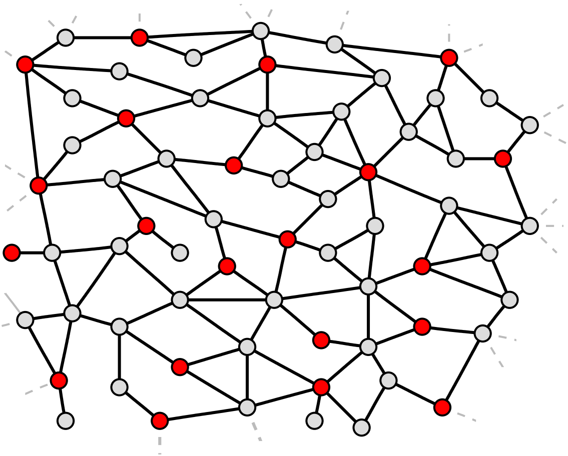
- All processors run the same algorithm
- The network itself is the input
- Each node outputs its own part in the solution

Problem examples



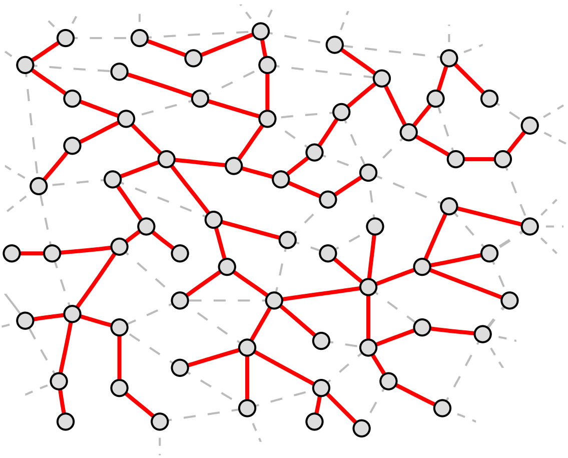
Focus on graph problems

Problem examples



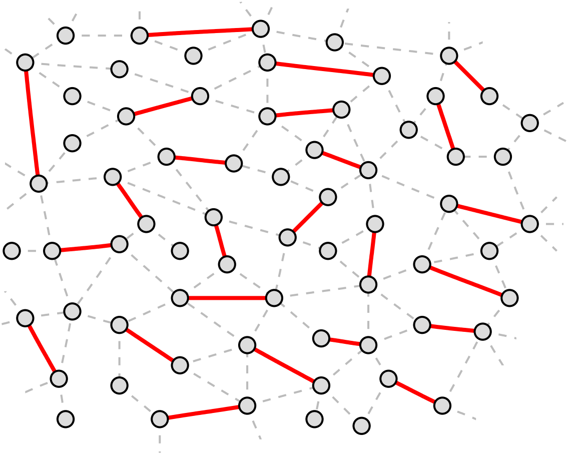
Dominating sets

Problem examples



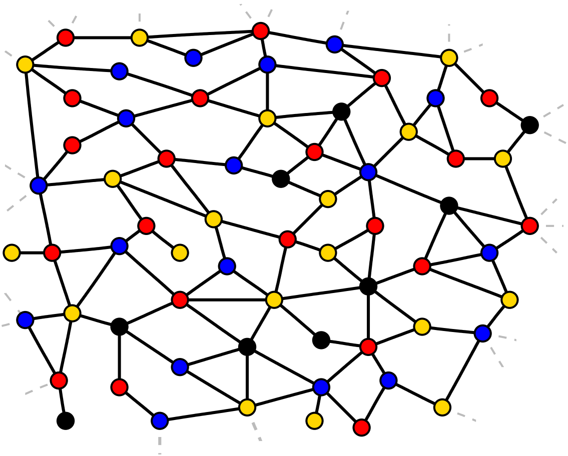
Spanning trees

Problem examples



Matchings

Problem examples

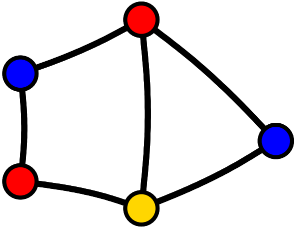


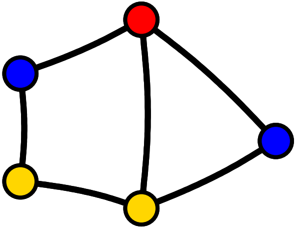
Graph colouring

Graph colouring

The task is to give each node a colour such that

- colours = numerical labels
- adjacent nodes have different colours
- the total number of colours is small

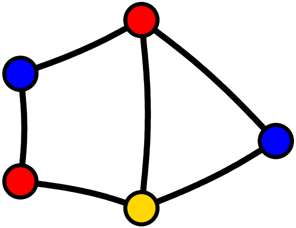


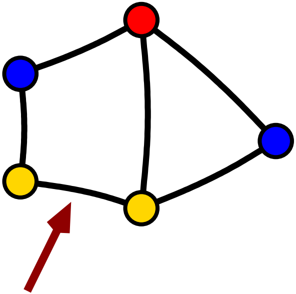


Graph colouring

The task is to give each node a colour such that

- colours = numerical labels
- adjacent nodes have different colours
- the total number of colours is small

