Algorithmique Répartie

Jeremy Krebs - Guillaume Soulié

Université Paris Saclay

9 novembre 2017

Background and Motivations :

Background and Motivations :

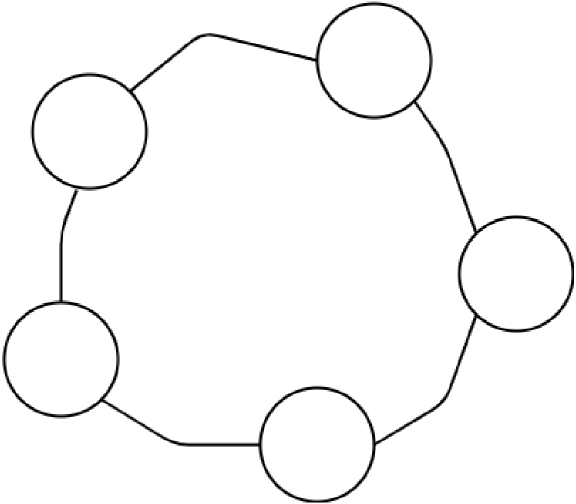
- **mobile robot network** goal : achieve tasks by a team of mobile robot with weak capacity.

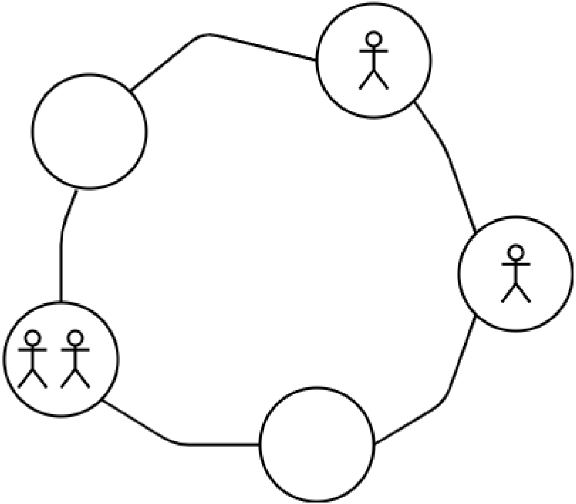
Background and Motivations :

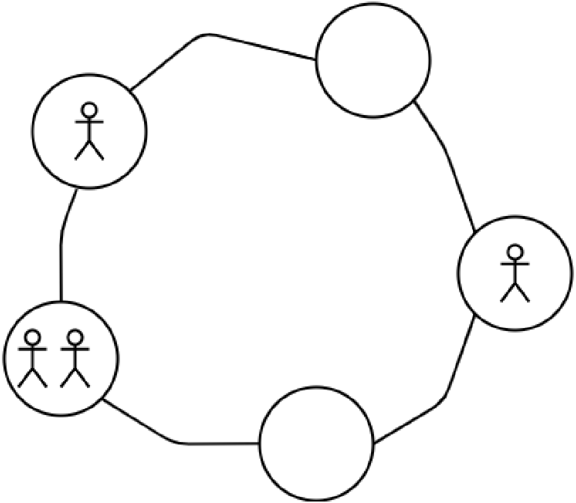
- **mobile robot network** goal : achieve tasks by a team of mobile robot with weak capacity.
- **Pioneering work** Suzuki, I., & Yamashita, M. (1999). Distributed anonymous mobile robots : Formation of geometric patterns.

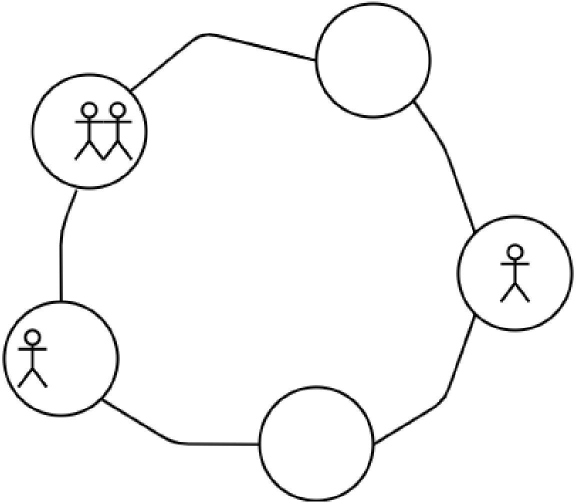
Background and Motivations :

- **mobile robot network** goal : achieve tasks by a team of mobile robot with weak capacity.
- **Pioneering work** Suzuki, I., & Yamashita, M. (1999). Distributed anonymous mobile robots : Formation of geometric patterns.
- it studies **self-stabilizing** algorithms for anonymous and oblivious robots in uniform ring network.

