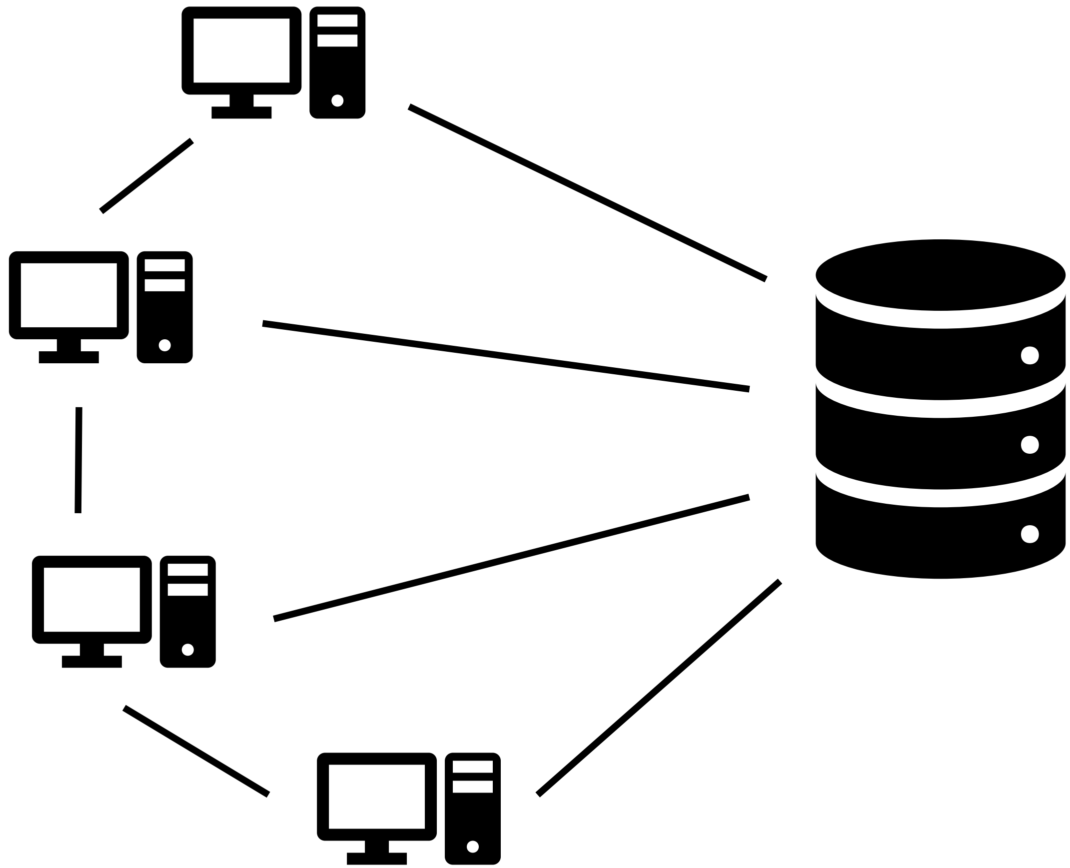
Election vs. Selection: How Much Advice is Needed to Find the Largest Node in a Graph?

Avery Miller
University of Manitoba
avery@averymiller.ca

Andrzej Pelc
Université du Québec en Outaouais
andrzej.pelc@uqo.ca

First presented at SPAA 2016
Presentation - Damien Aymon, 08.05.2018
ETHZ - Seminar in Distributed Computing FS 2018

Application – Shared Resource

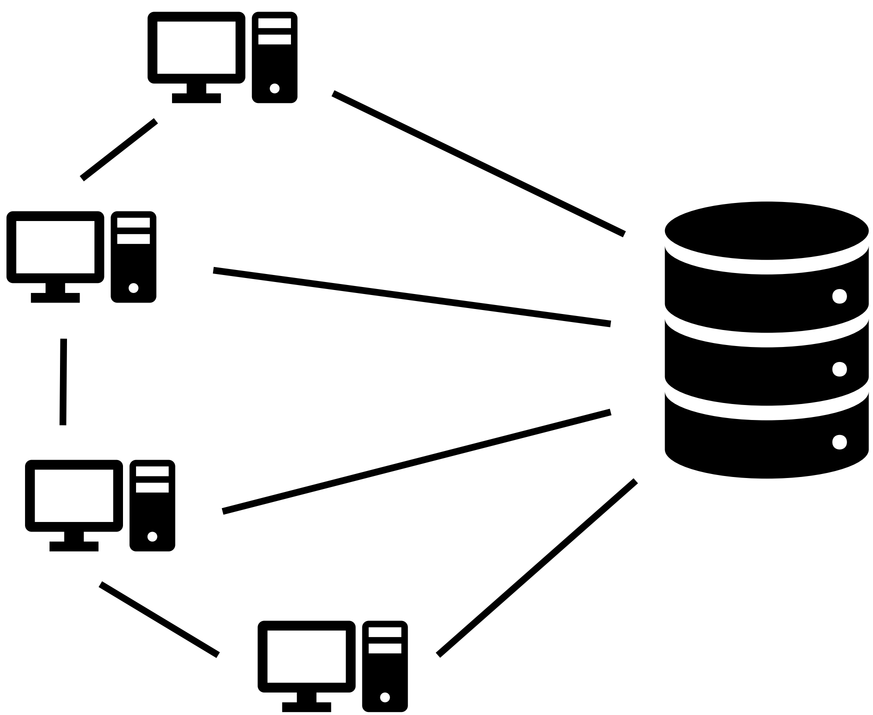


Microsoft Docs, **Leader Election Pattern,** 23.06.2017
https://docs.microsoft.com/en-us/azure/architecture/patterns/leader-election