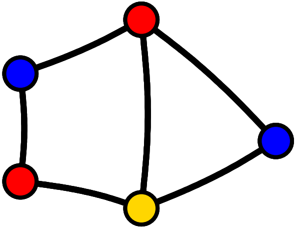


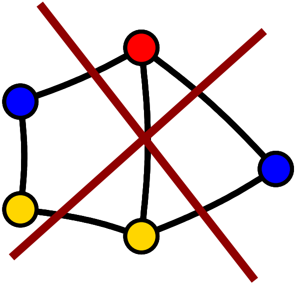


Graph colouring

The task is to give each node a colour such that

- colours = numerical labels
- adjacent nodes have different colours
- the total number of colours is small



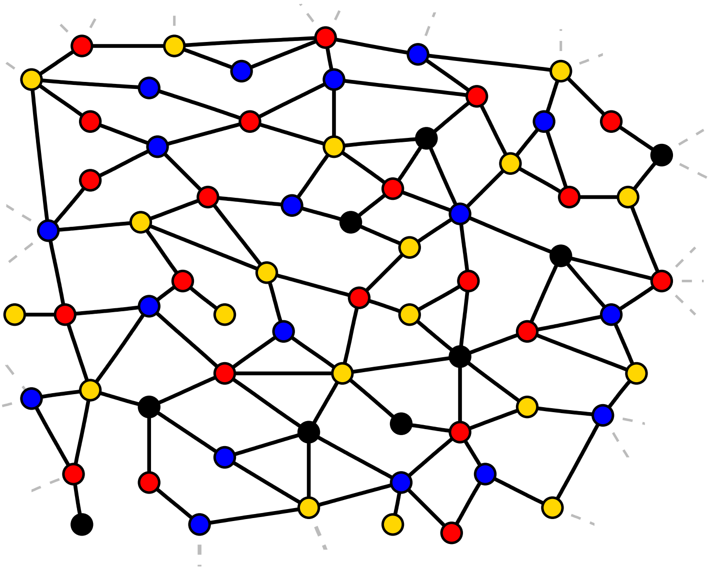


Application: Colourings as schedules

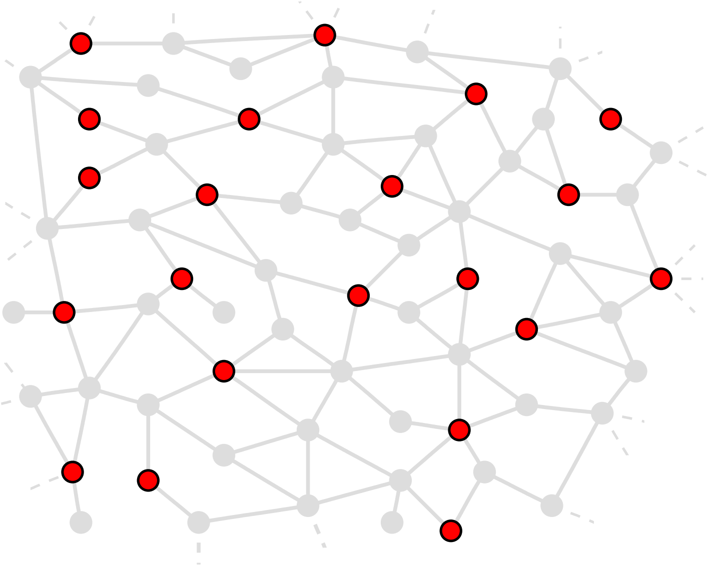
Suppose we have a wireless sensor network..

- all nodes are equipped with radio transmitters
- near-by nodes cannot transmit simultaneously due to interference

