Gathering and Exclusive Searching on Rings under Minimal Assumptions

Gianlorenzo D'Angelo[1] Alfredo Navarra[1] Nicolas Nisse[2]

[1] Dipartimento di Matematica e Informatica, Univ. degli Studi di Perugia, Italy
[2] Inria and Univ. Nice Sophia Antipolis, CNRS, I3S, France

ICDCN 2014

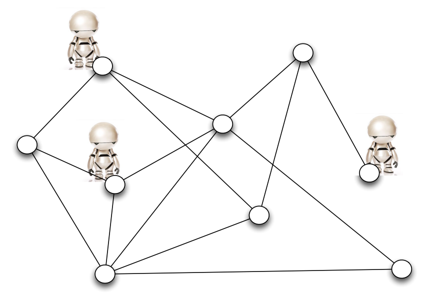
Coimbatore, January 5th, 2014

Let's throw some "stupid" robots in a graph

What can they do?

Distributed computing with robots

Coordination of a team robots in graphs.
Perform some task in a distributed setting.



Robots occupy nodes and slide along edges

Distributed Model

Robots' abilities and Model of computation

Very weak Robots

- oblivious (no memory of past actions)
- anonymous and uniform (cannot be distinguished)
- no chirality (no common sense of orientation)
- no explicit way of communication (deaf and mute)

Look-Compute-Move (CORDA)

Each robot operates in asynchronous cycles

1. take a *snapshot* of the graph — Look
2. decide what to do (to move or not) — Compute
3. eventually execute the move — Move

Robots' abilities and Model of computation

Very weak Robots

- oblivious (no memory of past actions)
- anonymous and uniform (cannot be distinguished)
- no chirality (no common sense of orientation)
- no explicit way of communication (deaf and mute)

Look-Compute-Move (CORDA)

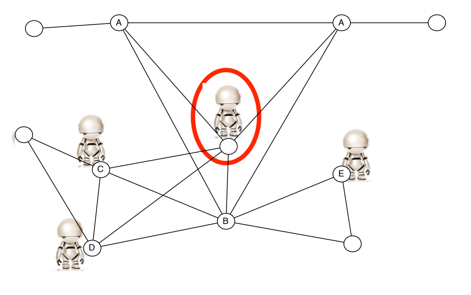
Each robot operates in asynchronous cycles

1. take a *snapshot* of the graph — Look
2. decide what to do (to move or not) — Compute
3. eventually execute the move — Move

In other words...

Robot at node v:

configuration (graph + positions of robots) $\rightarrow$ neighbor in $N[v]$ to move



Example: Red Robot sees C, D, E occupied...

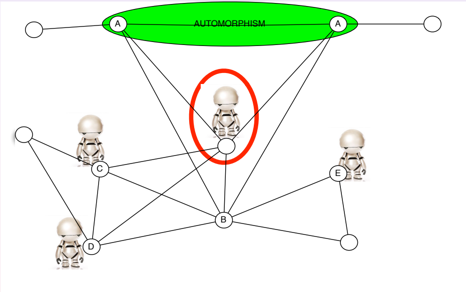
...and must decide to move or stay

BUT...

In other words...

Robot at node v:

configuration (graph + positions of robots) $\rightarrow$ neighbor in $N[v]$ to move



Example: Red Robot sees C, D, E occupied...

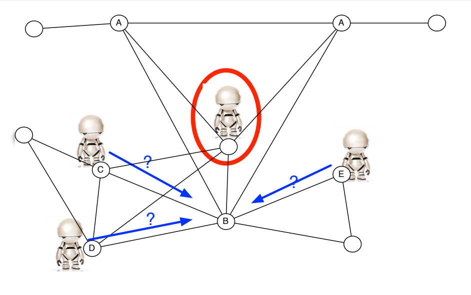
...and must decide to move or stay

Cannot distinguish the 2 nodes A (symmetry)

In other words...

Robot at node v:

configuration (graph + positions of robots) $\rightarrow$ neighbor in $N[v]$ to move



Example: Red Robot sees C, D, E occupied...