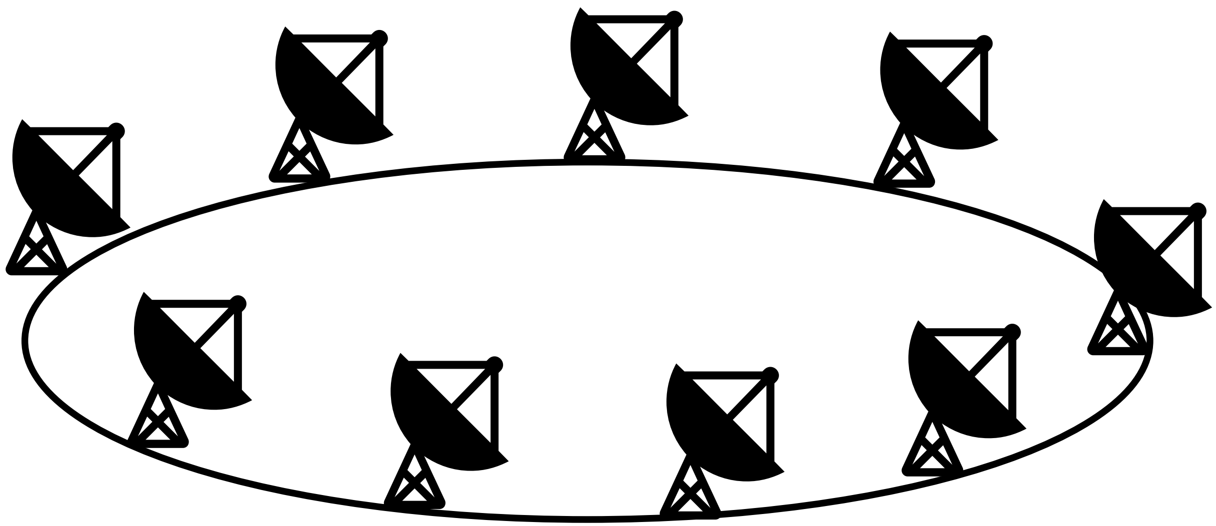
Application – Shared Resource

Access to shared resource -> need coordinator

Failure resilience -> need new leader

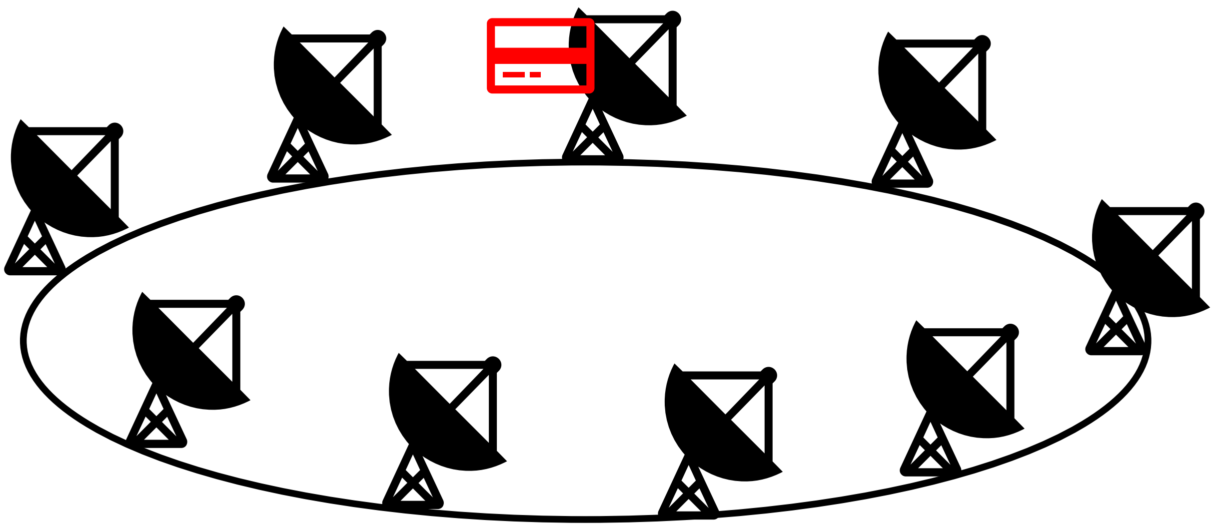


Application – Radiocom

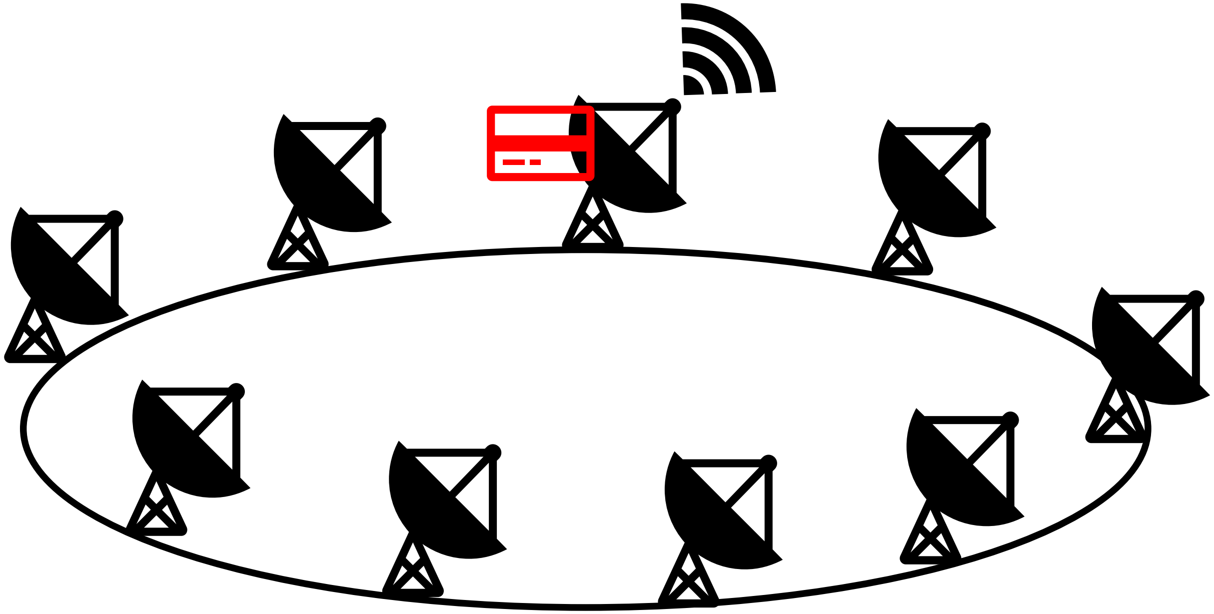


G. Le Lann, **Distributed Systems - Towards a Formal Approach**
Proc. IFIP Congress, 1977, 155-160, North Holland.

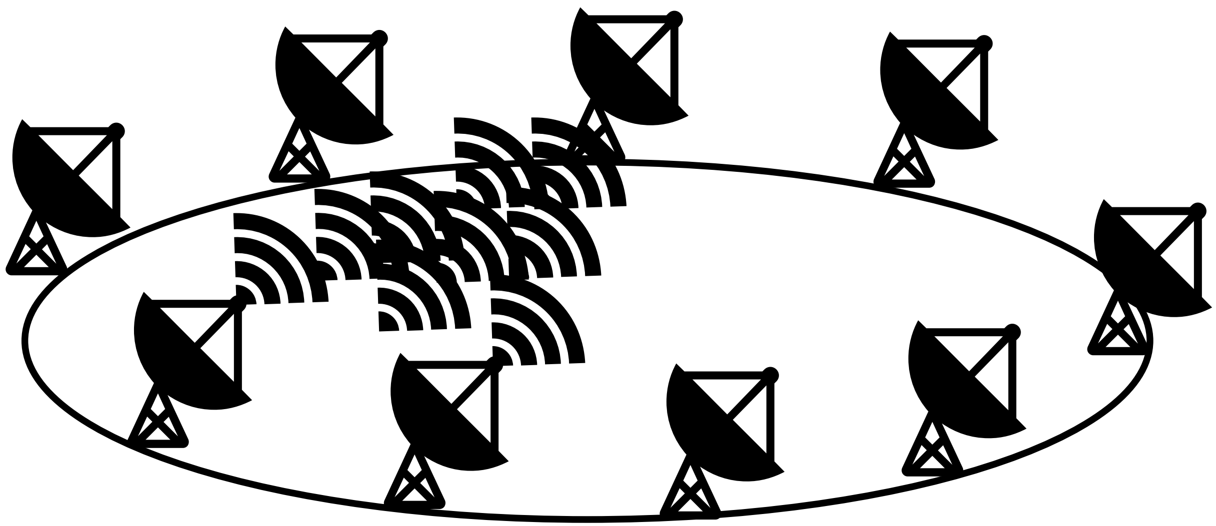
Application – Radiocom



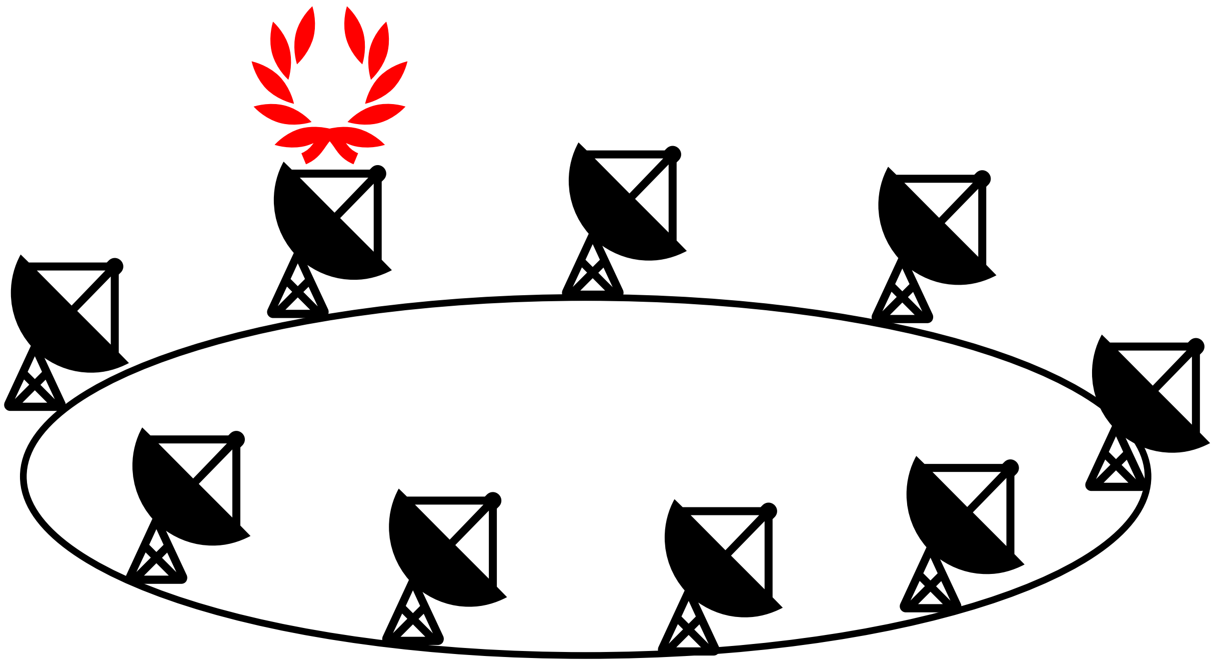
Application – Radiocom



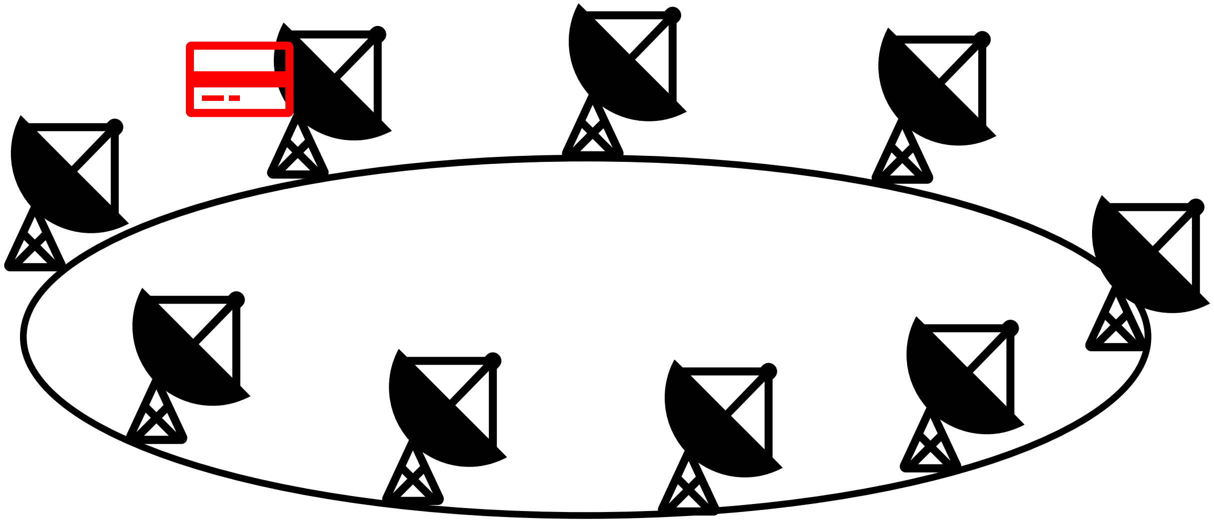
Application – Radiocom



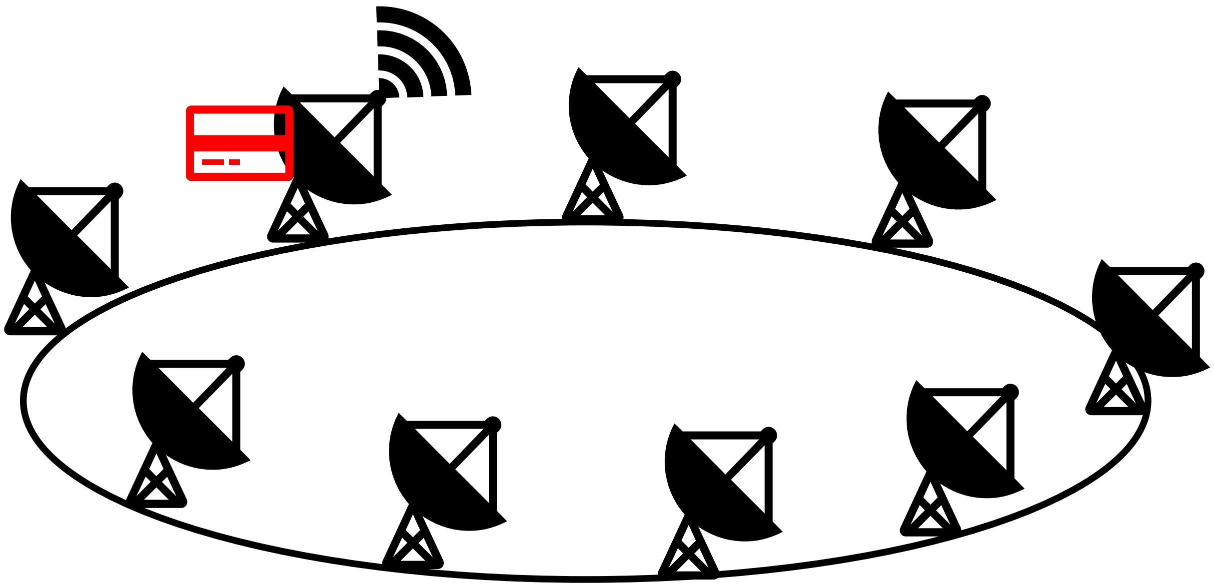
Application – Radiocom



Application – Radiocom



Application – Radiocom



Election vs. Selection: How Much Advice is Needed to Find the Largest Node in a Graph?

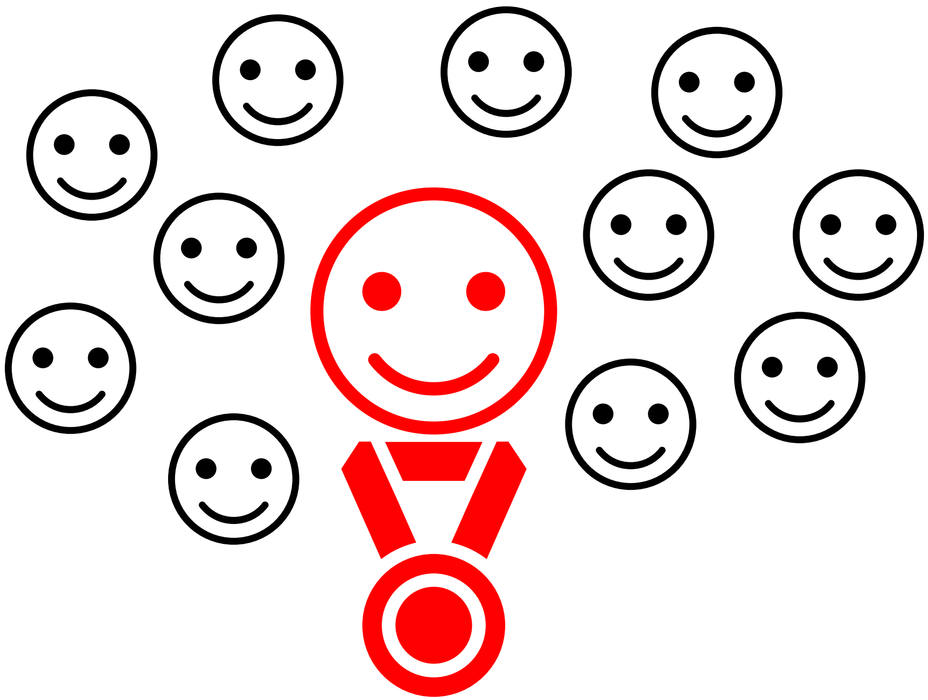
Avery Miller
University of Manitoba
avery@averymiller.ca