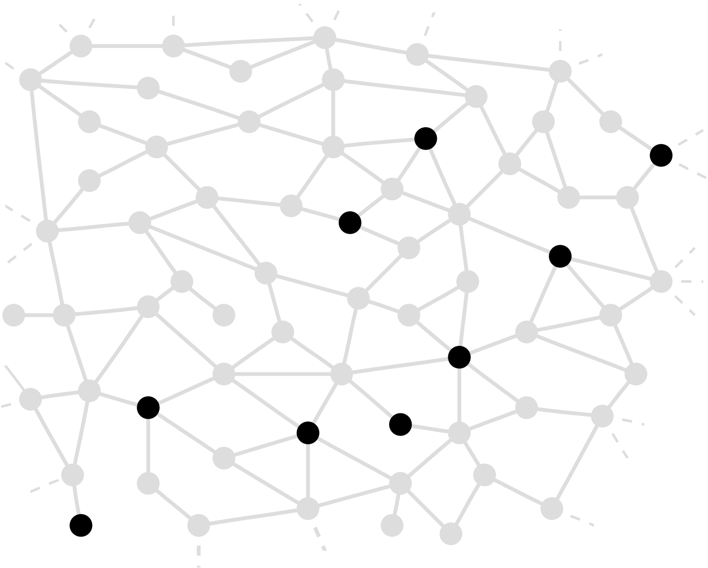
Application: Colourings as schedules



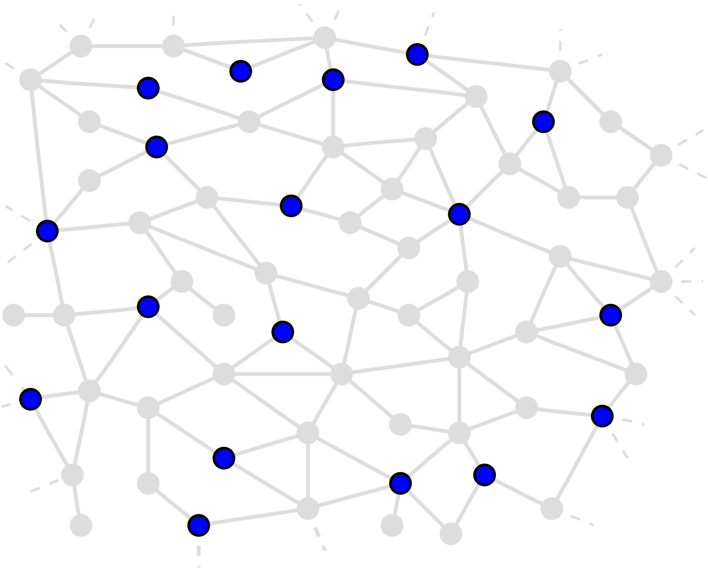
Application: Colourings as schedules



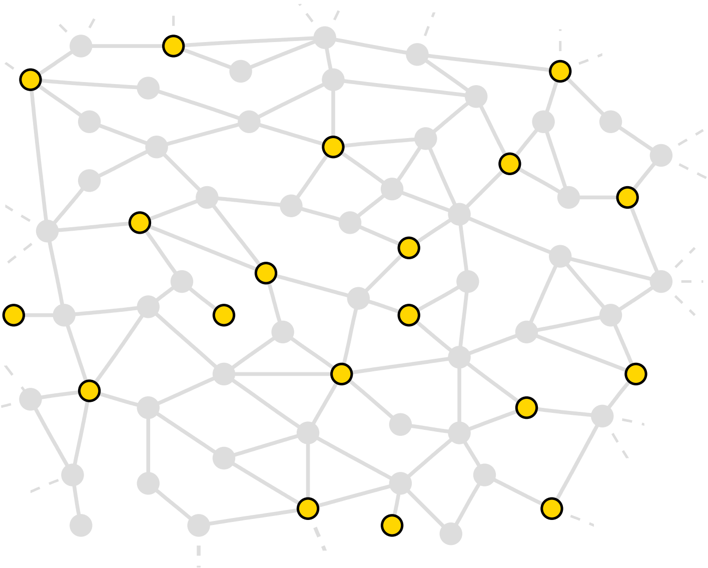
Application: Colourings as schedules



Application: Colourings as schedules



Application: Colourings as schedules

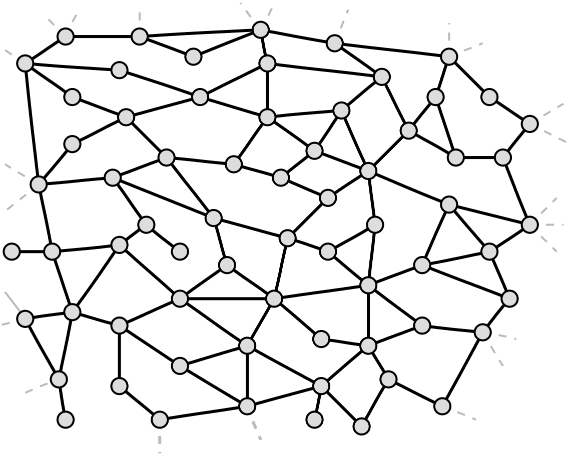


Locality

Locality is an important theme in distributed computing:

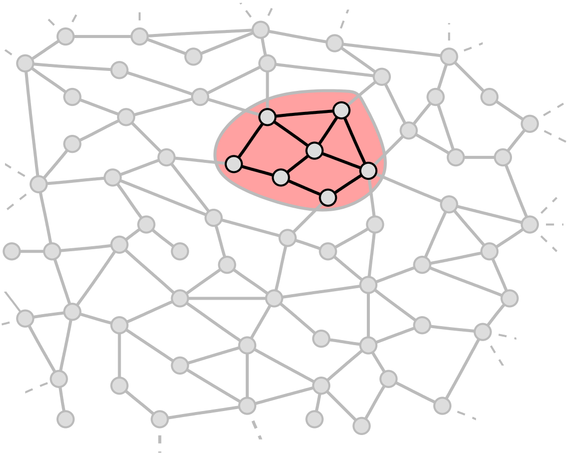
- Many problems are global in nature (e.g., optimal colouring)
- But an efficient distributed algorithm cannot depend on global knowledge.

Locality



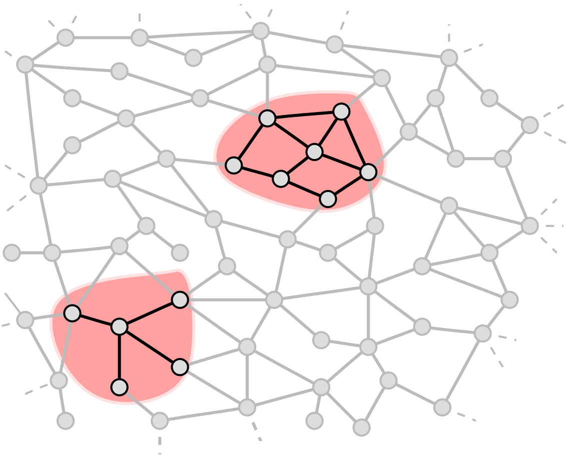
The aim is to find **global** solutions using **local** information.