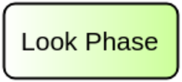




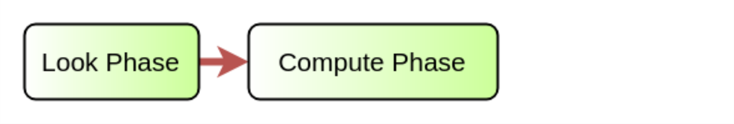




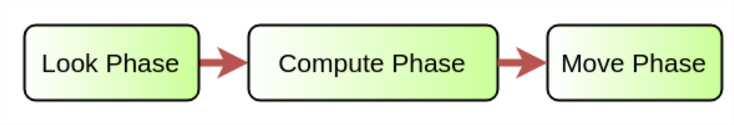
Each robot repeat cycles. Cycle has three phases :



Each robot repeat cycles. Cycle has three phases :



Each robot repeat cycles. Cycle has three phases :



Our Contribution : investigate the difficulty of probabilistic self-stabilizing algorithms using weak assumptions.

Our Contribution : investigate the difficulty of probabilistic self-stabilizing algorithms using weak assumptions.

- no such algorithms on very weak condition (ASYNC or global-weak multiplicity)

Our Contribution : investigate the difficulty of probabilistic self-stabilizing algorithms using weak assumptions.

- no such algorithms on very weak condition (ASYNC or global-weak multiplicity)
- we propose three algorithms :
  self-stabilizing gathering algorithm
  self-stabilizing orientation algorithm
  self-stabilizing formation algorithm

Related work :

Related work :

- Several work on **continuous model.**

Related work :

- Several work on **continuous model.**
- other work on **discrete model :**

Related work :

- Several work on **continuous model.**
- other work on **discrete model :**
    - deterministic / stochastic

Related work :

- Several work on **continuous model.**
- other work on **discrete model :**
  - deterministic / stochastic
  - none of them are self stabilizing

There are a few hypotheses on the robots :

There are a few hypotheses on the robots :

- Robots are identical. No distinction, same algorithm,

There are a few hypotheses on the robots :

- Robots are identical. No distinction, same algorithm,
- Robots are oblivious. They have no memory of their moves,

There are a few hypotheses on the robots :

- Robots are identical. No distinction, same algorithm,
- Robots are oblivious. They have no memory of their moves,
- Robots cannot communicated directly.

There are a few hypotheses on the robots :

- Robots are identical. No distinction, same algorithm,
- Robots are oblivious. They have no memory of their moves,
- Robots cannot communicated directly.

However they can observe the positions of the other robots, and it is one of those two cases :

There are a few hypotheses on the robots :

- Robots are identical. No distinction, same algorithm,
- Robots are oblivious. They have no memory of their moves,
- Robots cannot communicated directly.

However they can observe the positions of the other robots, and it is one of those two cases :

- Global-Strong Multiplicity Detection

There are a few hypotheses on the robots :

- Robots are identical. No distinction, same algorithm,
- Robots are oblivious. They have no memory of their moves,
- Robots cannot communicated directly.

However they can observe the positions of the other robots, and it is one of those two cases :

- Global-Strong Multiplicity Detection
- Local-Strong and Global-Weak Multiplicity Detection

The scheduler can be of two types :

The scheduler can be of two types :

SSYNC Semi-Synchronous - For each round, a set of robots are activated/executed at the same time.

The scheduler can be of two types :

SSYNC Semi-Synchronous - For each round, a set of robots are activated/executed at the same time.

ASYNC Asynchronous - The robots are activated/executed asynchronously

Gathering Problem : The goal of the gathering problem is to group all the robots on the same node.