Andrzej Pelc*
Université du Québec en Outaouais
andrzej.pelc@uqo.ca

Election

Find a **leader**

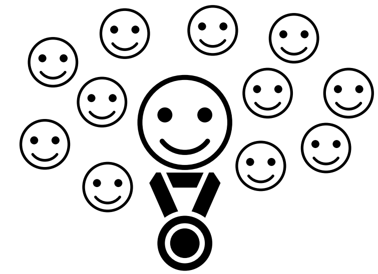
Everyone knows its identity

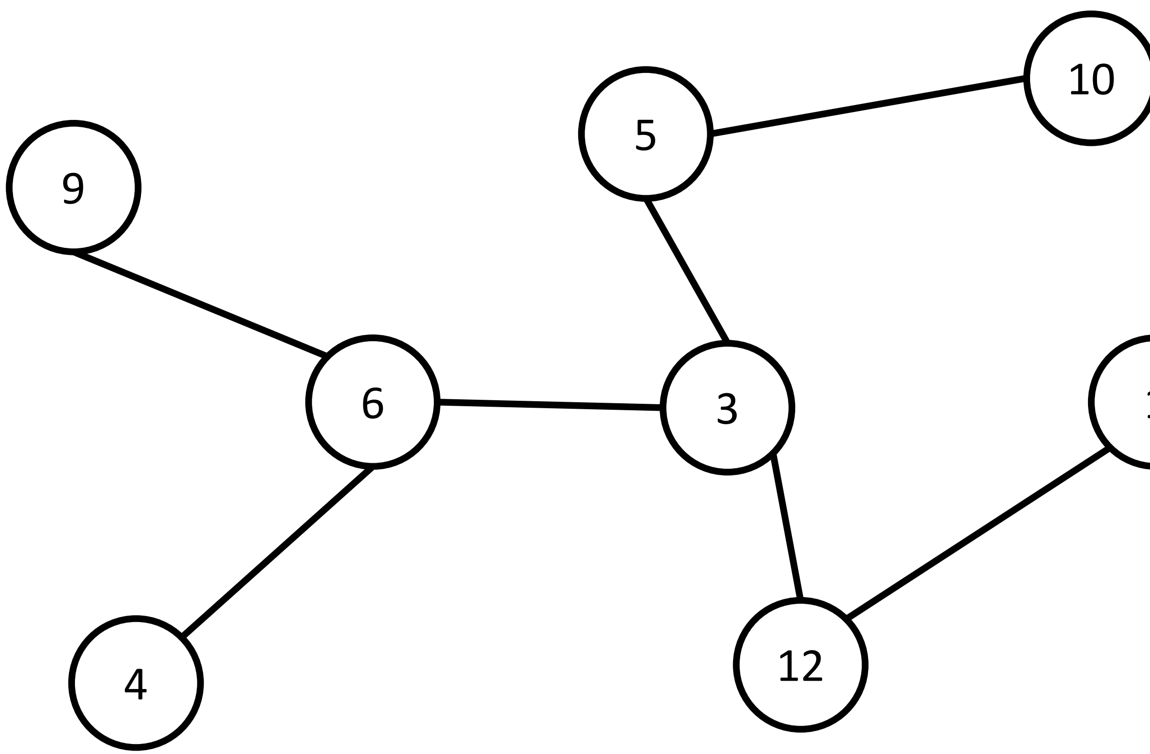


Election

Find a **leader**

Everyone knows its identity

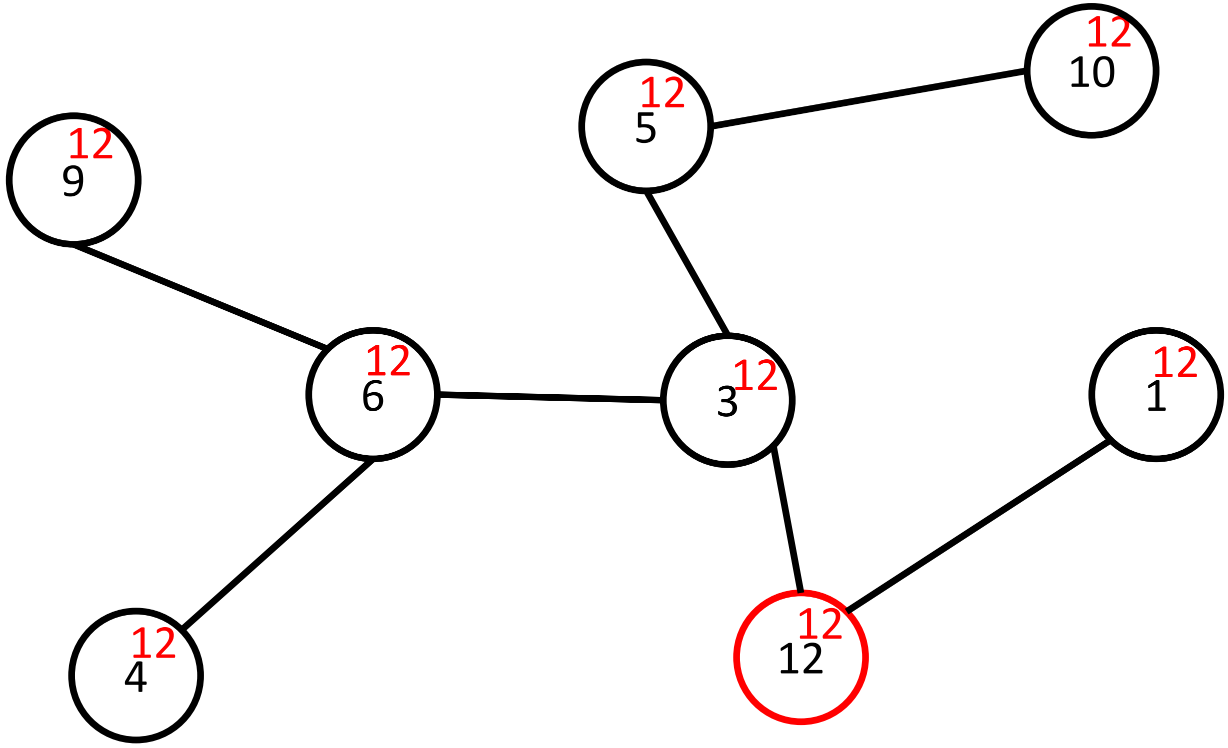




Election

Find a **leader**

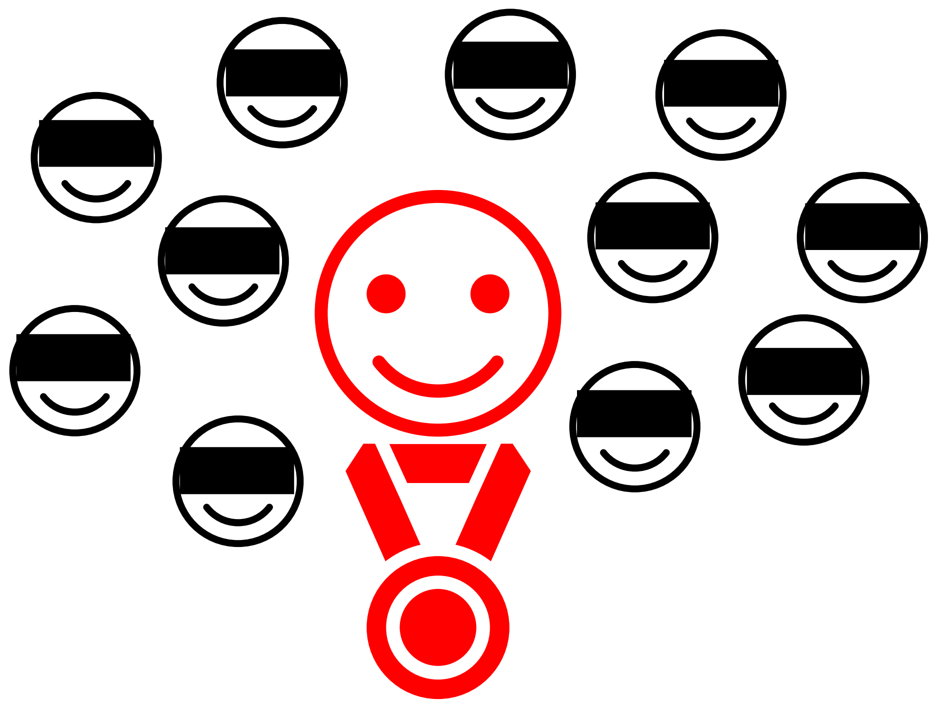
Everyone knows its identity



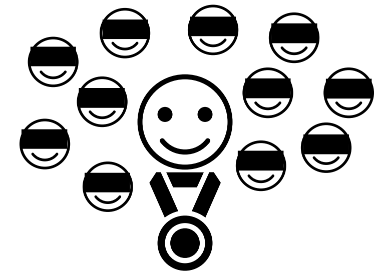
Selection

Leader outputs **1**

Every other node outputs **0**

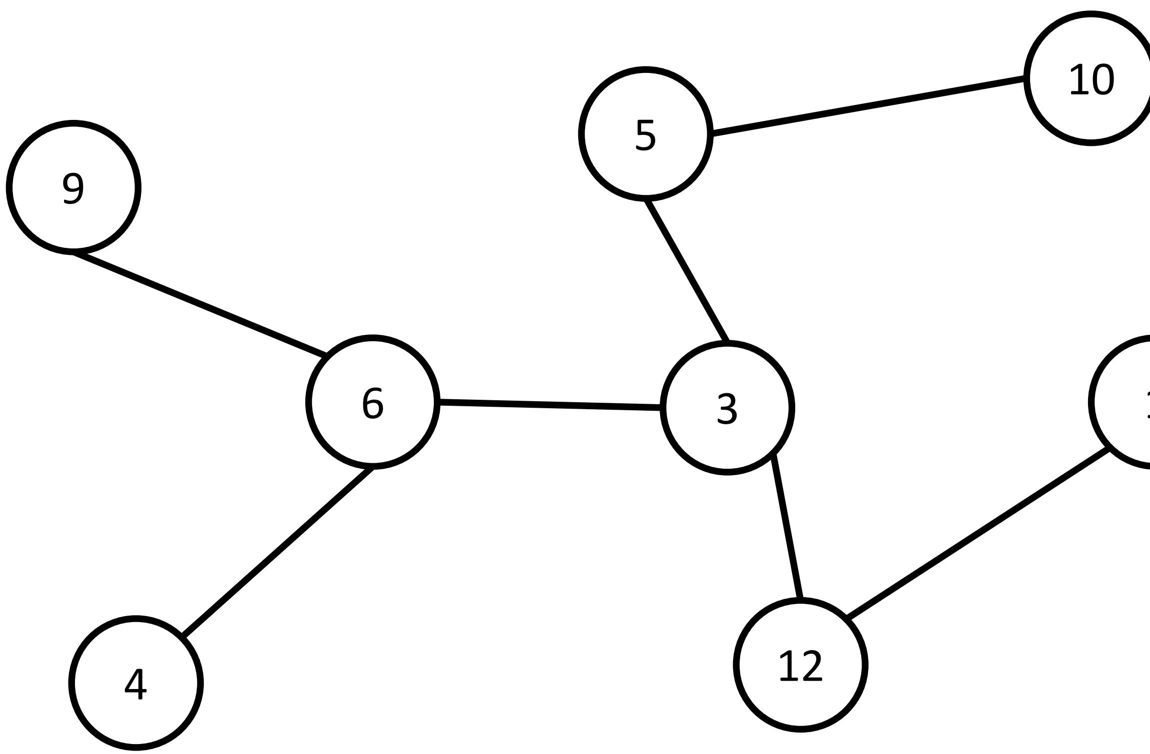


Selection

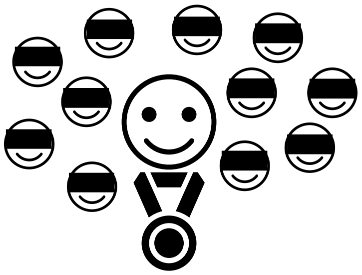


Leader outputs **1**

Every other node outputs **0**

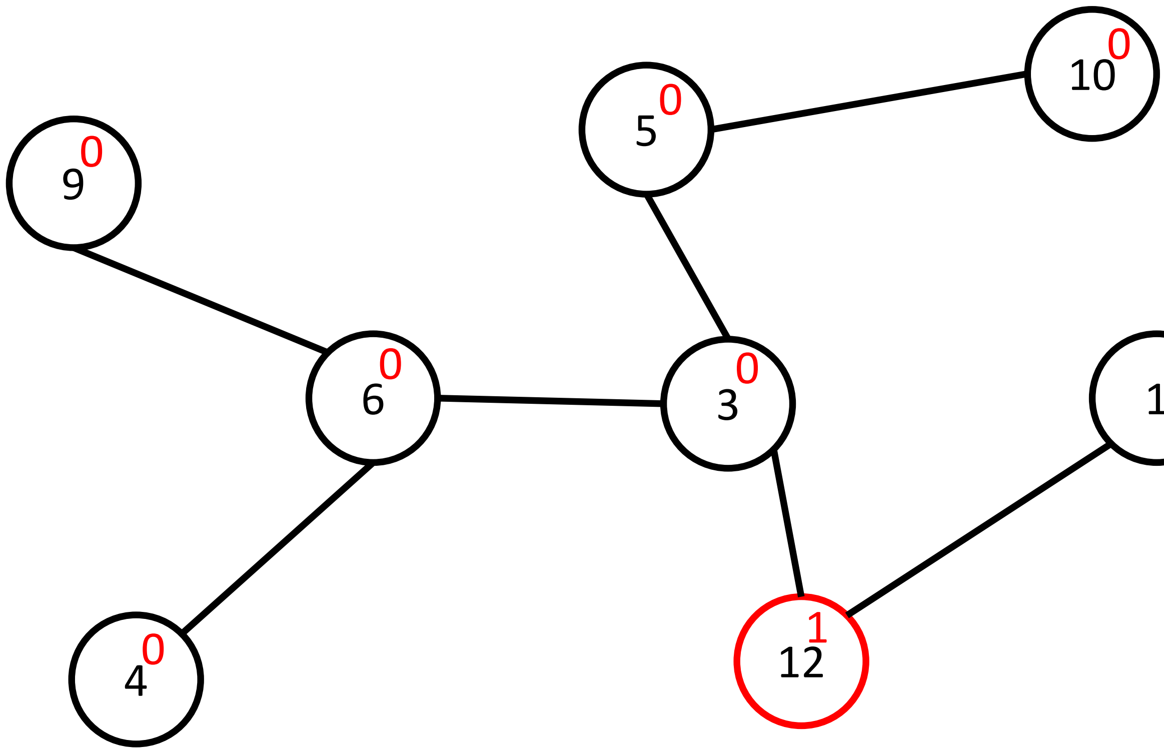


Selection

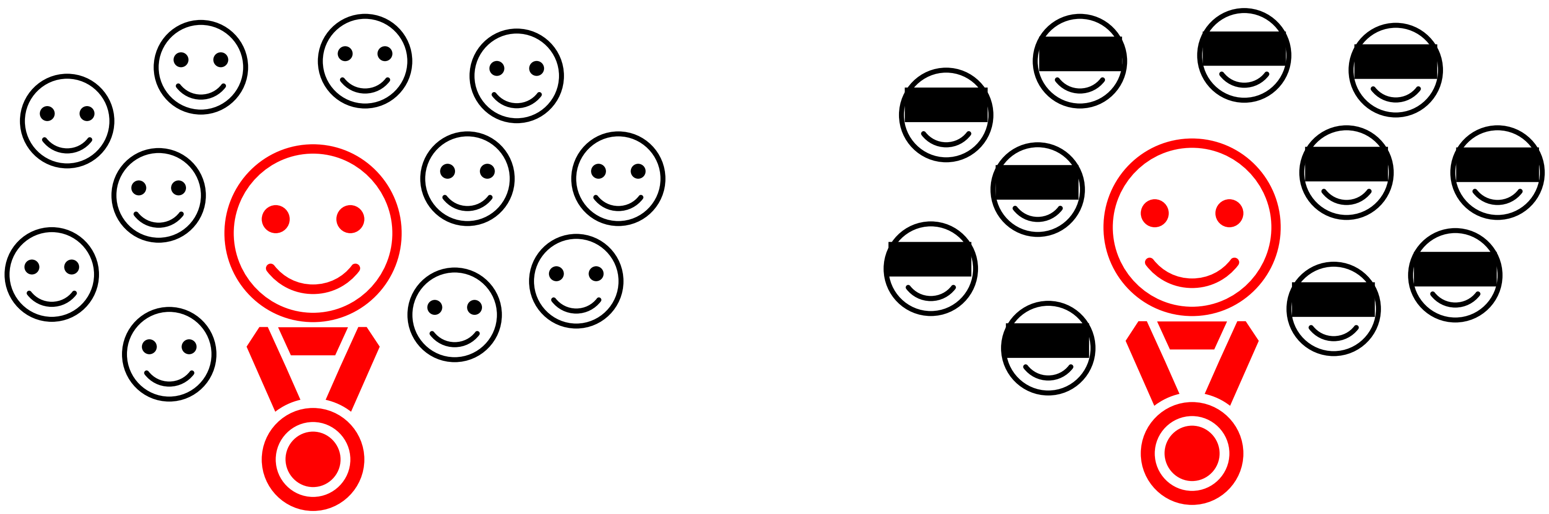


Leader outputs **1**

Every other node outputs **0**



Election vs Selection

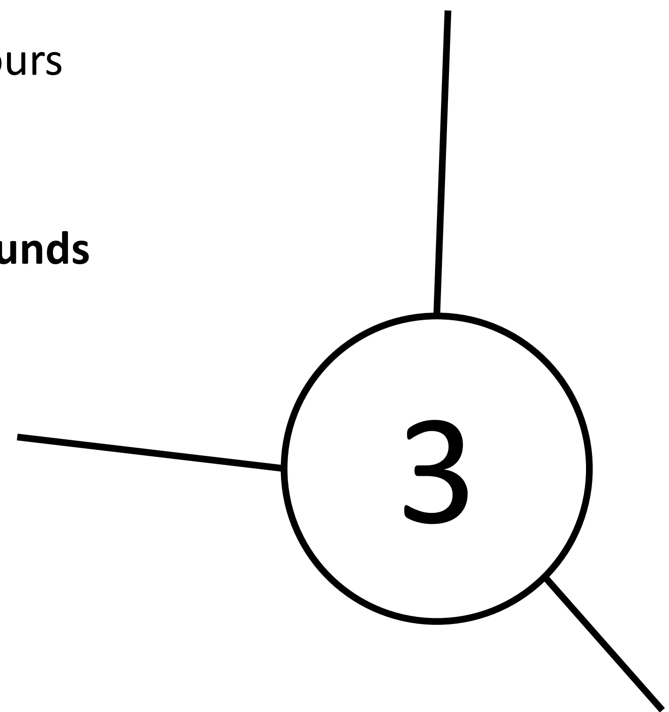


Synchronous Algorithms

In each **round**:

- **Send** messages to neighbours
- **Recei**ve messages from neighbours
- Do some **computation**

Time complexity is the **number of rounds**

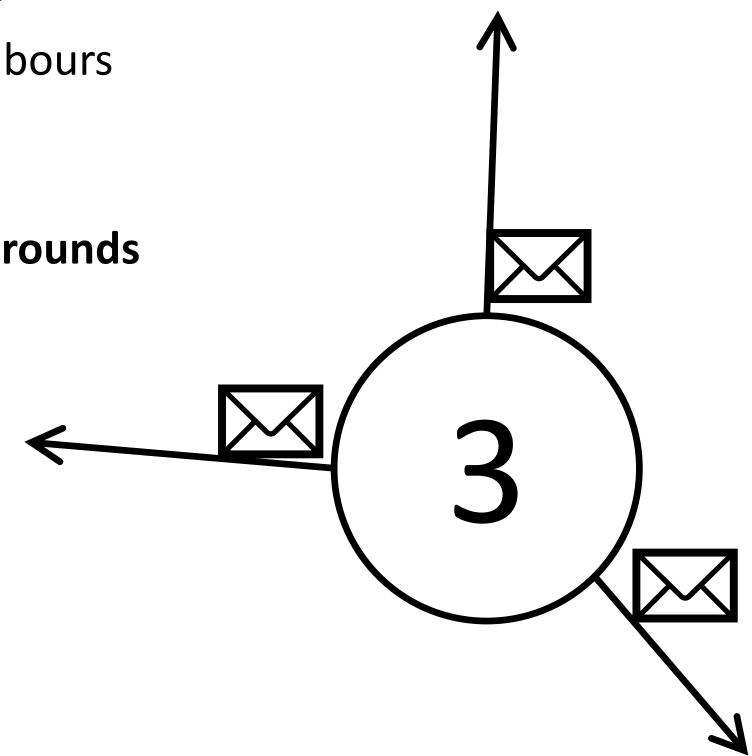


Synchronous Algorithms

In each **round**:

- **Send** messages to neighbours
- **Recei**ve messages from neighbours
- Do some **computation**

Time complexity is the **number of rounds**

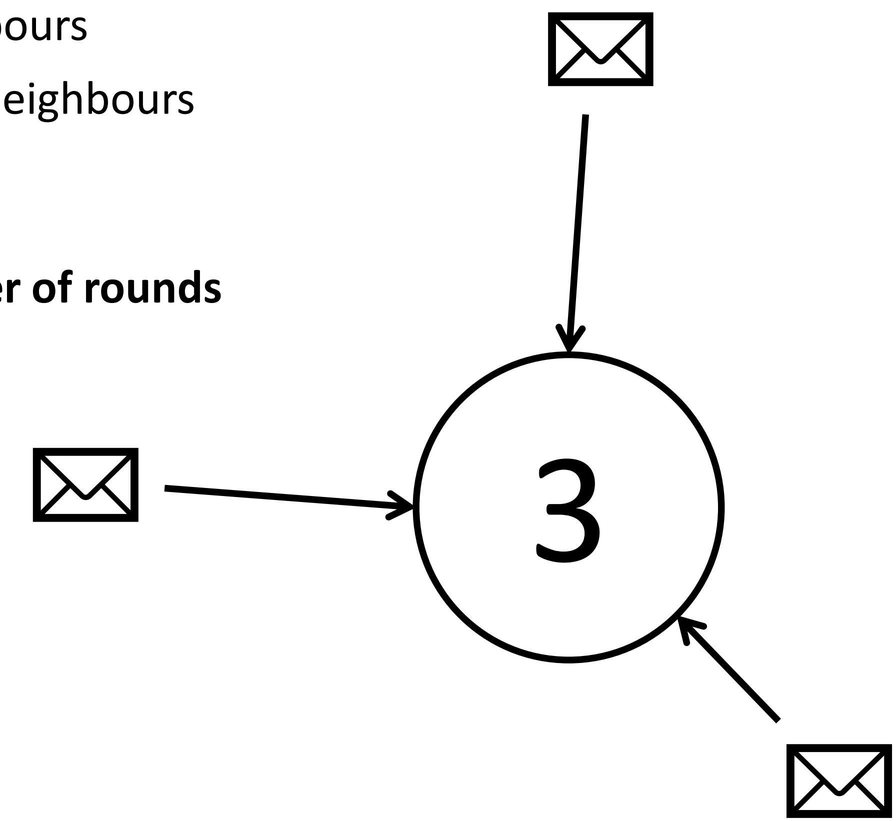


Synchronous Algorithms

In each **round**:

- **Send** messages to neighbours
- **Recei**ve messages from neighbours
- Do some **computation**

Time complexity is the **number of rounds**

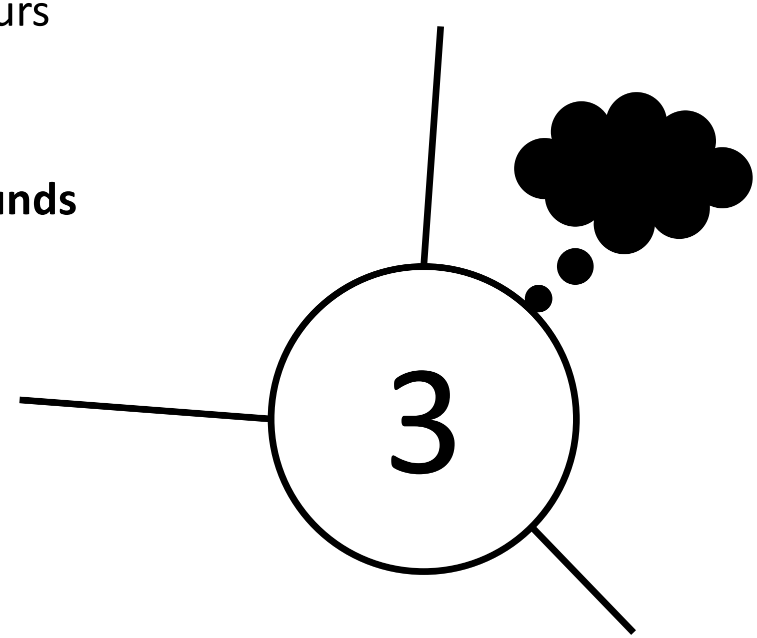


Synchronous Algorithms

In each **round**:

- **Send** messages to neighbours
- **Recei**ve messages from neighbours
- Do some **computation**

Time complexity is the **number of rounds**

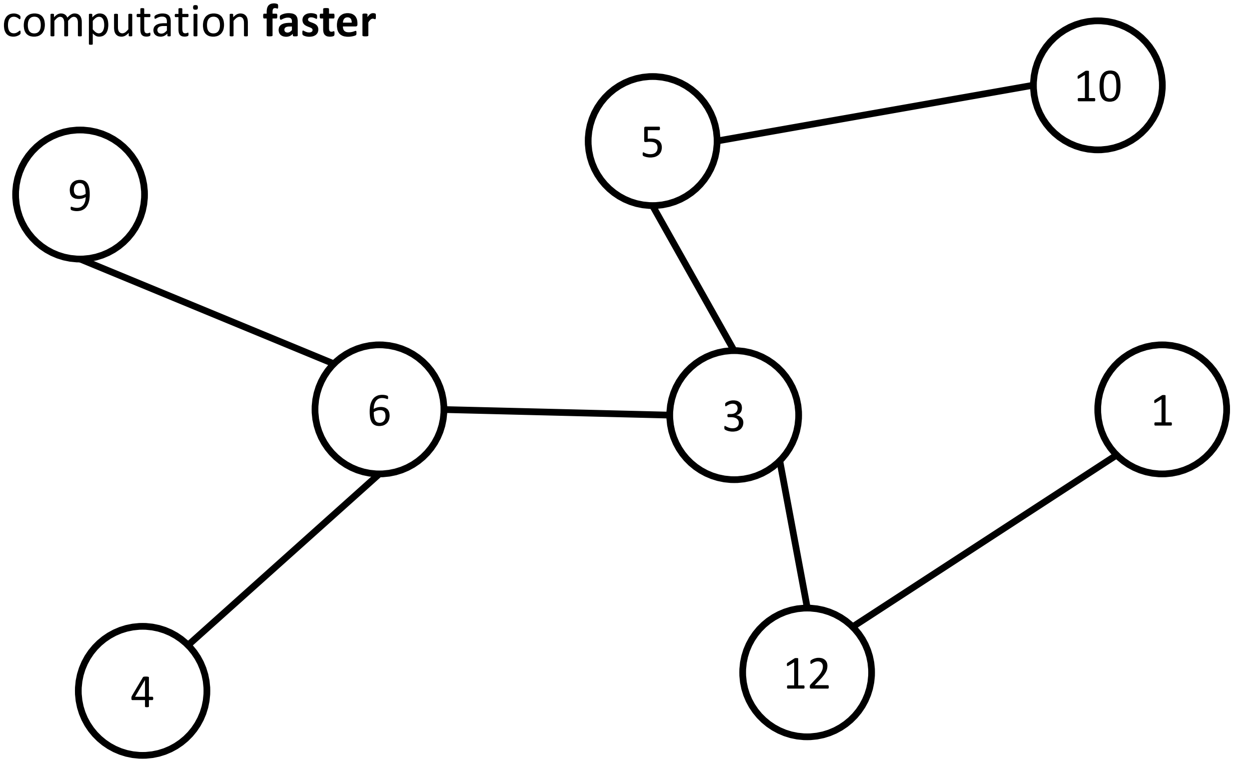


Algorithm with advice

Oracle with **full knowledge**

Gives same **advice** to each node

Goal: make computation **faster**

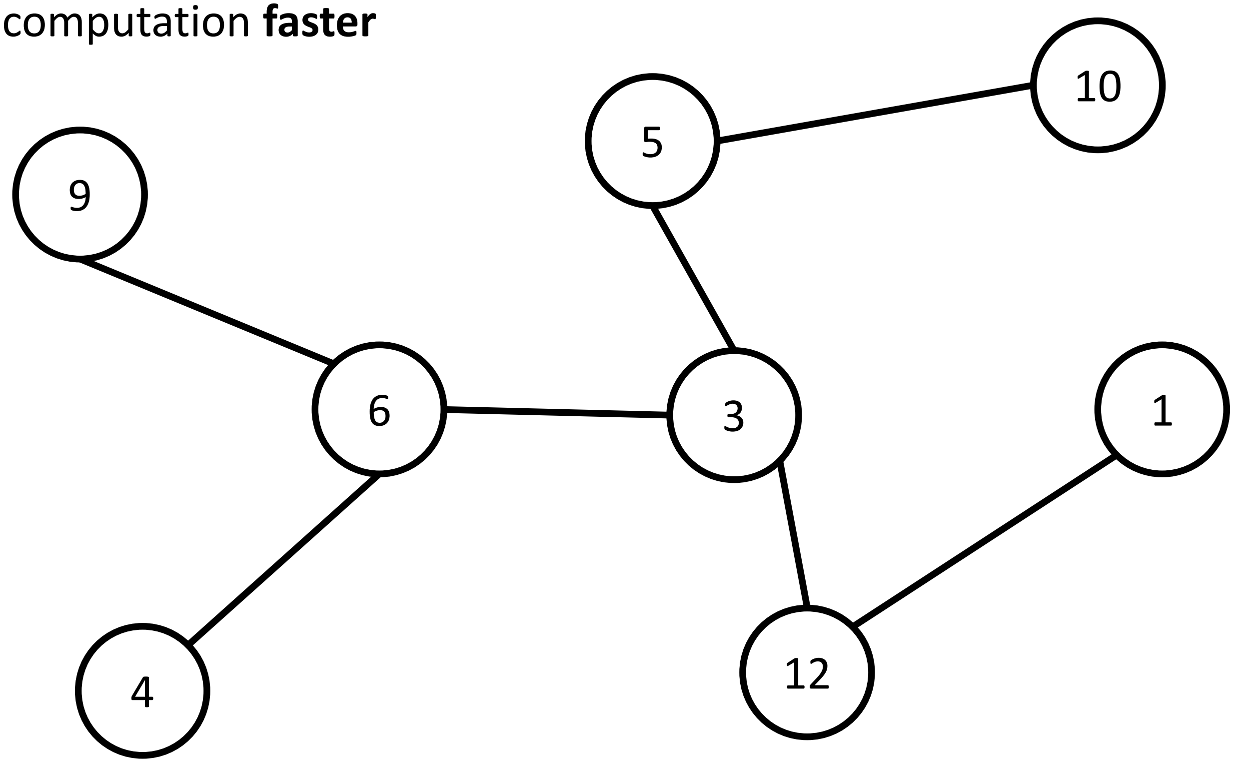


Algorithm with advice

Oracle with **full knowledge**

Gives same **advice** to each node

Goal: make computation **faster**

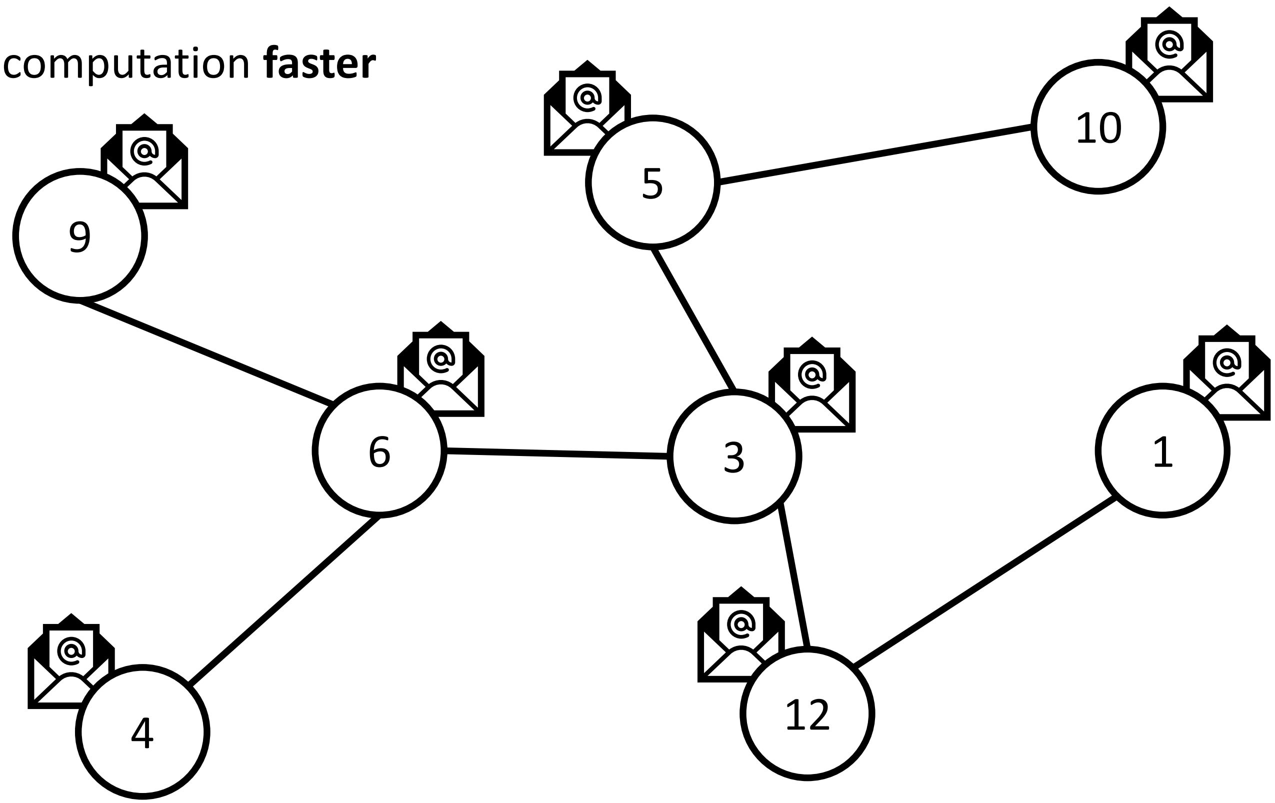


Algorithm with advice

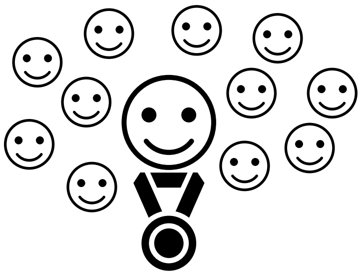
Oracle with **full knowledge**

Gives same **advice** to each node

Goal: make computation **faster**



Algorithm with advice

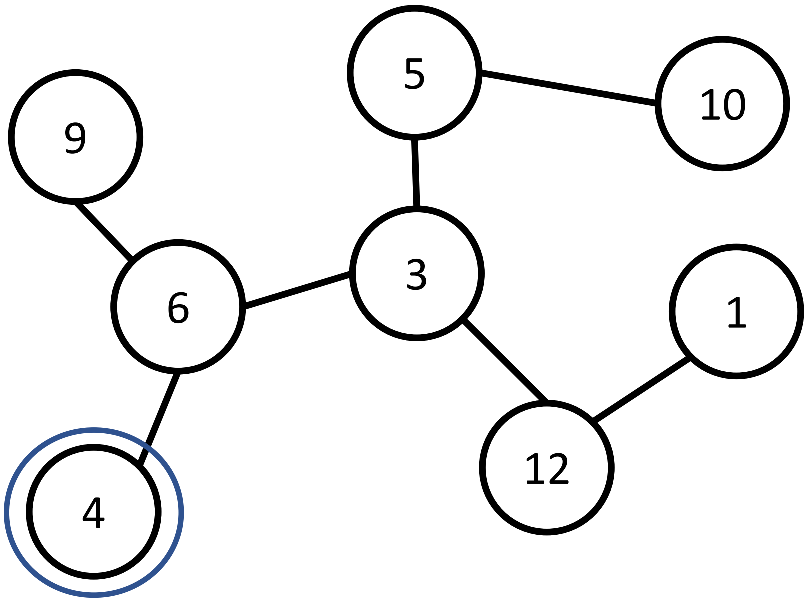


Example : Election **without** advice

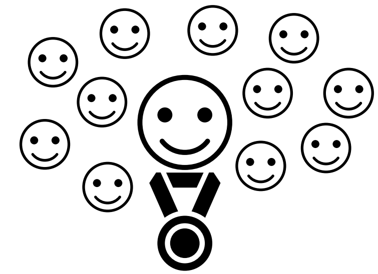
Notations:

$K(r, v) =$ knowledge of v after r rounds

$\Lambda(r, v)$ set of labels induced by $K(r, v)$



Algorithm with advice



Example : Election **without** advice

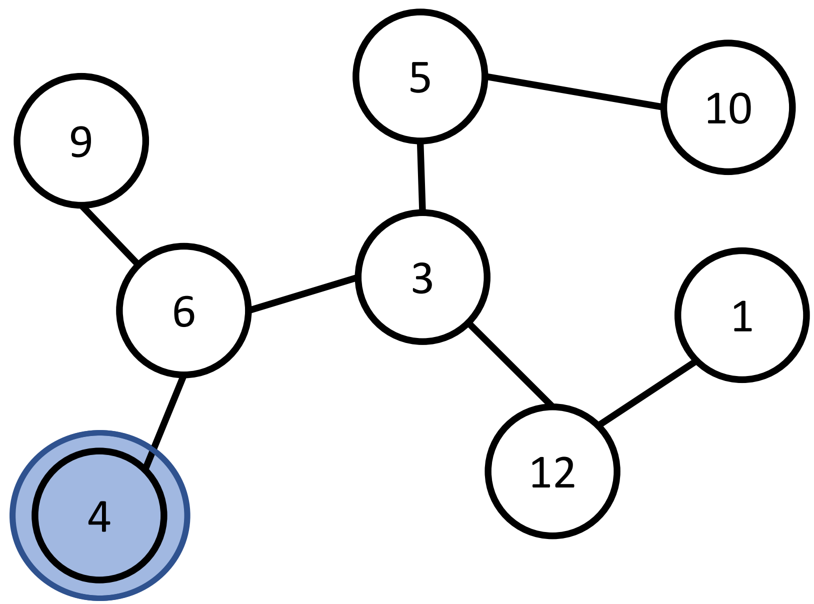
Notations:

$K(r, v)$ = knowledge of v after r rounds

$\Lambda(r, v)$ set of labels induced by $K(r, v)$

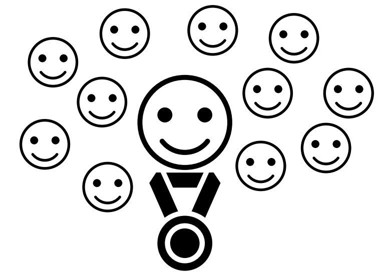
$K(0, 4)$

$\Lambda(0, 4) = \{4\}$



Round 0

Algorithm with advice



Example : Election **without** advice

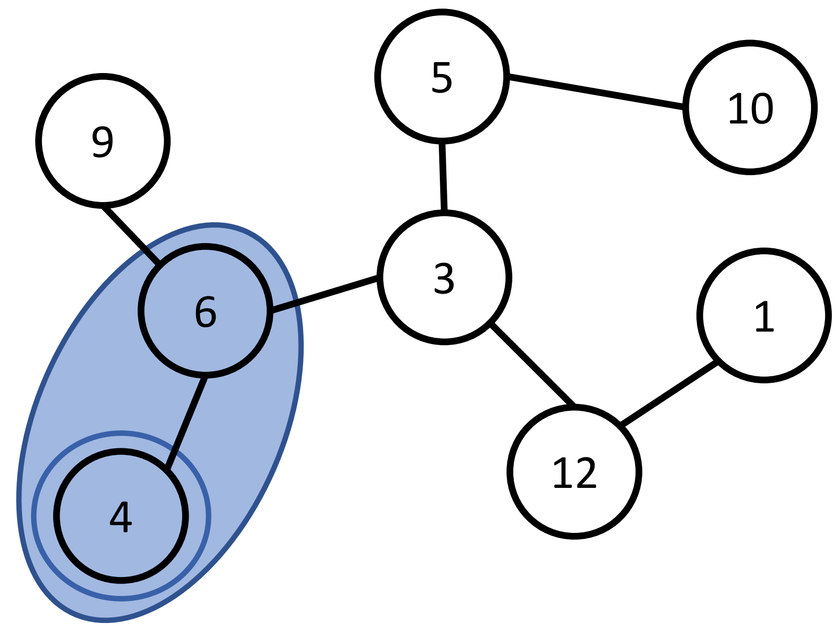
Notations:

$K(r, v)$ = knowledge of v after r rounds

$\Lambda(r, v)$ set of labels induced by $K(r, v)$

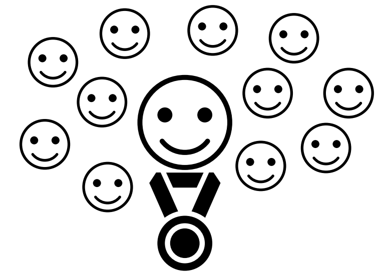
$K(1, 4)$

$\Lambda(1, 4) = \{4, 6\}$



Round 1

Algorithm with advice



Example : Election **without** advice

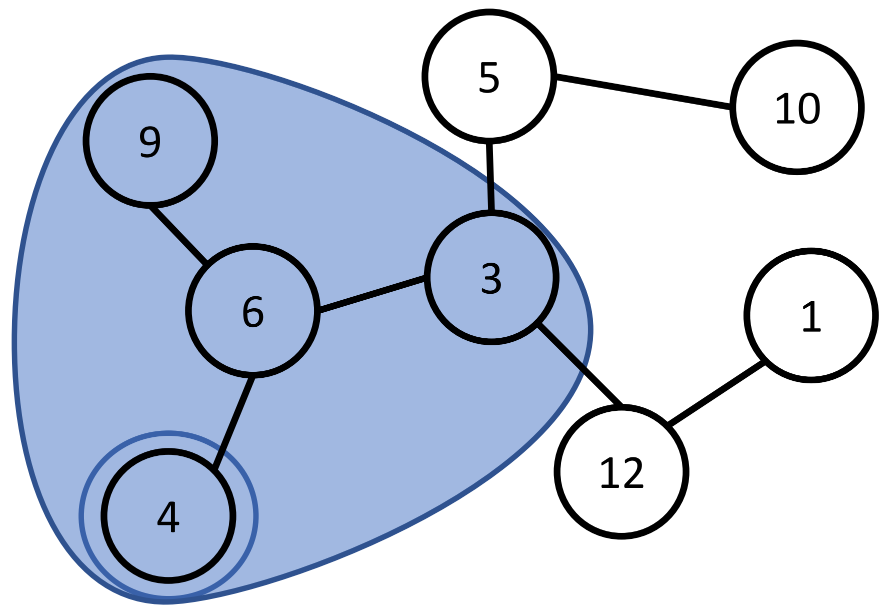
Notations:

$K(r, v)$ = knowledge of v after r rounds

$\Lambda(r, v)$ set of labels induced by $K(r, v)$

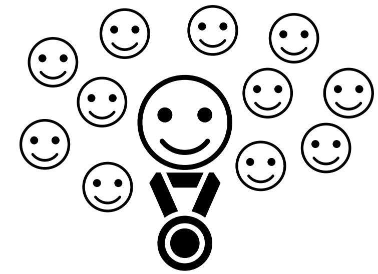
$K(2, 4)$

$\Lambda(2, 4) = \{3, 4, 6\}$



Round 2

Algorithm with advice



Example : Election **without** advice

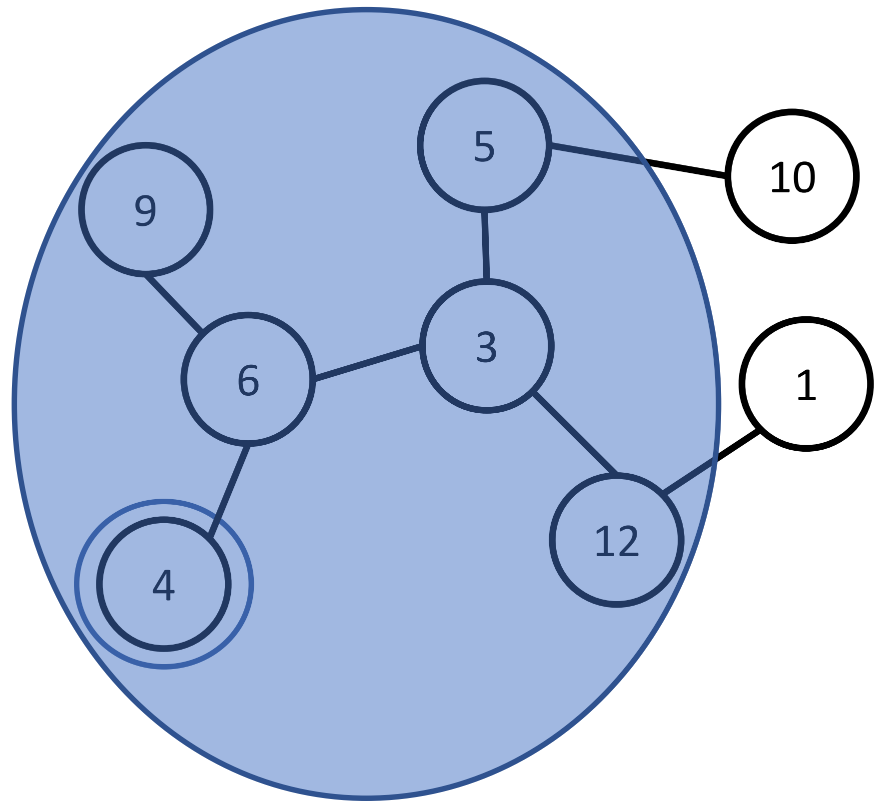
Notations:

$K(r, v)$ = knowledge of v after r rounds

$\Lambda(r, v)$ set of labels induced by $K(r, v)$

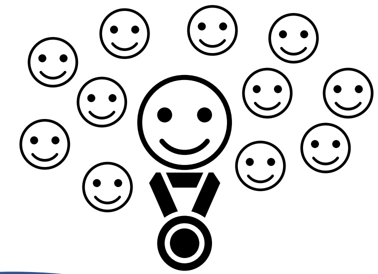
$K(3,4)$

$\Lambda(3,4) = \{3, 4, 5, 6, 12\}$



Round 3

Algorithm with advice



Example : Election **without** advice

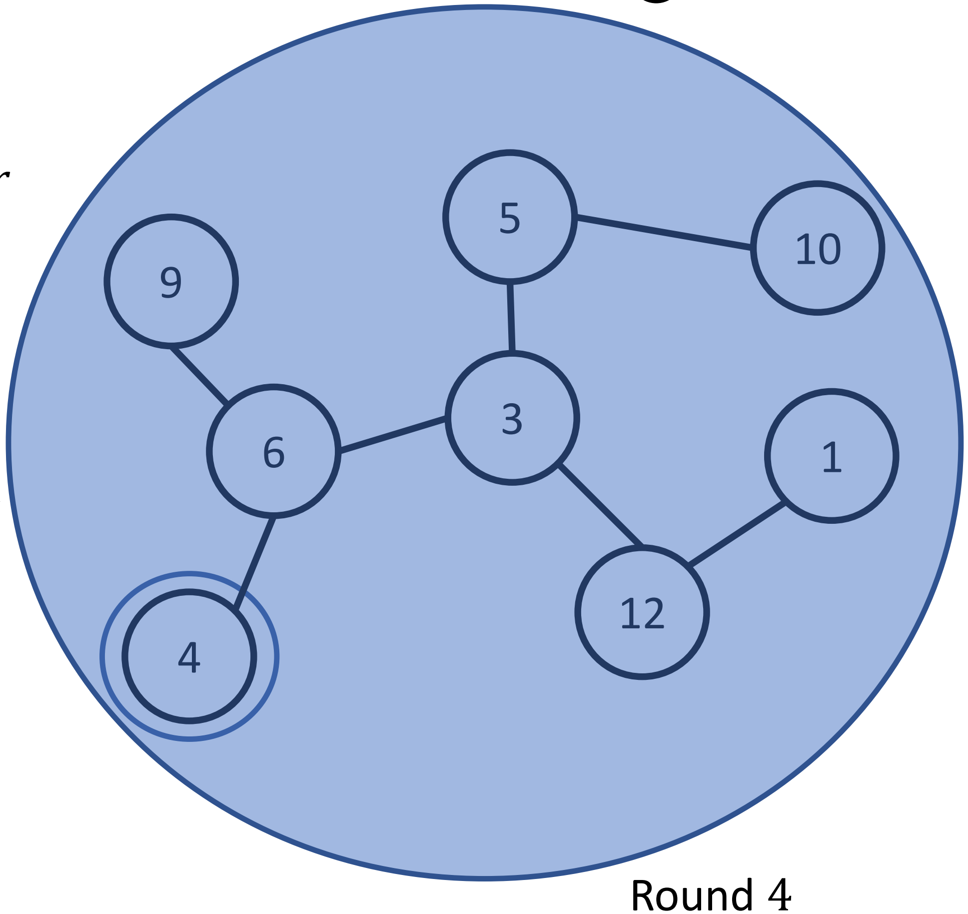
Notations:

$K(r, v)$ = knowledge of v after r rounds

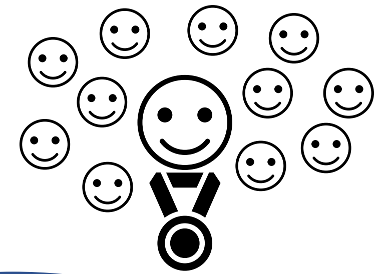
$\Lambda(r, v)$ set of labels induced by $K(r, v)$

$K(4,4)$

$\Lambda(4,4) = \{1, 3, 4, 5, 6, 10, 12\}$



Algorithm with advice

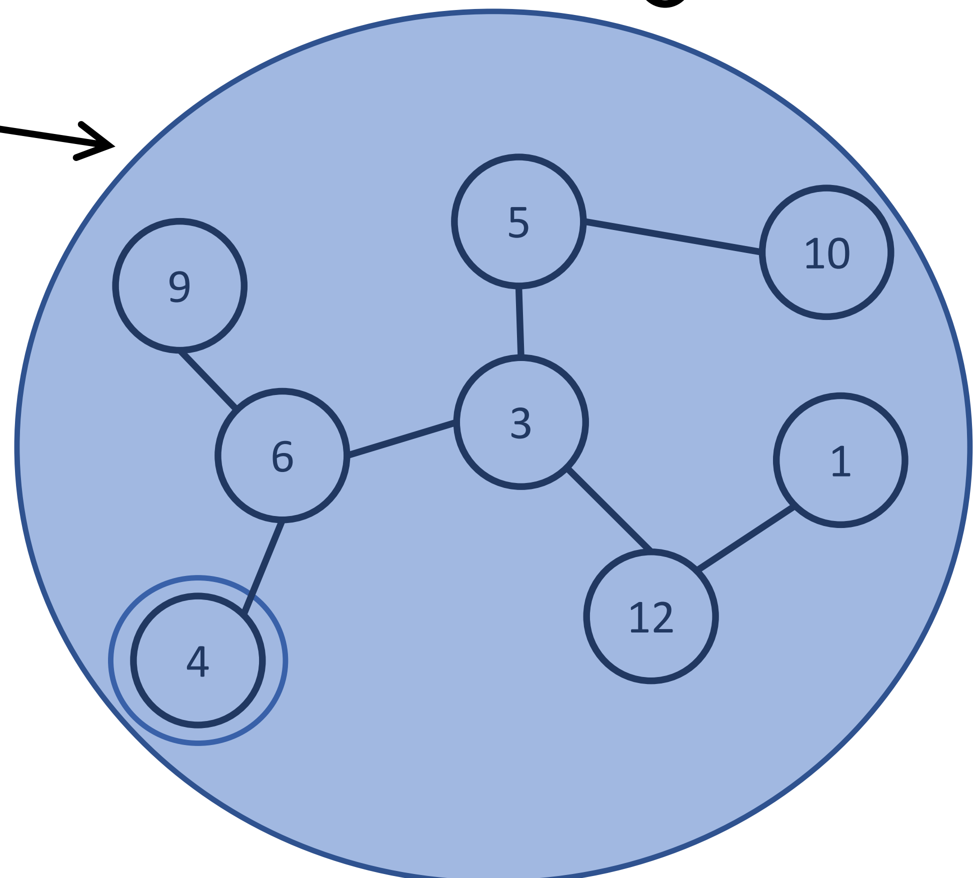


Example : Election **without** advice

$K(4,4)$

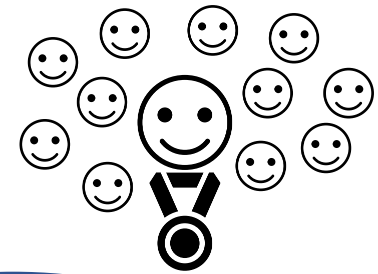
$\Lambda(4,4) = \{1,3,4,5,6,10,12\}$

Requires another round
to terminate!