Locality



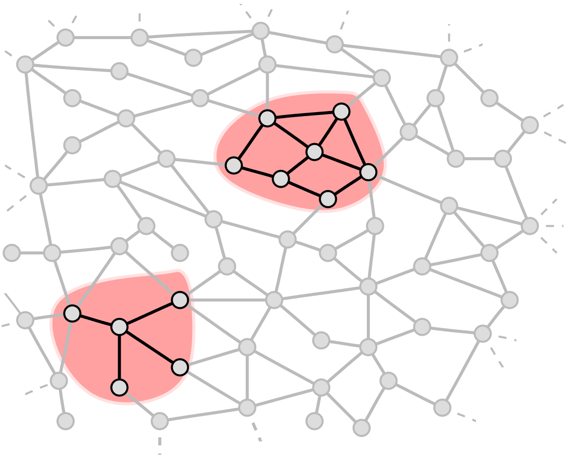
The aim is to find **global** solutions using **local** information.

Locality



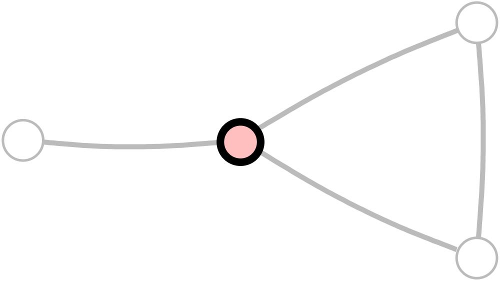
Minimal dependence with others

Locality



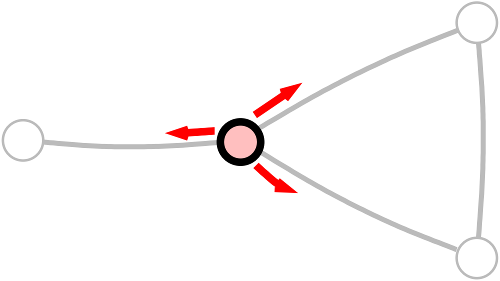
Think global, act local

Model of distributed computation



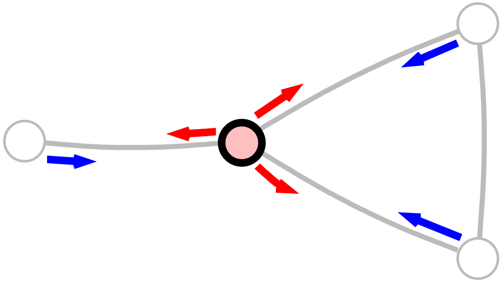
The system proceeds in discrete synchronous communication rounds.

Model of distributed computation



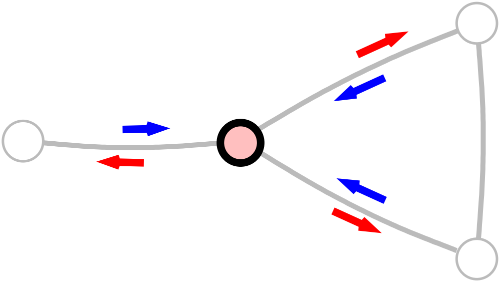
1. Send messages

Model of distributed computation



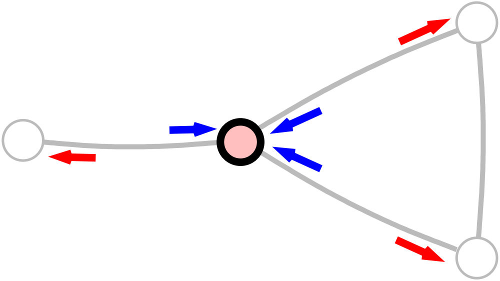
1. Send messages

Model of distributed computation



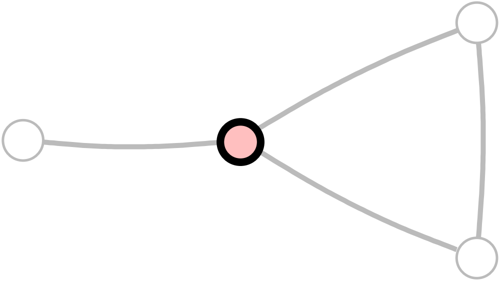
2. Wait for the messages to propagate

Model of distributed computation



3. Receive messages

Model of distributed computation



4. Perform local computation. Continue or declare output.

Model of distributed computation

During a single communication round each node

1. sends messages to neighbours
2. waits for the messages to propagate
3. receives messages from neighbours
4. performs local computation

Repeat until all nodes have declared their output.

> Time complexity = Number of communication rounds

Model of distributed computation

During a single communication round each node

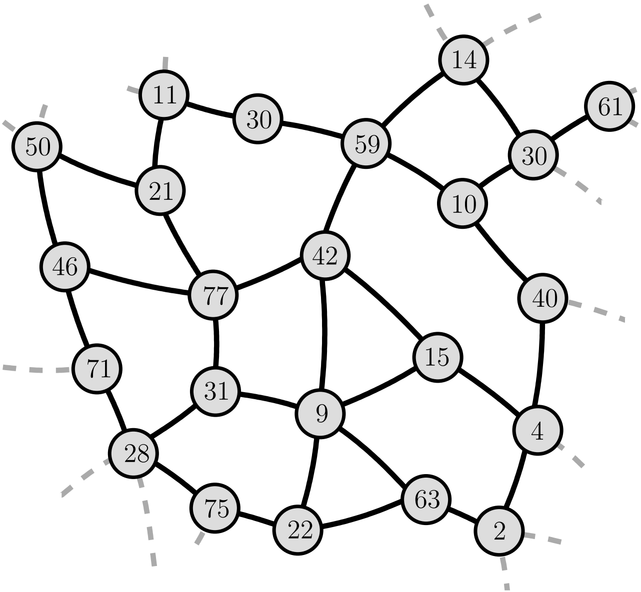
1. sends messages to neighbours
2. waits for the messages to propagate
3. receives messages from neighbours
4. performs local computation

Repeat until all nodes have declared their output.