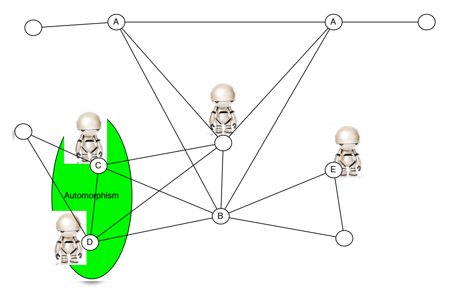
...and must decide to move or stay

Other Robot(s) may have already moved to B (asynchronicity)

In other words...

Robot at node v:

configuration (graph + positions of robots) $\rightarrow$ neighbor in $N[v]$ to move

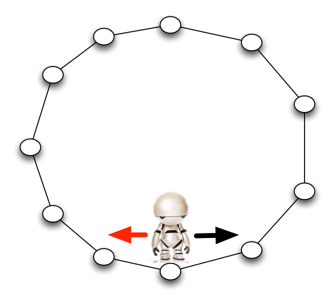


Other difficulty: Exclusivity constraint...

(at most one robot per node)

Robots at C and D cannot move $\Rightarrow$ otherwise collision

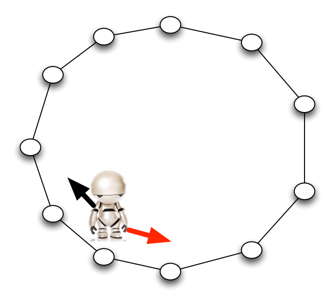
Simple Example: one or two robots in a cycle



1 Robot: cannot distinguish its 2 neighbors (nodes are anonymous)

Adversarial Model: worst case decision

Simple Example: one or two robots in a cycle



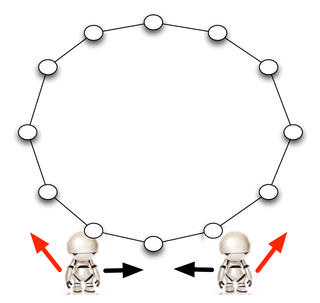
1 Robot: cannot distinguish its 2 neighbors (nodes are anonymous)

Adversarial Model: worst case decision

$\Rightarrow$ The robot oscillates along the same edge

Simple Example: one or two robots in a cycle

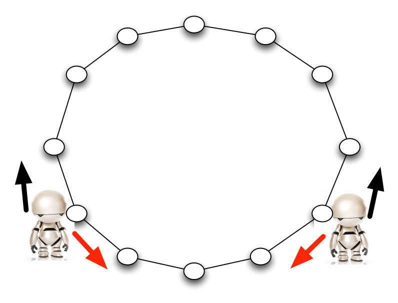
Exclusivity constraint: the 2 Robots must not collide in a node



Initially, Robots have to move in opposite direction

Simple Example: one or two robots in a cycle

Exclusivity constraint: the 2 Robots must not collide in a node



Either, at some step, they change of direction

$\Rightarrow$ Each robot oscillates along some edge

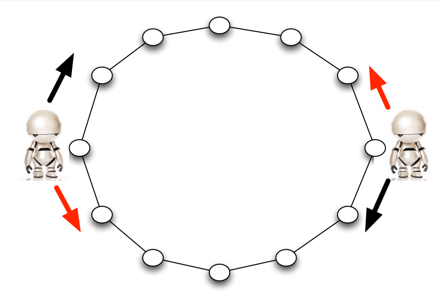
Simple Example: one or two robots in a cycle

Exclusivity constraint: the 2 Robots must not collide in a node

Or, they eventually reach periodic configuration... (case *n* even)

Simple Example: one or two robots in a cycle

Exclusivity constraint: the 2 Robots must not collide in a node

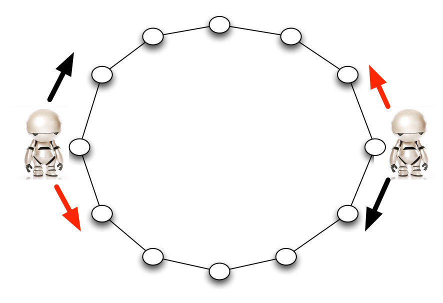


Or, they eventually reach periodic configuration... (case *n* even)