Round 4

Algorithm with advice



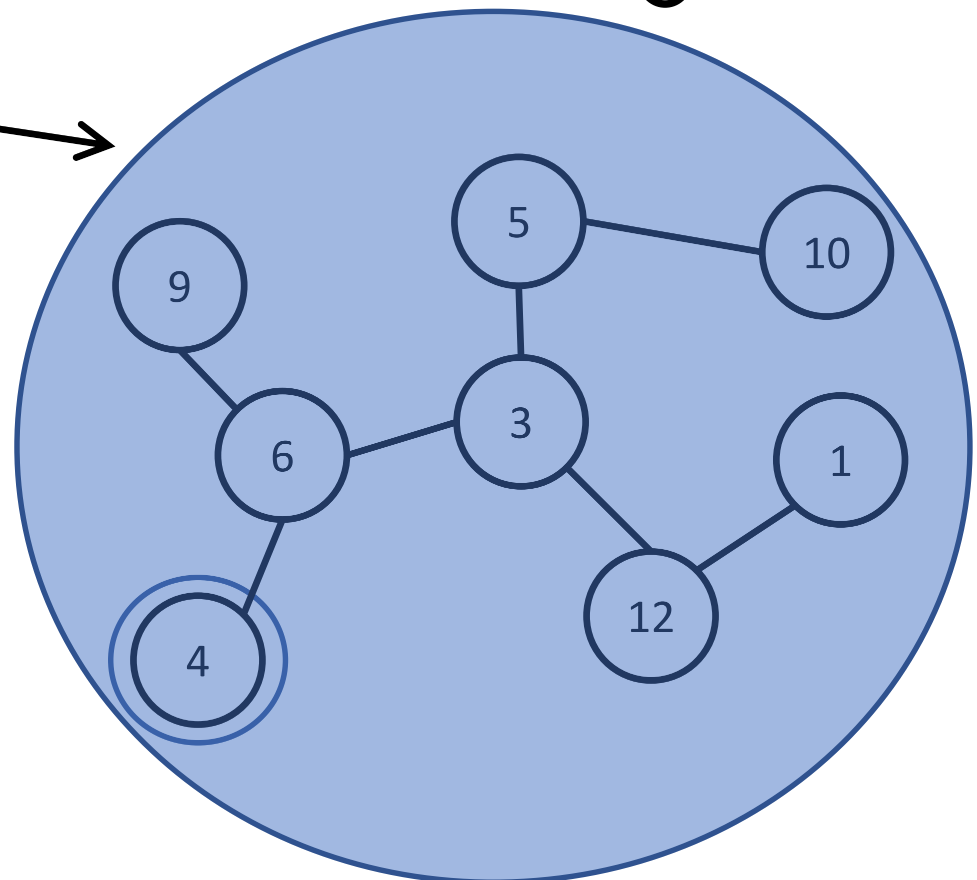
Example : Election **without** advice

$K(4,4)$

$\Lambda(4,4) = \{1,3,4,5,6,10,12\}$

Requires another round
to terminate!

What if we give some **advice**?



Round 4

Task – Measure of Difficulty

Time constraint for the execution

How much **advice** needed ?

Upper and **lower** bound the size of advice

Tight Bounds on Advice

Tight bounds are given on the size of advice.

Tight Bounds on Advice

Tight bounds are given on the size of advice.

$$\Theta\big(f(x)\big) \;\Leftrightarrow\; \Omega\big(f(x)\big) \wedge O\big(f(x)\big)$$

Tight Bounds on Advice

Tight bounds are given on the size of advice.

$$\Theta\big(f(x)\big) \iff \Omega\big(f(x)\big) \wedge \mathrm{O}\big(f(x)\big)$$

Lower bound l

Find a class of graphs for which a least l advice needed for any algorithm

Upper bound u

Find an algorithm for which at most u advice needed on all graphs

How is it helpful?

Can **rule out** entire classes of algorithms

How is it helpful?

Can **rule out** entire classes of algorithms

Given result: Task T needs $\Theta(\log n)$ bits of advice

How is it helpful?

Can **rule out** entire classes of algorithms

Given result: Task T needs $\Theta(\log n)$ bits of advice

Proposed algorithm: Needs linear upper bound on n as advice.

How is it helpful?

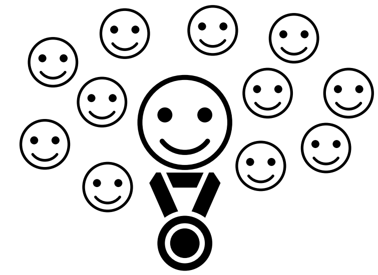
Can **rule out** entire classes of algorithms

Given result: Task T needs $\Theta(\log n)$ bits of advice

Proposed algorithm: Needs linear upper bound on n as advice.

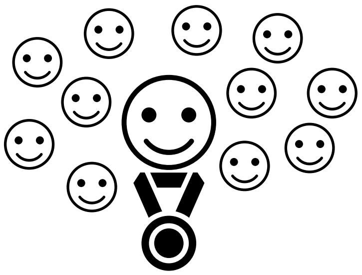
Contradiction: Advice can be given by $\lceil \log n \rceil$, using $\Theta(\log \log n)$ bits.

Results - Election



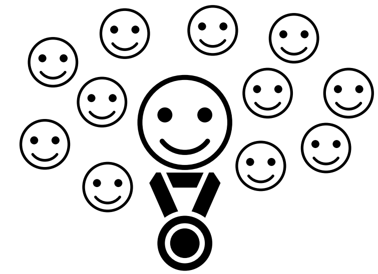
Time	Advice
$> \ diam$	0
$diam$	$\Theta(\log diam)$
$< \ diam$	$\Theta(\log n)$

Results - Election



Time	Advice
$> \ diam$	0
$diam$	$\Theta(\log diam)$
$< \ diam$	$\Theta(\log n)$

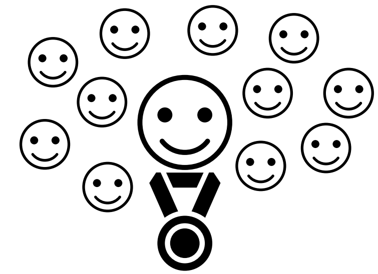
Results - Election



Time	Advice
$> \ diam$	0
$diam$	$\Theta(\log diam)$
$< \ diam$	$\Theta(\log n)$

Provide the **diameter** of the graph

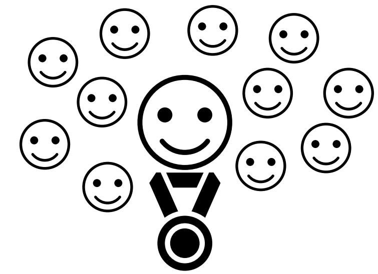
Results - Election



Time	Advice
$> \ diam$	0
$diam$	$\Theta(\log diam)$
$< \ diam$	$\Theta(\log n)$

No better advice than to **give** the **solution**

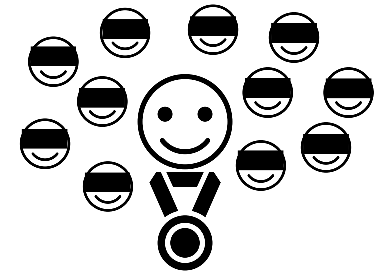
Results - Election



Time	Advice
$> \ diam$	0
$diam$	$\Theta(\log diam)$
$< \ diam$	$\Theta(\log n)$

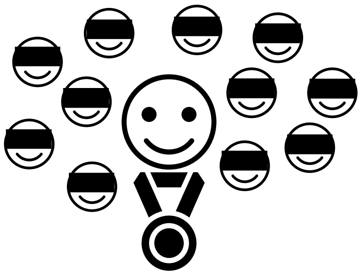
Intra-task jumps

Results - Selection



Time	Advice
$> \; diam$	0
$a \cdot diam,$ $a \in (0, 1)$	$\Theta(\log \log diam)$
$diam^e,$ $e < 1$	$\Theta(\log diam)$

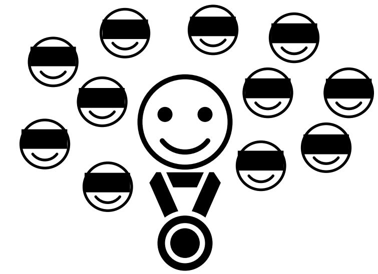
Results - Selection



Time	Advice
$> \ diam$	0
$a \cdot diam,$ $a \in (0,1)$	$\Theta(\log\log diam)$
$diam^e,$ $e < 1$	$\Theta(\log diam)$

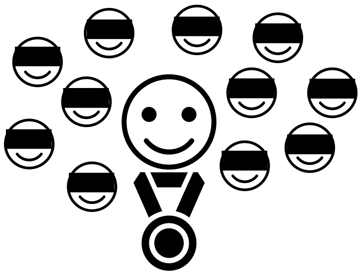
Only valid for rings

Results - Selection



Time	Advice
$> \ diam$	0
$a \cdot diam,$ $a \in (0,1)$	$\Theta(\log\log diam)$
$diam^e,$ $e < 1$	$\Theta(\log diam)$

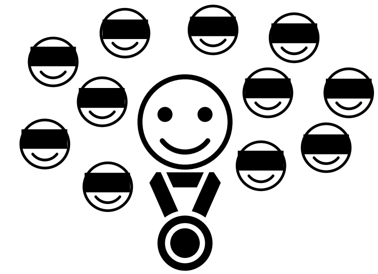
Results - Selection



Time	Advice
$> \ diam$	0
$a \cdot diam$, $a \in (0,1)$	$\Theta(\log\log diam)$
$diam^e$, $e < 1$	$\Theta(\log diam)$

We will go through the algorithm for the upper bound

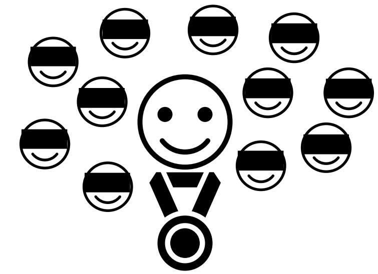
Results - Selection



Time	Advice
$> \ diam$	0
$a \cdot diam,$ $a \in (0,1)$	$\Theta(\log\log diam)$
$diam^e,$ $e < 1$	$\Theta(\log diam)$

For rings $\Theta(\log diam) =$
$\Theta(\log n)$

Results - Selection



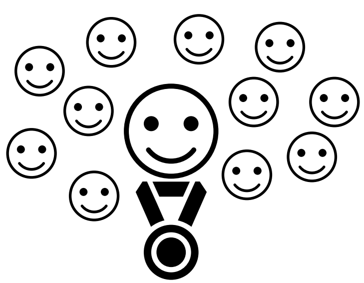
Time	Advice
$> \ diam$	0
$a \cdot diam$, $a \in (0,1)$	$\Theta(\log\log diam)$
$diam^e$, $e < 1$	$\Theta(\log diam)$

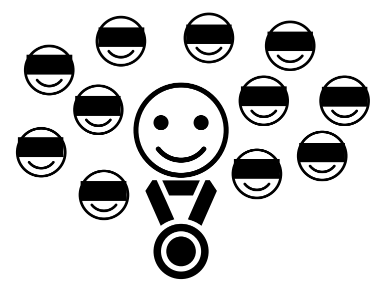
Intra-task jumps

Results

Time	Advice
$> \ diam$	0
$diam$	$\Theta(\log diam)$
$< \ diam$	$\Theta(\log n)$

Time	Advice
$> \ diam$	0
$a \cdot diam,$ $a \in (0,1)$	$\Theta(\log \log diam)$
$diam^e,$ $e < 1$	$\Theta(\log diam)$



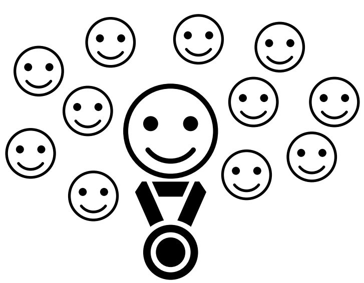


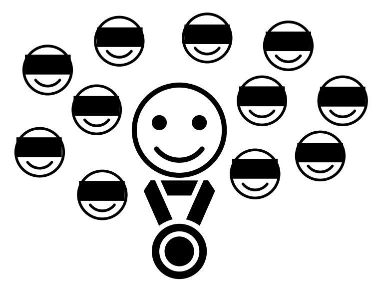
Results

Inter-task jump

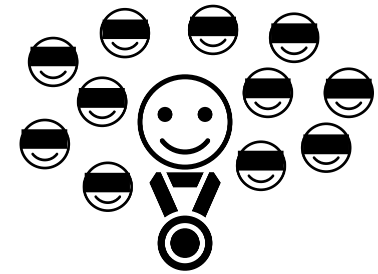
Time	Advice
$> \ diam$	0
$diam$	$\Theta(\log diam)$
$< \ diam$	$\Theta(\log n)$

Time	Advice
$> \ diam$	0
$a \cdot diam$, $a \in (0, 1)$	$\Theta(\log \log diam)$
$diam^e$, $e < 1$	$\Theta(\log diam)$





Selection - Example

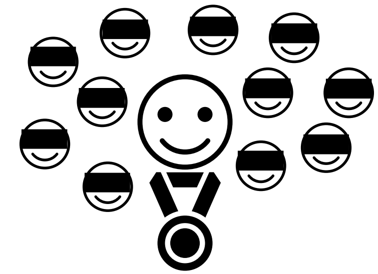


Time constraint: t $= \ a \cdot diam\ , a \in (0,1)$

Goal : Prove that size of advice is $\mathrm{O}(\log\log diam(R))$ for any ring R

Simplification: $a = 1$, so $\ t = diam$

Selection - Example



Algorithm consists of two stages

1. Round-by-round discovery
2. Eliminate resulting nodes from first stage

Advice string split in two parts $A = A_1 A_2$

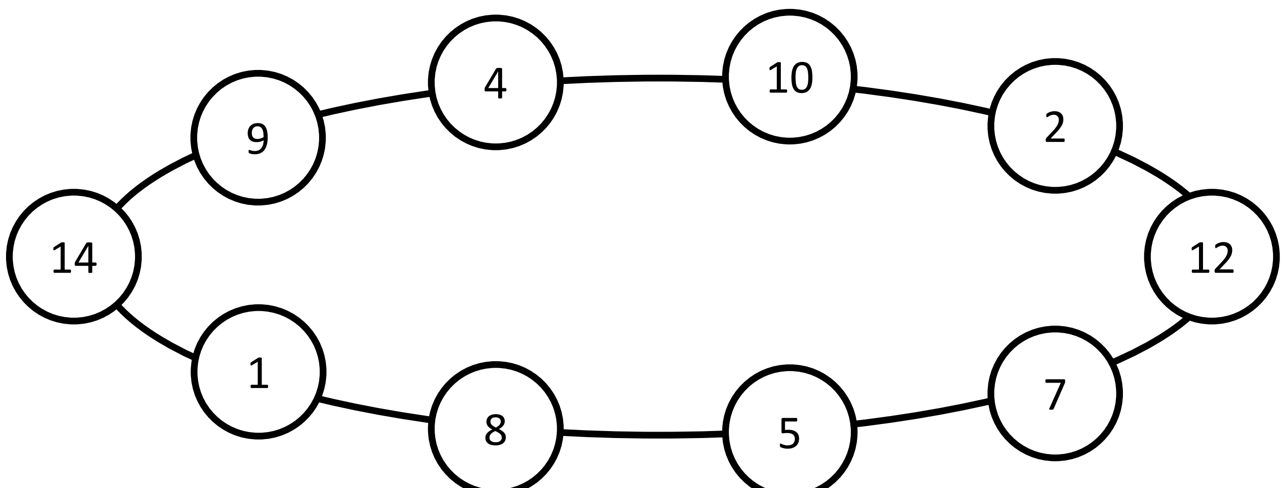
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$