Time complexity = Number of communication rounds

Space complexity = Size of sent messages

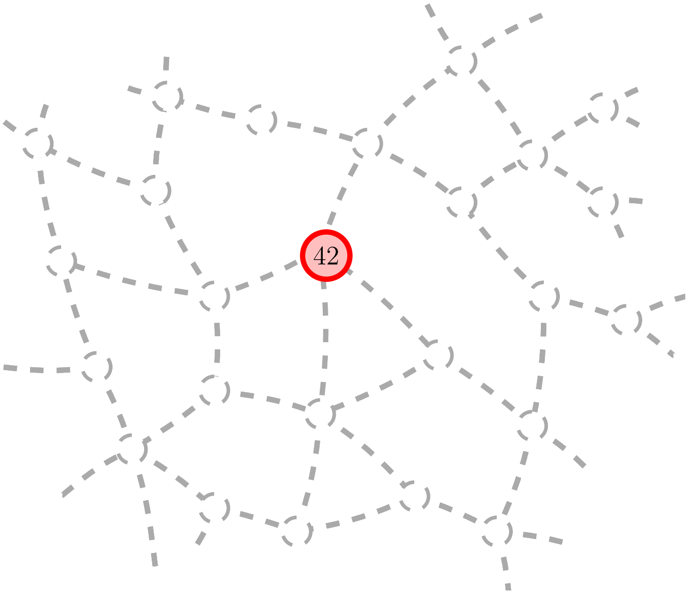
Unique identifiers



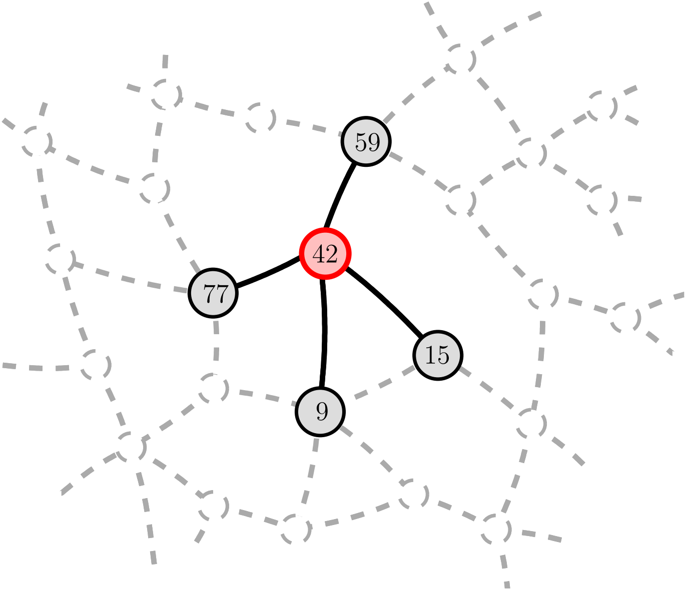
Local views

We can use so-called local views to reason about distributed algorithms.

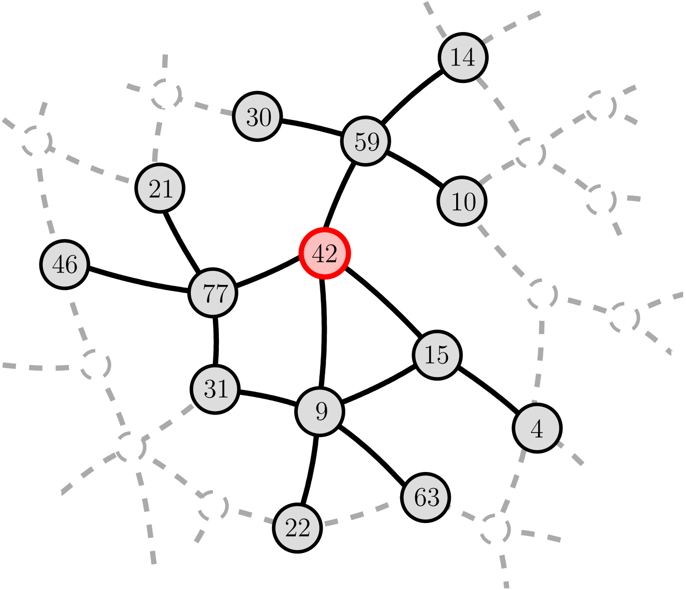
Local views (0 rounds)



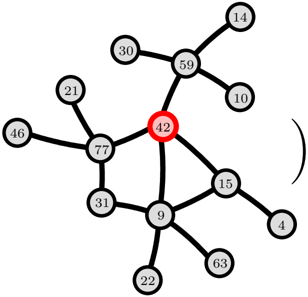
Local views (1 round)