$\Rightarrow$ ...and turn around the cycle

Simple Example: one or two robots in a cycle

Exclusivity constraint: the 2 Robots must not collide in a node

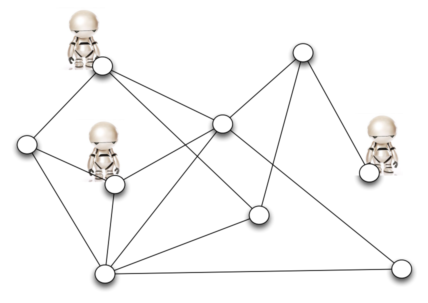


Almost nothing is possible with at most 2 Robots in a cycle (in CORDA)

2 Tasks:
Gathering and Graph Searching

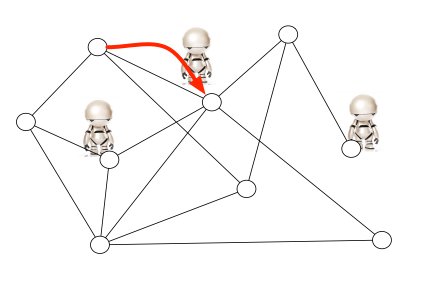
Gathering

Gathering: all robots eventually occupy the same node (a.k.a. *Rendez-vous*)



Gathering

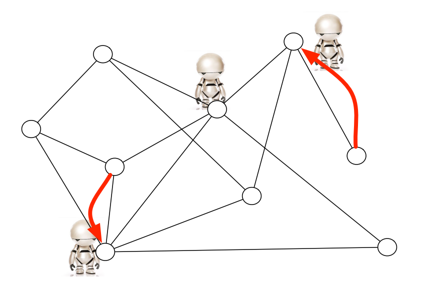
Gathering: all robots eventually occupy the same node (a.k.a. *Rendez-vous*)



Robots occupy nodes and slide along edges

Gathering

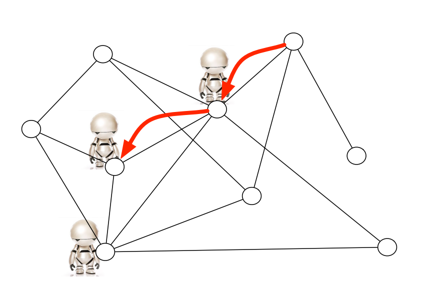
Gathering: all robots eventually occupy the same node (a.k.a. *Rendez-vous*)



Robots occupy nodes and slide along edges

Gathering

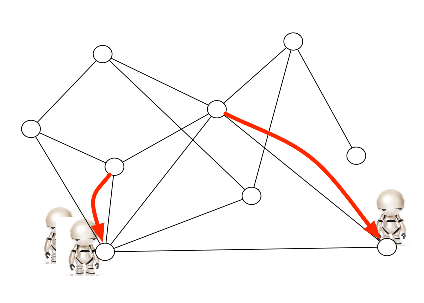
Gathering: all robots eventually occupy the same node (a.k.a. *Rendez-vous*)



Robots occupy nodes and slide along edges

Gathering

Gathering: all robots eventually occupy the same node (a.k.a. *Rendez-vous*)

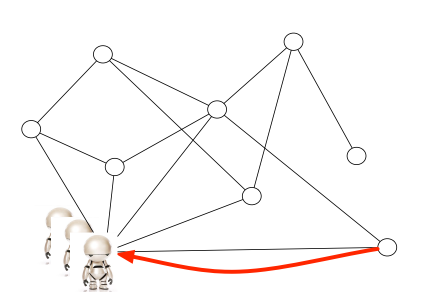


Can we distinguish the number of robots on a node?

local multiplicity detection: a robot knows if it is alone on its node or not

Gathering

Gathering: all robots eventually occupy the same node (a.k.a. *Rendez-vous*)



Can we distinguish the number of robots on a node?