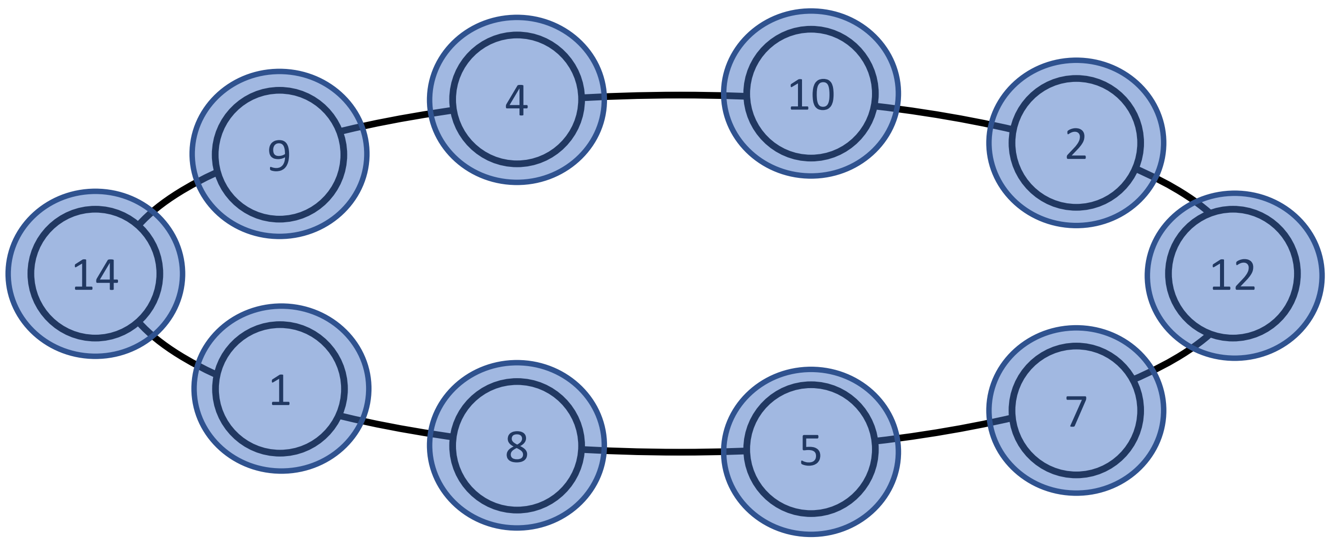
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$



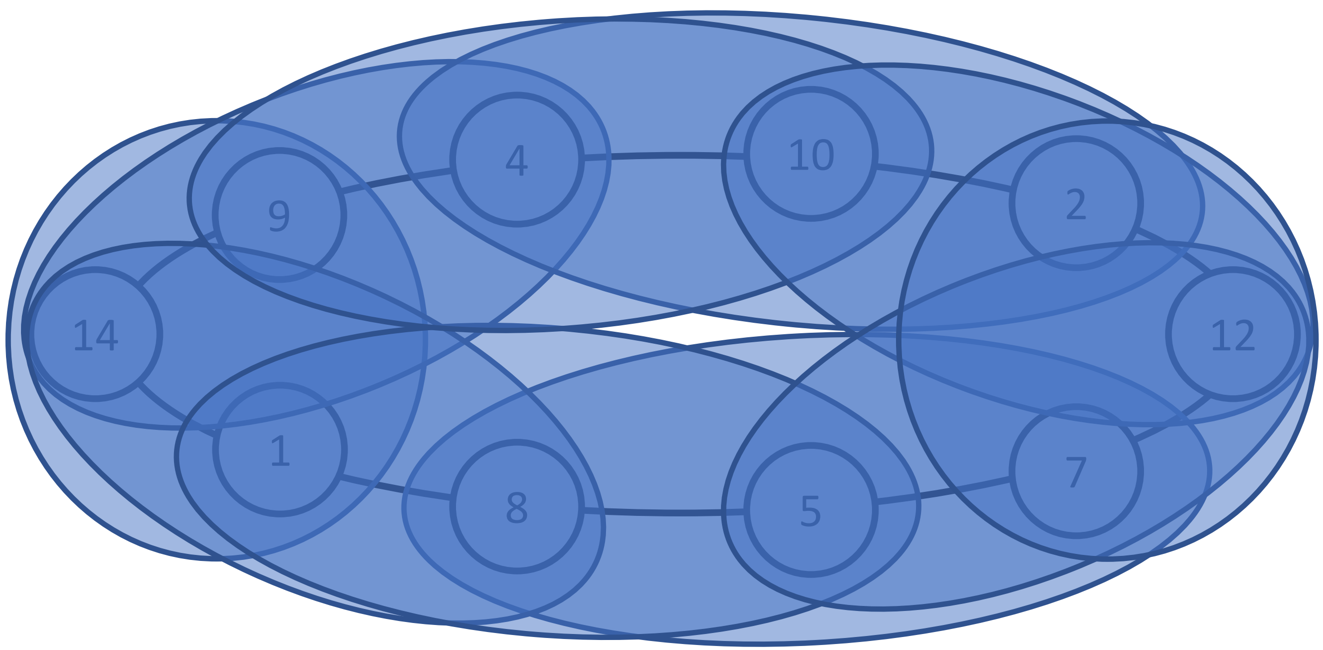
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$



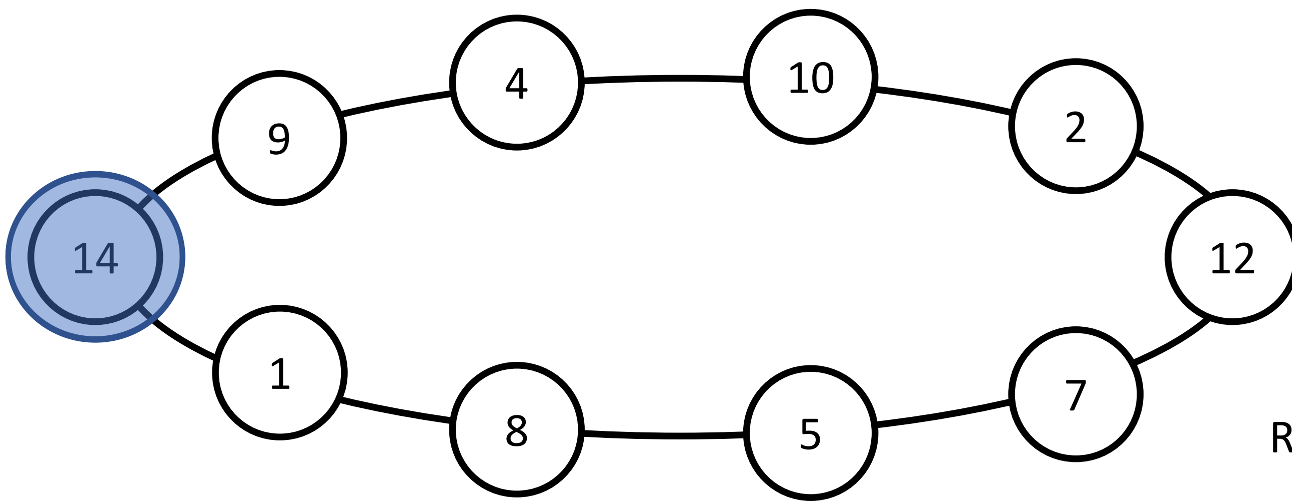
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $r = 4$



Round 0

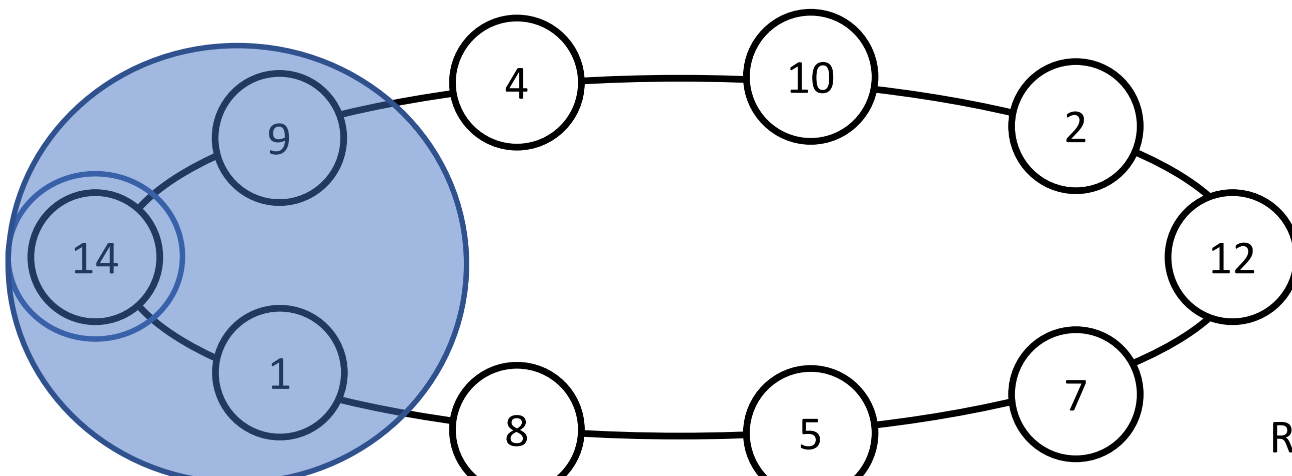
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\qquad$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $\qquad$ $r = 4$



Round 1

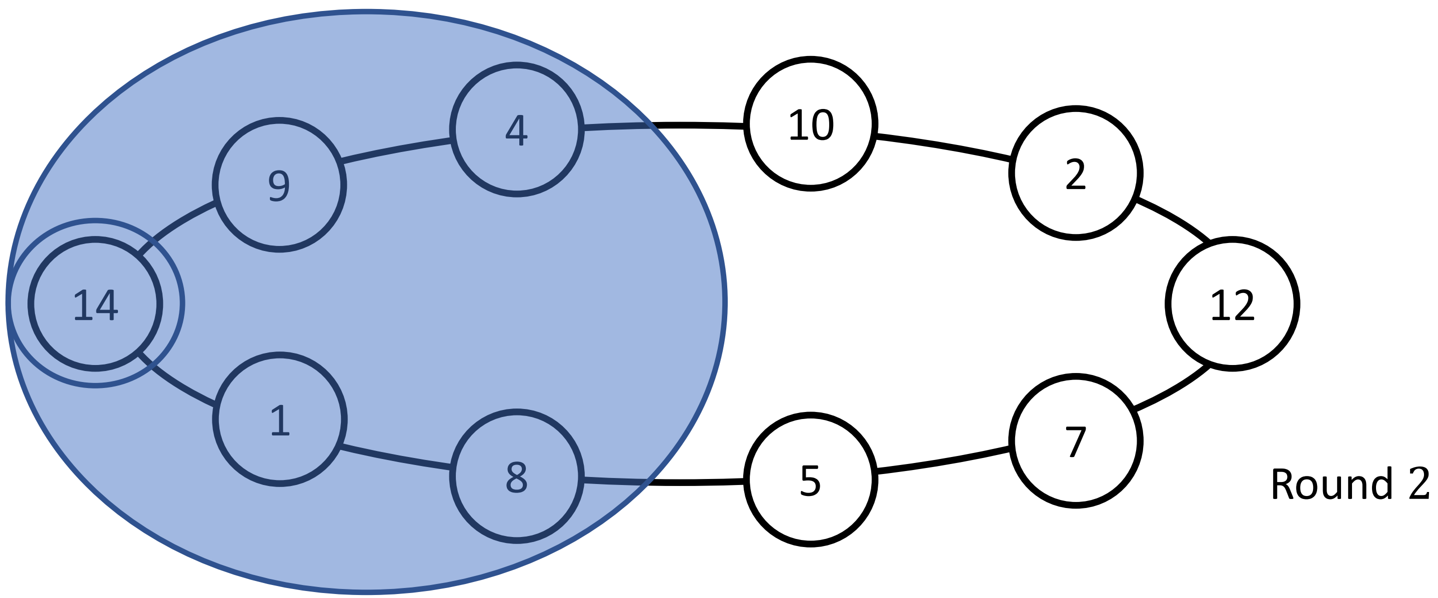
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $r = 4$



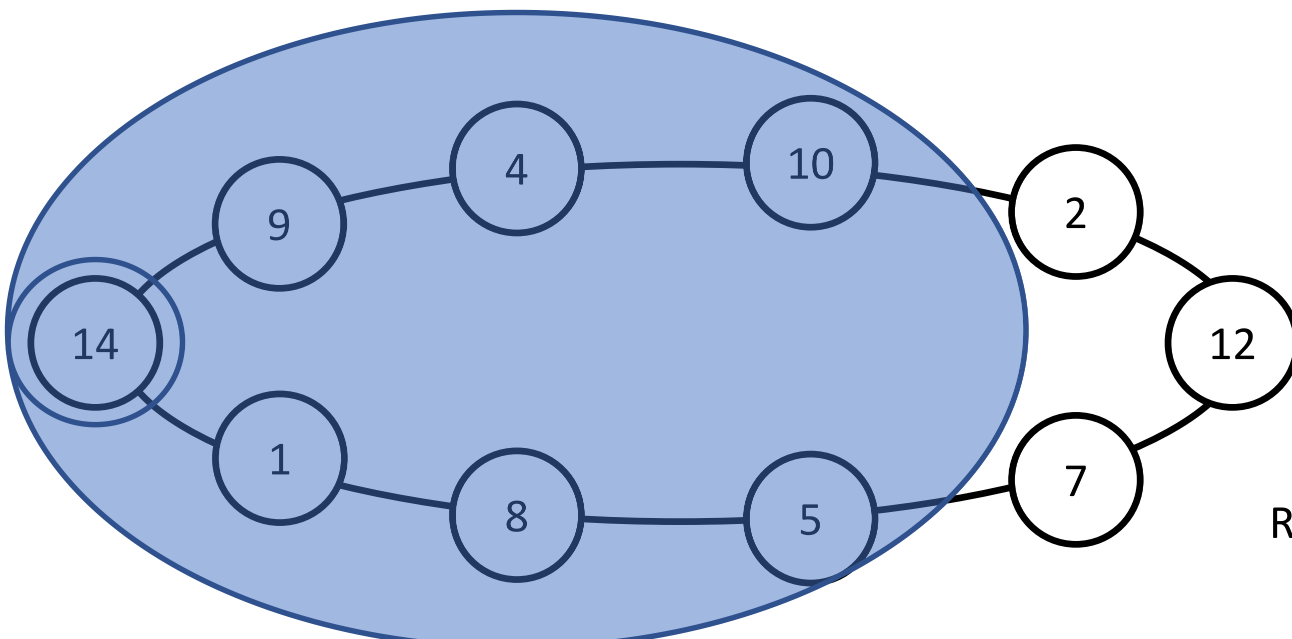
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $r = 4$