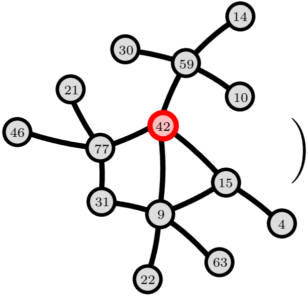
Local views (2 rounds)



Algorithms as mappings

$f\big($  $\big) \in \{0, 1, \ldots, k-1\}$

Algorithms as mappings

$f($ 

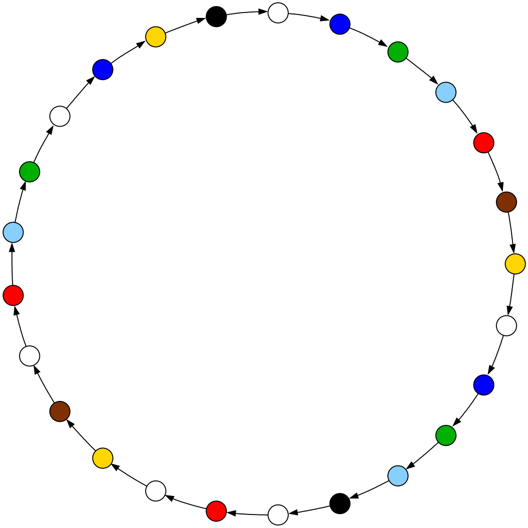
$) \in \{0, 1, \ldots, k-1\}$

output values

radius-2 neighbourhood = 2 rounds

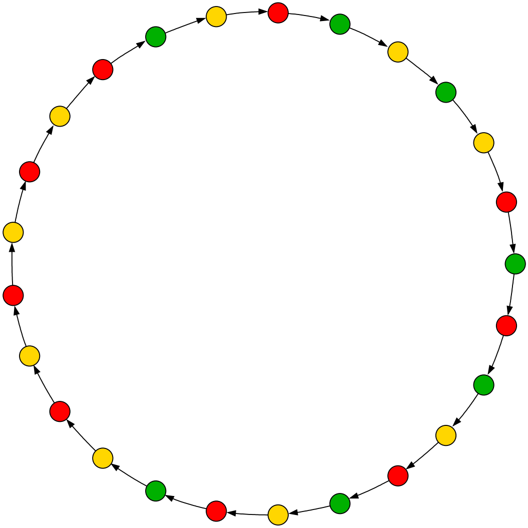
Colour reduction in directed cycles

Input: a k-coloured directed cycle (UIDs = colouring)



Colour reduction in directed cycles

Output: a 3-colouring



Log-star

The $\log^*$ function appears often in distributed computing.

Definition: $\log^* k = \min\{i : \overbrace{\log\big(\cdots\log(k)\big)}^{i\text{ times}} \leq 1\}$

Log-star

The $\log^*$ function appears often in distributed computing.

Definition: $\log^* k = \min\{i : \overbrace{\log\left(\cdots \log(k)\right)}^{i \text{ times}} \leq 1\}$

The $\log^*$ function grows *very* slowly:

- $\log^* 2 = 1$
- $\log^* 4 = 2$
- $\log^* 16 = 3$
- $\log^* 2^{16} = 4$
- $\log^* 2^{65536} = 5$

Colouring directed cycles

The work focuses on 3-colouring directed cycles:

- A fundamental problem in distributed computing
- Always possible in $O(\log^* k)$ rounds (Cole and Vishkin 1986)
- Cannot be done in $o(\log^* k)$ rounds (Linial 1987)

What is the *exact* complexity of the problem?

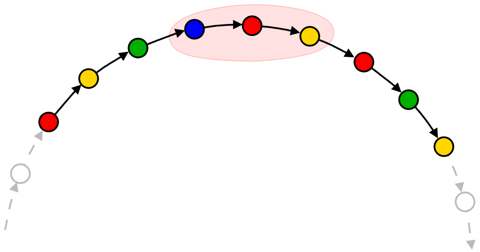
The key idea: neighbourhood graphs

Colourability of certain graphs $\iff$ existence of distributed colouring algorithms:

- finding optimal colourings gives optimal algorithms

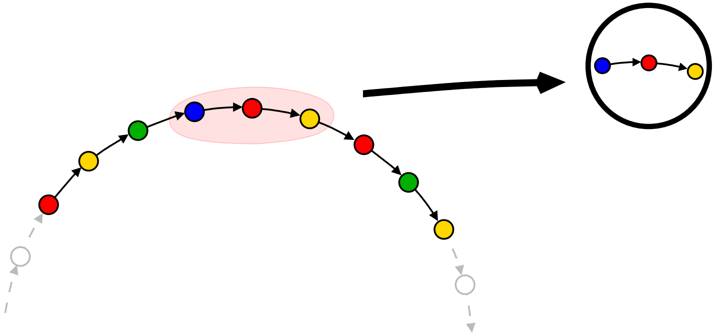
Solving finite combinatorial problems tells us how fast distributed algorithms exist!