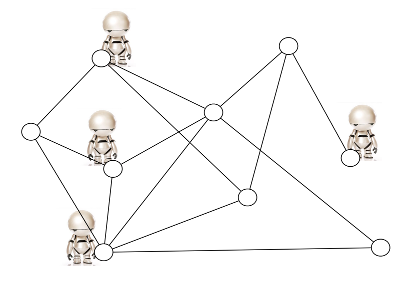
local multiplicity detection: a robot knows if it is alone on its node or not

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



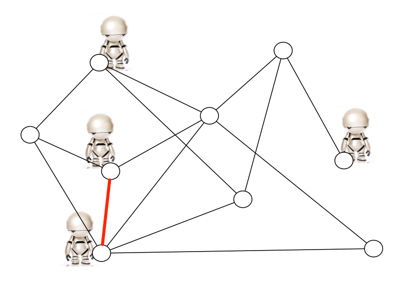
Clear an edge: slide along it OR occupy both its ends

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



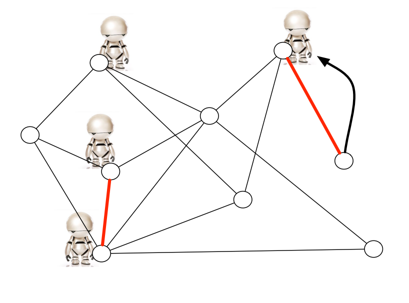
Clear an edge: slide along it OR occupy both its ends

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



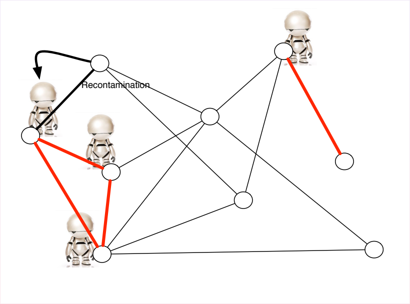
Clear an edge: slide along it OR occupy both its ends

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



Clear an edge: slide along it OR occupy both its ends

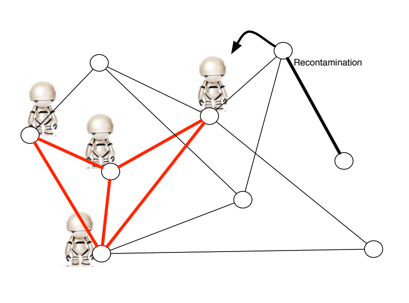
Recontamination: when *cleared* edge not *protected* by Robot

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



Clear an edge: slide along it OR occupy both its ends

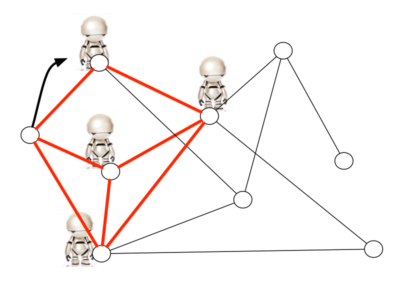
Recontamination: when *cleared* edge not *protected* by Robot

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



Clear an edge: slide along it OR occupy both its ends

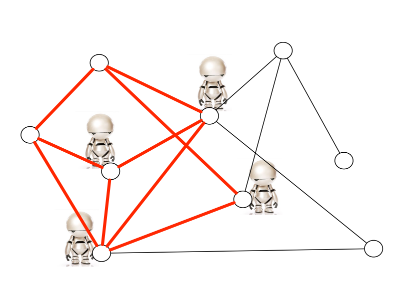
Recontamination: when *cleared* edge not *protected* by Robot

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



Clear an edge: slide along it OR occupy both its ends

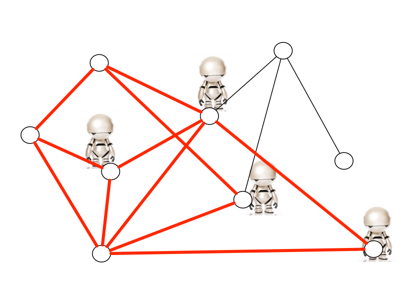
Recontamination: when *cleared* edge not *protected* by Robot

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



Clear an edge: slide along it OR occupy both its ends

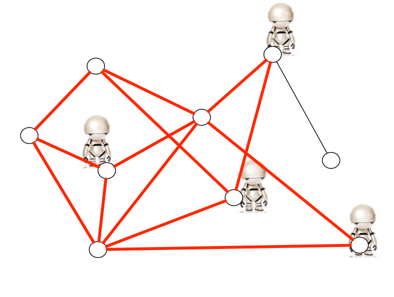
Recontamination: when *cleared* edge not *protected* by Robot