Round 3

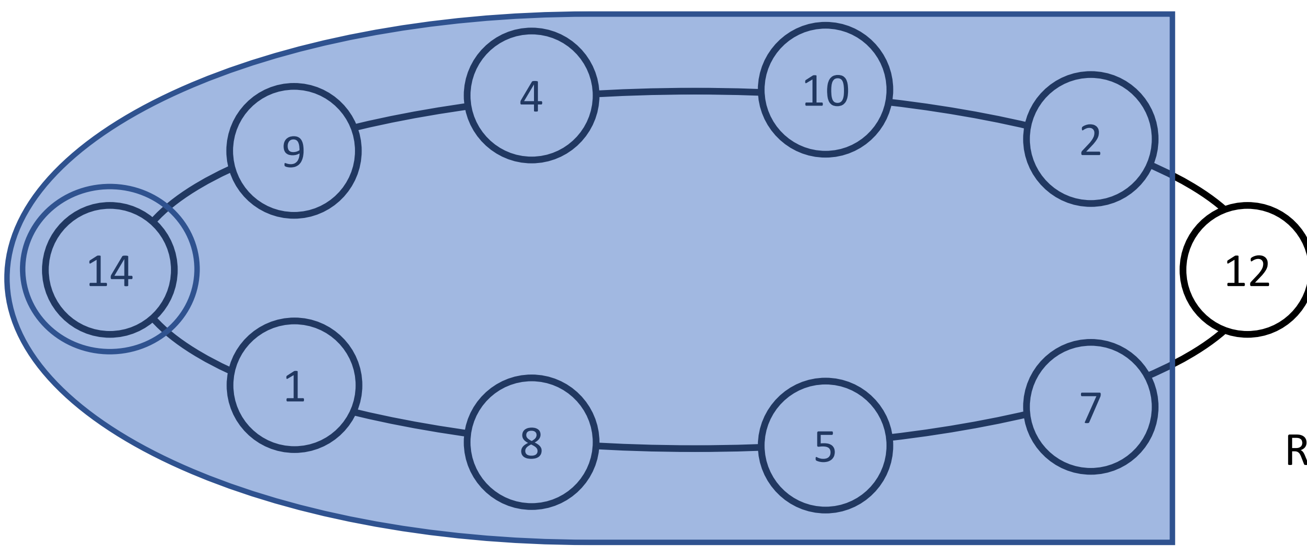
Selection – Example

Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $r = 4$



Round 4

Selection – Example

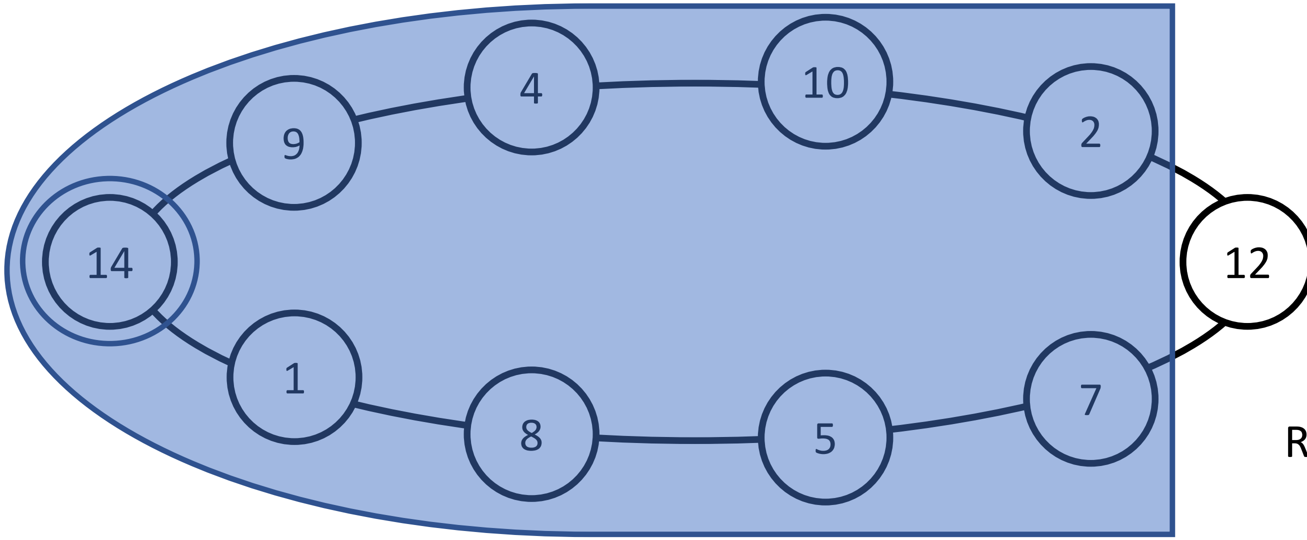
Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $r = 4$

If $v \neq \max(\Lambda(r, v))$ output 0



Round 4

Selection – Example

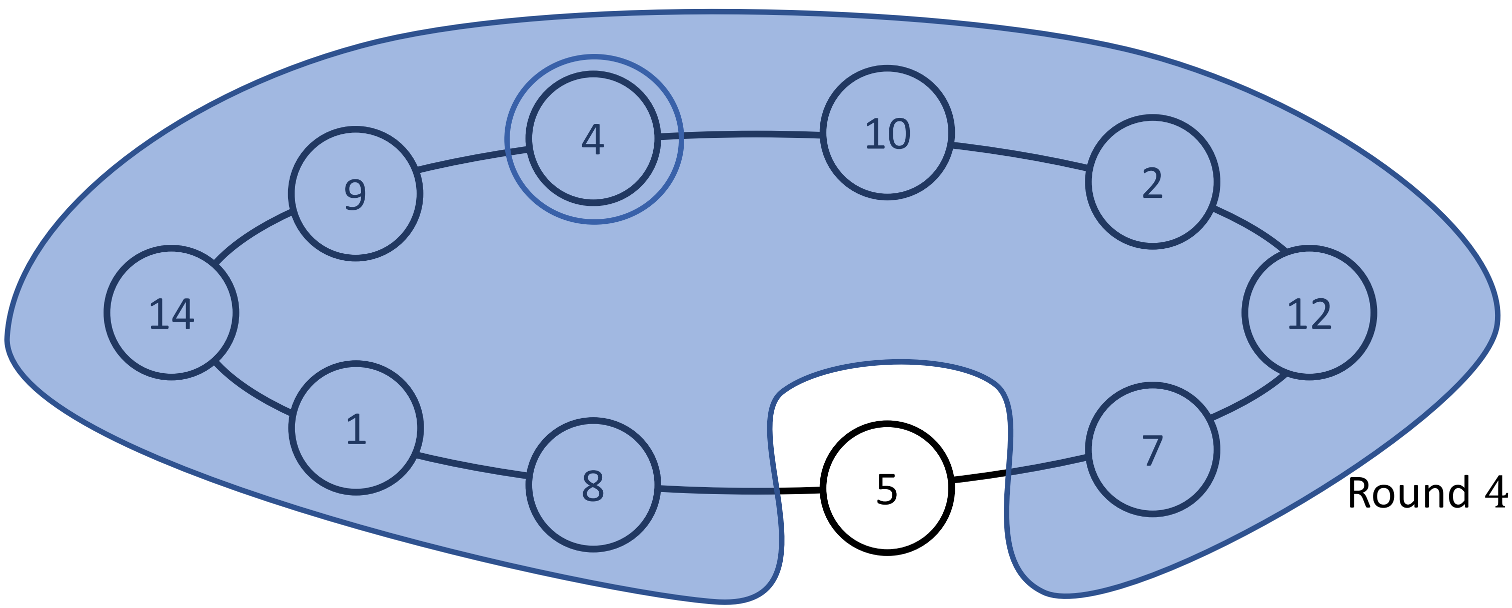
Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $r = 4$

If $v \neq \max(\Lambda(r, v))$ output 0



Selection – Example

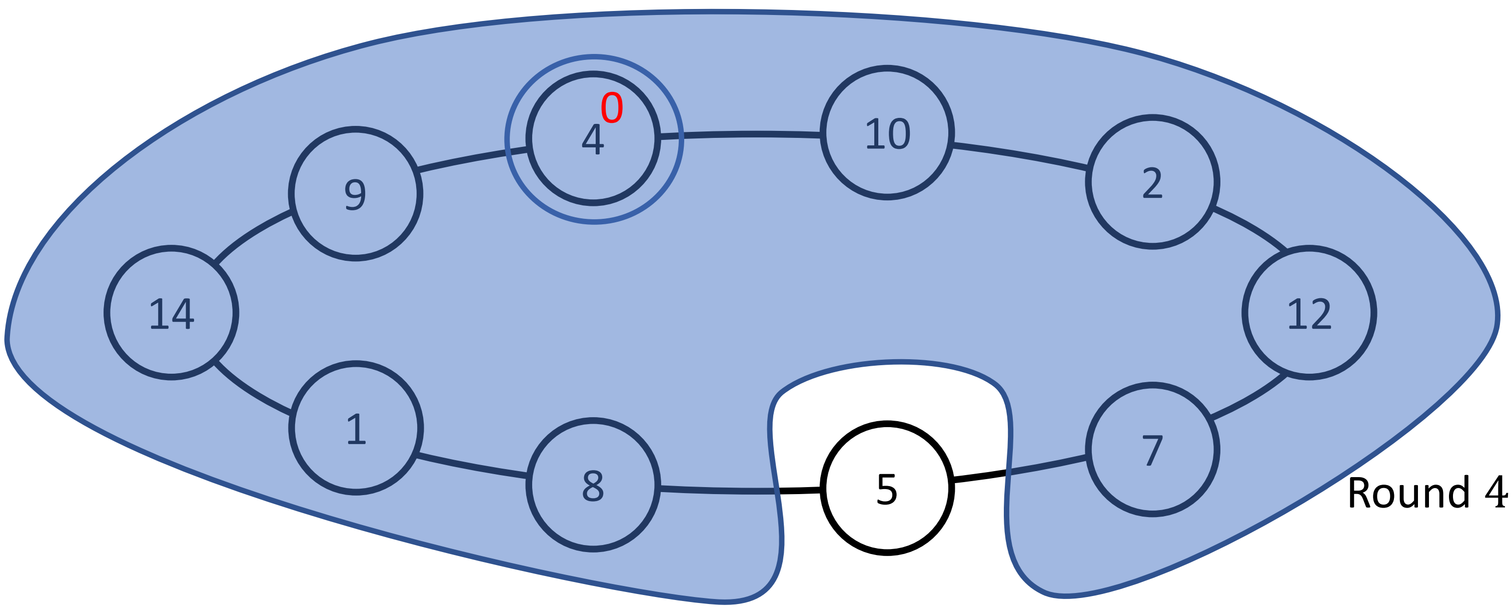
Algorithm - Stage 1

$A_1 = \lfloor \log(diam(R)) \rfloor$ $\lfloor \log(5) \rfloor = 2$

On each node v

Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$ $r = 4$

If $v \neq \max(\Lambda(r, v))$ output 0



Selection – Example

Algorithm - Stage 2: Advice construction

$C_R = \{\gamma_0, \gamma_1, \ldots, \gamma_{|C_R|-1}\}$ = set of resulting nodes where γ_0 is largest

Goal: eliminate all but γ_0

Selection – Example

Algorithm - Stage 2: Advice construction

$C_R = \{\gamma_0, \gamma_1, \ldots, \gamma_{|C_R|-1}\}$ = set of resulting nodes where γ_0 is largest

Goal: eliminate all but γ_0

Solution: for each $\gamma_{j,j>0}$, find difference with γ_0 and provide it as advice

Selection – Example

Algorithm - Stage 2: Advice construction

$C_R = \{\gamma_0, \gamma_1, \dots, \gamma_{|C_R|-1}\}$ = resulting set of nodes where γ_0 is largest

Goal: eliminate all but γ_0

Solution: for each $\gamma_{j,j>0}$, find difference with γ_0 and provide it as advice

$$\gamma_0 = 100111$$

$$\gamma_1 = 100011$$

Selection – Example

Algorithm - Stage 2: Advice construction

$C_R = \{\gamma_0, \gamma_1, \ldots, \gamma_{|C_R|-1}\}$ = resulting set of nodes where γ_0 is largest

Goal: eliminate all but γ_0

Solution: for each $\gamma_{j, j>0}$, find difference with γ_0 and provide it as advice

$$\gamma_0 = 100111$$

$$\gamma_2 = 100101$$

Selection – Example

Algorithm - Stage 2: Advice construction

A_2 set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \emptyset$

Selection – Example

Algorithm - Stage 2: Advice construction

A_2 set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \emptyset$

$$\gamma_0 = 100111$$

$$\gamma_1 = 100011$$

Selection – Example

Algorithm - Stage 2: Advice construction

A_2 set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \{3\}$

$$\gamma_0 = 100111$$

$$\gamma_1 = 100011$$

Selection – Example

Algorithm - Stage 2: Advice construction

A_2 set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \{3\}$

$$\gamma_0 = 100111$$

$$\gamma_2 = 100101$$

Selection – Example

Algorithm - Stage 2: Advice construction

A_2 set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \{3, 4\}$

$$\gamma_0 = 100111$$

$$\gamma_2 = 100101$$

Selection – Example

Algorithm - Stage 2: Advice construction

A_2 set of indices satisfying:

For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

Construction Example

$A_2 = \{3, 4\}$

$$\gamma_0 = 100111$$

$$\gamma_3 = 100001$$

Selection – Example

Algorithm - Stage 2: Algorithm

On each node γ in $\mathrm{C_R}$

If there exists $\mathrm{i} \in A_2$ such that $\gamma[i] = 0$

Output 0

Else

Output 1

Selection – Example

Algorithm - Stage 2: Algorithm

On each node γ in $\mathrm{C_R}$

If there exists $\mathrm{i} \in A_2$ such that $\gamma[i] = 0$

Output 0

Else

Output 1

Algorithm at $\boldsymbol{\gamma_2}$

$A_2 = \{3, 4\}$

$$\gamma_2 = 100101$$

Selection – Example

Algorithm - Stage 2: Algorithm

On each node γ in $\mathrm{C_R}$

If there exists $\mathrm{i} \in A_2$ such that $\gamma[i] = 0$

Output 0

Else

Output 1

Algorithm at $\boldsymbol{\gamma_2}$

$$A_2 = \{3, 4\}$$

$$\gamma_2 = 100101$$

=> Output 0

Selection – Example

Algorithm - Recap

$A = A_1 A_2$

$A_1 = \lfloor \log(diam(R)) \rfloor$
A_2 : For all $\gamma_{j,j>0}$, there exists $i \in A_2$ such that $\gamma_j[i] = 0$ and $\gamma_0[i] = 1$

On each node v
    Run $r = 2^{A_1}$ rounds to learn $\Lambda(r, v)$
    If $v \neq \max(\Lambda(r, v))$ output 0

On each node γ in $\mathrm{C_R}$
    If there exists i $\in A_2$ such that $\gamma[i] = 0$
        Output 0
    Else
        Output 1

Selection – Example

Algorithm - Size of Advice

$A_1 = \lfloor \log diam(R) \rfloor$

Size of A_1 is $\mathrm{O}\big(\log \log diam(R)\big)$

$A_2 =$ is a set of at most $|C_R|$ indices

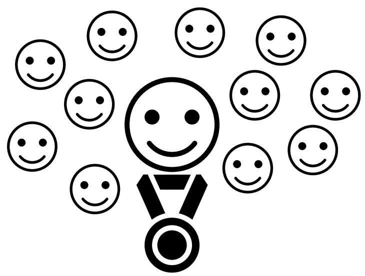
Size of each index is $\mathrm{O}\big(\log \log diam(R)\big)$
$|C_R|$ is a constant

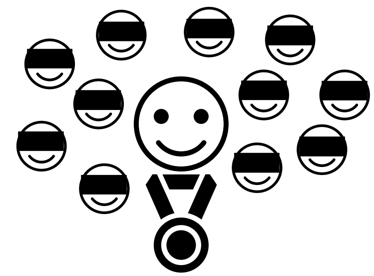
Size of A_2 is $\mathrm{O}\big(|C_R| \cdot \log \log diam(R)\big) = \mathrm{O}\big(\log \log diam(R)\big)$

Size of advice A is $\mathrm{O}\big(\log \log diam(R)\big)$

Note that the time constraint is also respected.

Summary





- Election vs Selection
- Algorithm with advice
- Measure of difficulty – size of advice
- Results
- Algorithm overview for selection in time linear in the diameter (upper bound)

Questions ?

Related Work

- Message complexity
- Non-unique labels
- Non-labelled graphs
- Election of arbitrary node
- Algorithms with advice for other problems
- Different advice for each node
- Different kind of difficulty measurement.