The neighbourhood graph construction

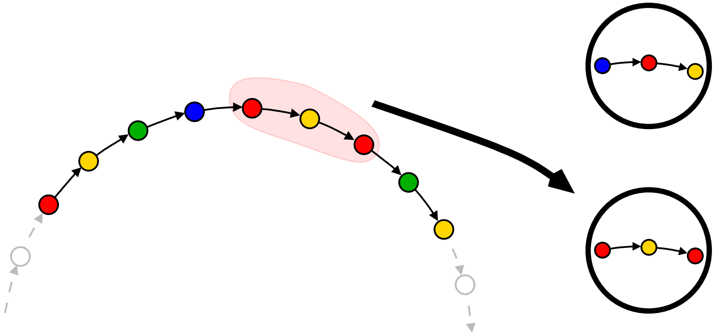


Example: Consider a k-coloured cycle and radius-1 neighbourhoods.

Neighbourhood graphs construction

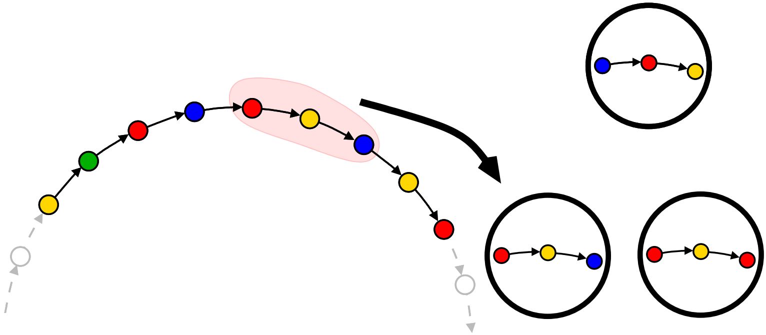


Neighbourhood graphs: nodes



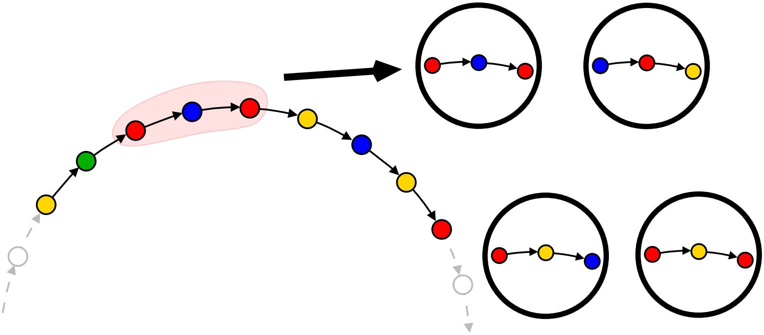
Make a node of each neighbourhood in a coloured cycle.

Neighbourhood graphs: nodes



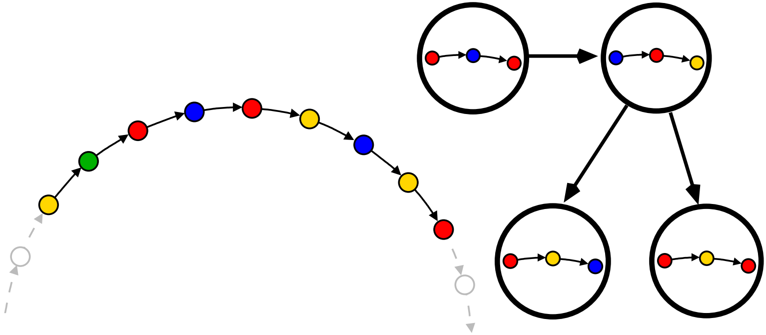
There are many possible neighbourhoods that could occur (”different worlds”).

Neighbourhood graphs: nodes



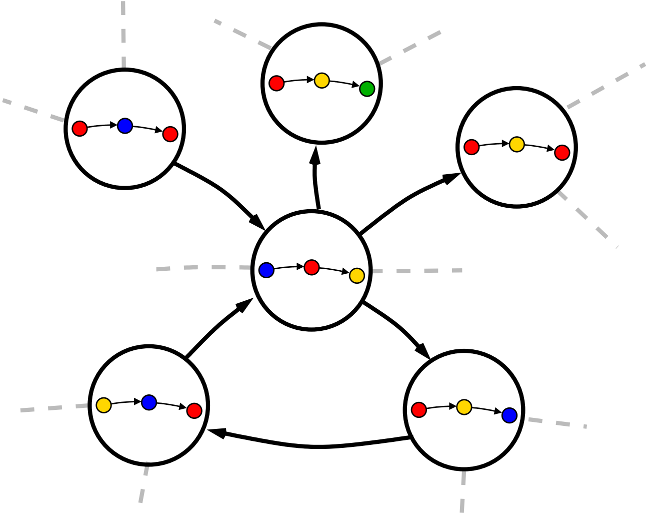
Keep adding all the neighbourhoods.

Neighbourhood graphs: edges

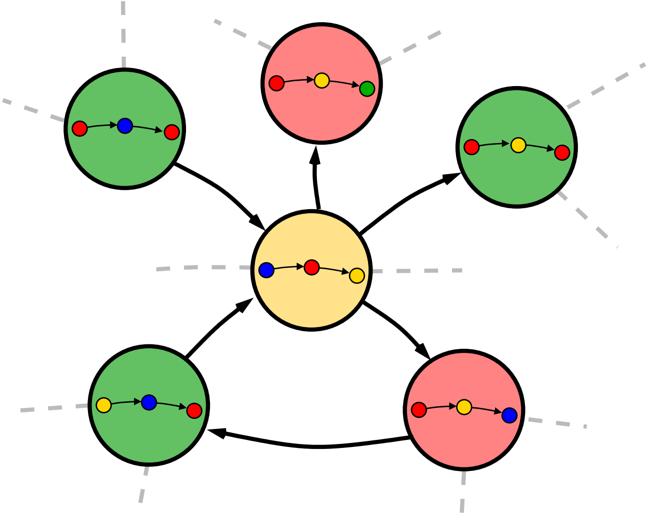


Finally, connect neighbourhoods that can be adjacent.