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



Clear an edge: slide along it OR occupy both its ends

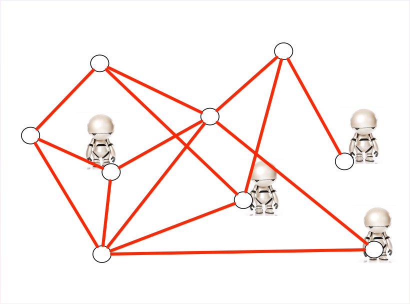
Recontamination: when *cleared* edge not *protected* by Robot

Graph Searching

Exclusive Graph Searching: robots must *clear* the graph G

$\Leftrightarrow$ robots must catch an invisible mobile item in G

Exclusivity constraint: at most one robot per node



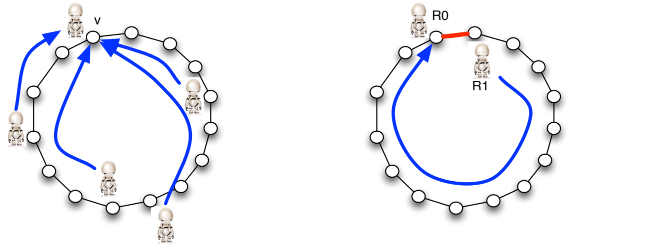
Clear an edge: slide along it OR occupy both its ends

Recontamination: when *cleared* edge not *protected* by Robot

Easy in the ring?

These tasks are easy in the ring when:

- centralized setting
- nodes/robots have identifiers
- sense of direction ...



Gathering: all robots reach v

Graph Searching: one Robot (R0) never moves
all other robots go clockwise

Note: 2 robots are sufficient to search a ring

The questions

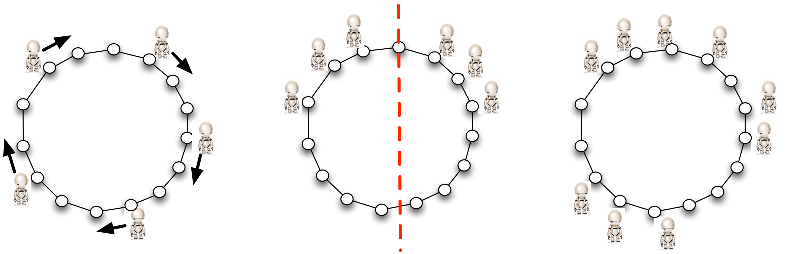
Consider k robots in an n-node ring

- Can they solve the Gathering Problem? the Graph Searching?
- From which initial configurations?

The questions

Consider k robots in an n-node ring

- Can they solve the Gathering Problem? the Graph Searching?
- From which initial configurations?



example of **Periodic configuration**

example of **Symmetric aperiodic configuration**

example of aperiodic and asymmetric = **Rigid configuration**

Note: a configuration is symmetric and aperiodic iff a unique axis

Related Work and Contribution

Related work: Coordination problems in CORDA model

<table>
<tr><th></th><th>Paths</th><th>Trees</th><th>Grids</th><th>Arbitrary graphs</th></tr>
<tr><td>Exploration with stop</td><td colspan="2">Flocchini,Ilcinkas,Pelc,Santoro
IPL11 | TCS10
+ multiplicity detection</td><td></td><td>Chalopin,Flocchini
Mans,Santoro,WG'10
+ local edge labeling</td></tr>
<tr><td>Exclusive Perpetual Exploration</td><td></td><td></td><td>Baldoni et al, IPL08
+ orientation
Bonnet et al, OPODIS'11
no orientation</td><td></td></tr>
<tr><td>Rendez-vous/ Gathering</td><td></td><td></td><td></td><td>DiStefano,Navarra
SIROCCO'13
+global mult. detection</td></tr>
<tr><td>Exclusive Perpetual Graph Searching</td><td colspan="2">Blin,Burman,Nisse
DISC'12,ESA'13</td><td></td><td></td></tr>
</table>