Gathering Problem : The goal of the gathering problem is to group all the robots on the same node.

Orientation Problem : The goal of the set formation problem is to make the robots gather in a configuration such that :

Gathering Problem : The goal of the gathering problem is to group all the robots on the same node.
Orientation Problem : The goal of the set formation problem is to make the robots gather in a configuration such that :

- There is exactly one tower node

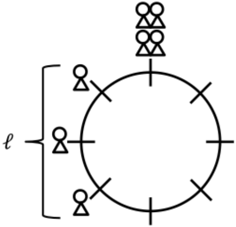
Gathering Problem : The goal of the gathering problem is to group all the robots on the same node.

Orientation Problem : The goal of the set formation problem is to make the robots gather in a configuration such that :

- There is exactly one tower node
- There is a 1-robot block of size l

Gathering Problem : The goal of the gathering problem is to group all the robots on the same node.
Orientation Problem : The goal of the set formation problem is to make the robots gather in a configuration such that :

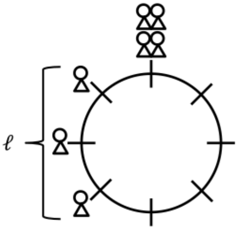
- There is exactly one tower node
- There is a 1-robot block of size l



Gathering Problem : The goal of the gathering problem is to group all the robots on the same node.

Orientation Problem : The goal of the set formation problem is to make the robots gather in a configuration such that :

- There is exactly one tower node
- There is a 1-robot block of size l



Set formation problem : The goal of the set formation problem is to gather the robots in a specific predefined configuration.

Proof of the non existence of gathering algorithm in two weak conditions :

Proof of the non existence of gathering algorithm in two weak conditions :

- ASYNC Model

Proof of the non existence of gathering algorithm in two weak conditions :

- ASYNC Model
- SSYNC Model - global-weak & local-strong multiplicity

Proof of the non existence of gathering algorithm in two weak conditions :

- ASYNC Model
- SSYNC Model - global-weak & local-strong multiplicity

How to prove non existence of an algorithm ?

Proof of the non existence of gathering algorithm in two weak conditions :

- ASYNC Model
- SSYNC Model - global-weak & local-strong multiplicity

How to prove non existence of an algorithm ? **play the scheduler**

(absurd) We assume that there is a algorithm A which works with probability at least $p(k, n)$.

(absurd) We assume that there is a algorithm A which works with probability at least $p(k, n)$.
We assume that we have a procedure $Proc(X)$ such as :

1. $Proc(X)$ activate each robot at least one

(absurd) We assume that there is a algorithm A which works with probability at least $p(k, n)$.
We assume that we have a procedure $Proc(X)$ such as :

1. $Proc(X)$ activate each robot at least one
2. $Proc(X)$ is complete in finite time

(absurd) We assume that there is a algorithm A which works with probability at least $p(k, n)$.
We assume that we have a procedure $Proc(X)$ such as :

1. $Proc(X)$ activate each robot at least one
2. $Proc(X)$ is complete in finite time
3. if $P(X) :=$ probability that there is no robot change during $Proc(X)$ execution, then $\lim_{X->\inf} P(X) = 1$

(absurd) We assume that there is a algorithm A which works with probability at least $p(k, n)$.
We assume that we have a procedure $Proc(X)$ such as :

1. $Proc(X)$ activate each robot at least one
2. $Proc(X)$ is complete in finite time
3. if $P(X) :=$ probability that there is no robot change during $Proc(X)$ execution, then $\lim_{X->\inf} P(X) = 1$

Probability that A achieve gathering in j cycle :

$$P^* < (1 - P(X_1)) + (1 - P(X_2)) + ... + (1 - P(X_j)) < p \square$$

Construct $Proc(X)$:

Construct $Proc(X)$:

- one robot per node

Construct $Proc(X)$:

- one robot per node
- $Q = +i$ $(0 \leq i \leq n)$ indicates $(v_0, ..., v_{i-1})$ decides to move forward.

Construct $Proc(X)$:

- one robot per node
- $Q = +i$ $(0 \leq i \leq n)$ indicates $(v_0, ..., v_{i-1})$ decides to move forward.
  We want to achieve $Q = +n$ or $Q = -n$.