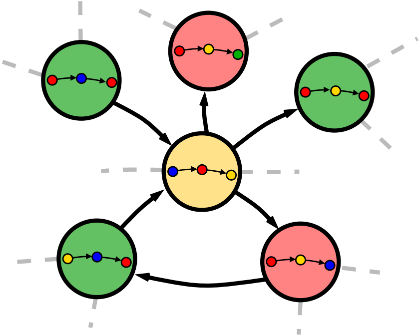
Neighbourhood graphs

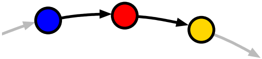


From neighbourhood graphs to algorithms

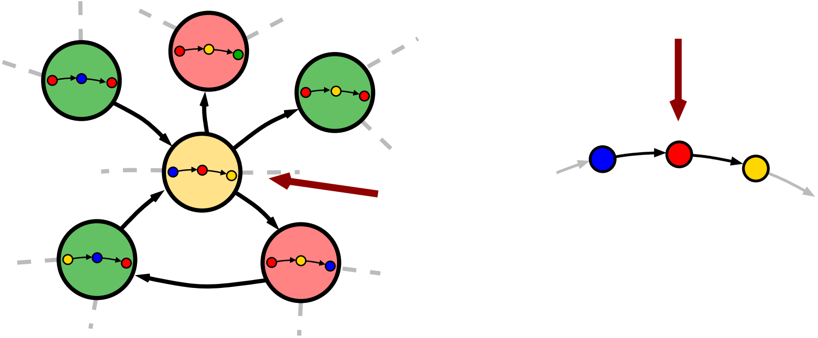


From neighbourhood graphs to algorithms

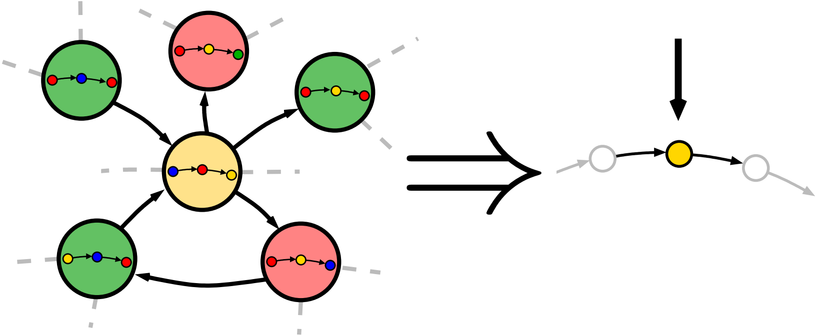




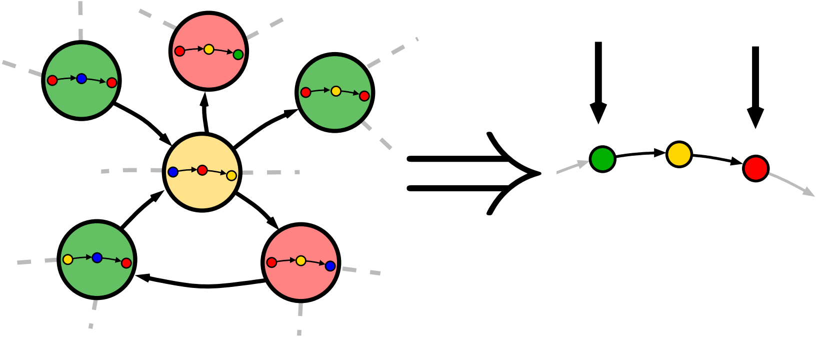
From neighbourhood graphs to algorithms



From neighbourhood graphs to algorithms



From neighbourhood graphs to algorithms



Exact bounds

Complexity of 3-colouring k-coloured cycles:

	Before	**Now**
Positive:	$\frac{1}{2}(\log^* k + 7)$	$\frac{1}{2}(\log^* k + 3)$

Exact bounds

Complexity of 3-colouring k-coloured cycles:

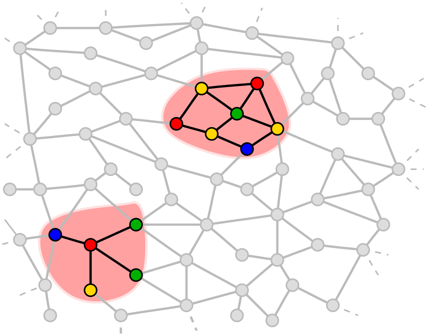
	Before	**Now**
Positive:	$\frac{1}{2}(\log^* k + 7)$	$\frac{1}{2}(\log^* k + 3)$
Negative:	$\frac{1}{2}(\log^* k - 3)$	$\frac{1}{2}(\log^* k + 1)$

- Earlier positive results follow from Cole and Vishkin (1986).
- Previous negative result due to Linial (1992).

Summary

In summary:

- distributed computing studies what can be computed efficiently in large networks
- efficient algorithms are local
- graph colouring is inherently global but can be computed with little communication
- existence of distributed algorithms can posed as a combinatorial problem



Thank you!