Related work: Coordination problems in CORDA model

	Rings
Exploration with stop	Flocchini,Ilcinkas,Pelc,Santoro OPODIS'07 + multiplicity detection
Exclusive Perpetual Exploration	Blin *et al*, DISC'10 D'Angelo,DiStephano,Navarra,Nisse,Suchan, APDCM'13
Rendez-vous/Gathering	
+global mult. detection	D'Angelo,DiStephano,Navarra,DISC'12
+local mult. detection	Izumi *et al* SIROCCO'10 (partial solutions)
	Kamei *et al*, SIROCCO'11,MFCS'12 (partial solutions)
	D'Angelo,DiStephano,Navarra,Nisse,Suchan, APDCM'13
Exclusive Perpetual Graph Searching	D'Angelo,DiStephano,Navarra,Nisse,Suchan, APDCM'13

All these works (but some of Kamei *et al*) assume:

aperiodic AND asymmetric initial configuration

Unified algorithm: D'Angelo,DiStephano,Navarra,Nisse,Suchan, APDCM'13

Related work: Coordination problems in CORDA model

	Rings
Exploration with stop	Flocchini,Ilcinkas,Pelc,Santoro OPODIS'07 + multiplicity detection
Exclusive Perpetual Exploration	Blin *et al*, DISC'10 D'Angelo,DiStephano,Navarra,Nisse,Suchan, APDCM'13
Rendez-vous/Gathering	
+global mult. detection	D'Angelo,DiStephano,Navarra,DISC'12
+local mult. detection	Izumi *et al* SIROCCO'10 (partial solutions)
	Kamei *et al*, SIROCCO'11,MFCS'12 (partial solutions)
	D'Angelo,DiStephano,Navarra,Nisse,Suchan, APDCM'13
Exclusive Perpetual Graph Searching	D'Angelo,DiStephano,Navarra,Nisse,Suchan, APDCM'13

In this work: initial configuration only aperiodic

i.e., it may be one axis of symmetry

This (almost) fully characterize the case of Gathering in rings.

Our contribution

Consider any n-node ring and $k \geq 1$ robots

Gathering in Ring

impossible	$k = 2$ or periodic configuration or 1 axis of symmetry passing through two edges (already known)
unknown cases	$k = 4$ or $k \geq n - 4$
possible	all other cases

Exclusive Perpetual Graph Searching in Ring

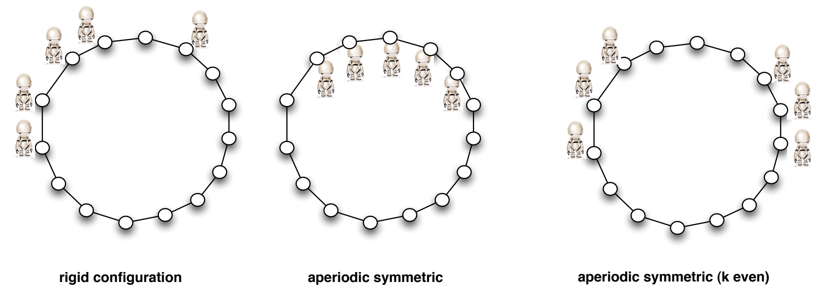
impossible	$n \geq 9$ or $k \leq 3$ or $k \geq n - 2$ [ADN+13] or (k even and 1 axis of symmetry passing through an empty node)
possible	all aperiodic configurations not listed above (but for $k = 4$ or ($n = 10$ and $k \in \{5, 6\}$))
unknown cases	$k = 4$ or ($n = 10$ and $k \in \{5, 6\}$) or periodic configurations

Main idea of the algorithms

2 Phases

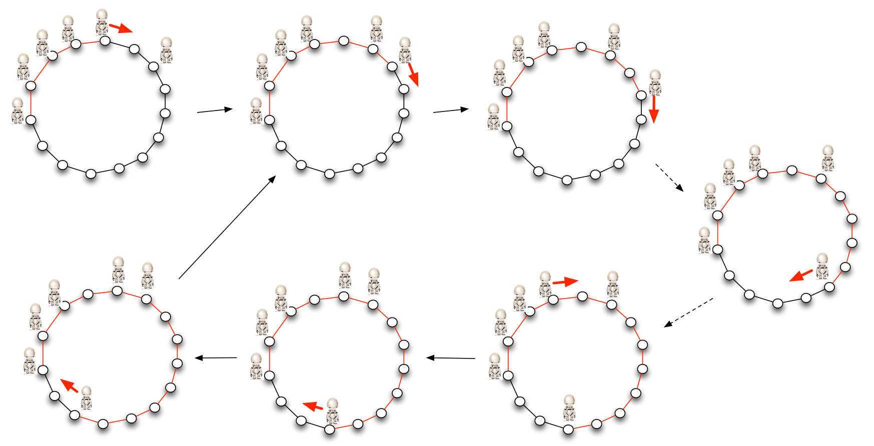
1. reach one of 3 specific configurations — difficult part
2. perform the task from these configurations — easy part

(hidden difficulty: make sure that the 2 phases do not interfer)



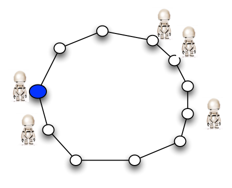
Example of Phase 2: Graph Searching from aperiodic configuration [APDCM'13]

Key point: at each step, a unique Robot moves (no ambiguity)



Flavour of Phase 1: supermin representation

Representation of configuration: binary string
0=Robot, 1=empty node



which string?

$(0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0)$
$(1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1)$
$(0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1)$
$\ldots$

supermin: minimum in lexicographical order $(0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1)$

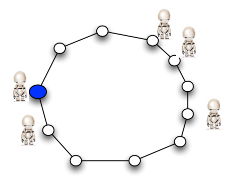
Lemma [D'Angelo,DiStephano,Navarra DISC'12]

configuration aperiodic $\Leftrightarrow$ unique supermin
configuration symmetric aperiodic $\Rightarrow$ 2 supermins

So we can (almost) distinguish all Robots

Flavour of Phase 1: supermin representation

Representation of configuration: binary string
0=Robot, 1=empty node



which string?

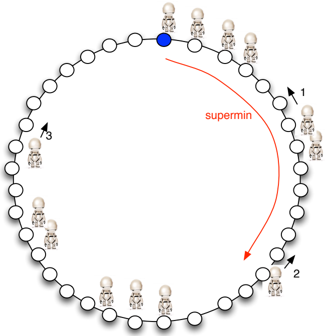
$(0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0)$
$(1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1)$
$(0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1)$
$\dots$

supermin: minimum in lexicographical order $(0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1)$

Lemma [D'Angelo,DiStephano,Navarra DISC'12]

configuration aperiodic $\Leftrightarrow$ unique supermin
configuration symmetric aperiodic $\Rightarrow$ 2 supermins

So we can (almost) distinguish all Robots



Example of rigid configuration (unique supermin)

- unique supermin ⇒ distinguish Robots
- *segment*: maximal sequence of consecutive Robots
- segments ordered as the supermin

3 simple rules
(assuming at least 2 robots adjacent)

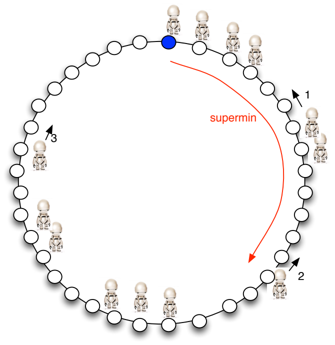
1. first robot of second "segment"
2. first robot of third "segment"
3. last robot of last "segment"

Algorithm starting from rigid configuration [APDCM'13]

for each rigid configuration, define the rule to be applied such that

- at each step, a unique Robot moves (no ambiguity)
- each move reduces the supermin (⇒ eventually reach the desired configuration)

Flavour of Phase 1: few rules



Example of rigid configuration (unique supermin)

- unique supermin ⇒ distinguish Robots
- *segment*: maximal sequence of consecutive Robots
- segments ordered as the supermin

3 simple rules
(assuming at least 2 robots adjacent)

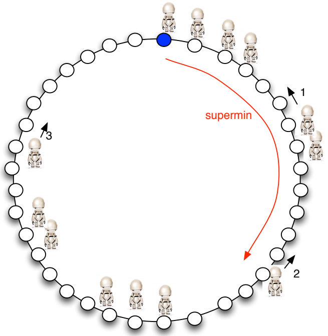
1. first robot of second "segment"
2. first robot of third "segment"
3. last robot of last "segment"

Algorithm starting from rigid configuration [APDCM'13]

for each rigid configuration, define the rule to be applied such that

- at each step, a unique Robot moves (no ambiguity)
- each move reduces the supermin (⇒ eventually reach the desired configuration)

Flavour of Phase 1: few rules



Example of rigid configuration (unique supermin)

- unique supermin $\Rightarrow$ distinguish Robots
- *segment*: maximal sequence of consecutive Robots
- segments ordered as the supermin

3 simple rules (assuming at least 2 robots adjacent)

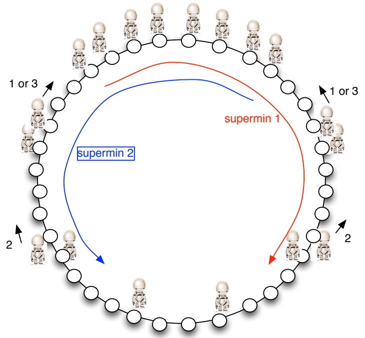
1. first robot of second "segment"
2. first robot of third "segment"
3. last robot of last "segment"

Algorithm starting from rigid configuration [APDCM'13]

for each rigid configuration, define the rule to be applied such that

- at each step, a unique Robot moves (no ambiguity)
- each move reduces the supermin ($\Rightarrow$ eventually reach the desired configuration)

Flavour of Phase 1: symmetry and asynchronicity



Example of aperiodic symmetric configuration (two supermins)

- two supermins $\Rightarrow$ ambiguity

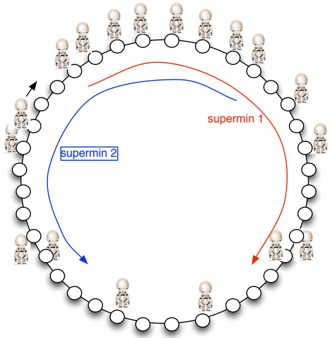
same rules but...

- two Robots may perform move

In configuration $\mathcal{C}$, R and R' must move,

if only R moves (due to asynchronicity) and reaches configuration $\mathcal{C}'$

Flavour of Phase 1: symmetry and asynchronicity



Example of aperiodic symmetric configuration (two supermins)

- two supermins $\Rightarrow$ ambiguity

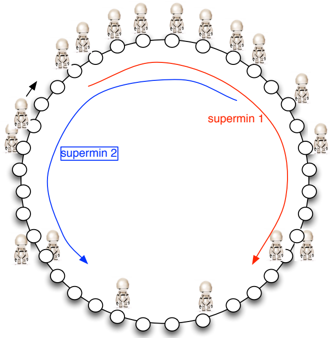
same rules but...

- two Robots may perform move

In configuration $\mathcal{C}$, R and R' must move,

if only R moves (due to asynchronicity) and reaches configuration $\mathcal{C}'$

Flavour of Phase 1: symmetry and asynchronicity



Example of aperiodic symmetric configuration (two supermins)

- two supermins $\Rightarrow$ ambiguity

same rules but...

- two Robots may perform move

In configuration $\mathcal{C}$, R and R' must move,

if only R moves (due to asynchronicity) and reaches configuration $\mathcal{C}'$

- we must ensure that R' executes his move in configuration $\mathcal{C}'$
  i.e., R' must be the only robot to move in $\mathcal{C}'$

Algorithm starting from aperiodic symmetric configuration

tedious characterization of aperiodic symmetric configurations to avoid ambiguity

Conclusion

Gathering in Ring

impossible	$k = 2$ or periodic configuration or 1 axis of symmetry passing through two edges (already known)
unknown cases	$k = 4$ or $k \geq n - 4$
possible	all other cases

Exclusive Perpetual Graph Searching in Ring

impossible	$n \geq 9$ or $k \leq 3$ or $k \geq n - 2$ [ADN+13] or (k even and 1 axis of symmetry passing through an empty node)
possible	all aperiodic configurations not listed above (but for $k = 4$ or ($n = 10$ and $k \in \{5, 6\}$))
unknown cases	$k = 4$ or ($n = 10$ and $k \in \{5, 6\}$) or periodic configurations

other coordination problems?
other graph topologies?

...

Thank you!