In $proc(X)$, scheduler repeat following steps :

- if $Q = 0$: look and compute phases on robot r on v_0 :
  - if r want to stay $=> Q := 0$
  - elif r want to move forward $=> Q := 1$
  - elif r want to move backward $=> Q := -1$

In $proc(X)$, scheduler repeat following steps :

- if $Q = 0$: look and compute phases on robot r on v_0 :
  - if r want to stay $=> Q := 0$
  - elif r want to move forward $=> Q := 1$
  - elif r want to move backward $=> Q := -1$
- if $Q = +i$: look and compute phases on robot r on v_{+i} :
  - if r want to stay $=> Q := +i$
  - if r want to move forward $=> Q := +(i+1)$
  - if r want to move backward : move phase for r and robot of $v_{+(i-1)} => Q := +(i-1)$.

In $proc(X)$, scheduler repeat following steps :

- if $Q = 0$: look and compute phases on robot r on v_0 :
  - if r want to stay $=> Q := 0$
  - elif r want to move forward $=> Q := 1$
  - elif r want to move backward $=> Q := -1$
- if $Q = +i$: look and compute phases on robot r on v_{+i} :
  - if r want to stay $=> Q := +i$
  - if r want to move forward $=> Q := +(i + 1)$
  - if r want to move backward : move phase for r and robot of $v_{+(i-1)} => Q := +(i - 1)$.
- if $Q = -i$: ...

In $proc(X)$, scheduler repeat following steps :

- if $Q = 0$: look and compute phases on robot r on v_0 :
  - if r want to stay $=> Q := 0$
  - elif r want to move forward $=> Q := 1$
  - elif r want to move backward $=> Q := -1$
- if $Q = +i$: look and compute phases on robot r on v_{+i} :
  - if r want to stay $=> Q := +i$
  - if r want to move forward $=> Q := +(i+1)$
  - if r want to move backward : move phase for r and robot of $v_{+(i-1)} => Q := +(i-1)$.
- if $Q = -i$: ...

Stop when $Q = +n$ or $Q = -n$ (or X steps).

- prop 1 and 2 or clearly statisfied.

- prop 1 and 2 or clearly statisfied.
- for prop 3 :
  - $P(Q_{h+1} -> Q_h + 1) = p_1$
  - $P(Q_{h+1} -> Q_h - 1) = p_2$
  - $P(Q_{h+1} -> Q_h) = 1 - p_1 - p_2$

- prop 1 and 2 or clearly statisfied.
- for prop 3 :
  - $P(Q_{h+1} -> Q_h + 1) = p_1$
  - $P(Q_{h+1} -> Q_h - 1) = p_2$
  - $P(Q_{h+1} -> Q_h) = 1 - p_1 - p_2$

  This implies from any configuration Q, $Q = +/- n$ is achieved is less than $2n$ step with probability $p > p_1^{2n} + p_2^{2n}$.

- prop 1 and 2 or clearly statisfied.
- for prop 3 :
  - $P(Q_{h+1} -> Q_h + 1) = p_1$
  - $P(Q_{h+1} -> Q_h - 1) = p_2$
  - $P(Q_{h+1} -> Q_h) = 1 - p_1 - p_2$

  This implies from any configuration Q, $Q = +/- n$ is achieved is less than $2n$ step with probability $p > p_1^{2n} + p_2^{2n}$.

$$P(X) \geq 1 - (1 - p_1^{2n} - p_2^{2n})^{\frac{X}{2n}}$$

- prop 1 and 2 or clearly statisfied.
- for prop 3 :
  - $P(Q_{h+1} -> Q_h + 1) = p_1$
  - $P(Q_{h+1} -> Q_h - 1) = p_2$
  - $P(Q_{h+1} -> Q_h) = 1 - p_1 - p_2$

  This implies from any configuration Q, $Q = +/- n$ is achieved is less than $2n$ step with probability $p > p_1^{2n} + p_2^{2n}$.

$$P(X) \geq 1 - (1 - p_1^{2n} - p_2^{2n})^{\frac{X}{2n}}$$
$$\lim_{X \to \inf} P(X) = 1$$

Gathering problem :

Gathering problem :

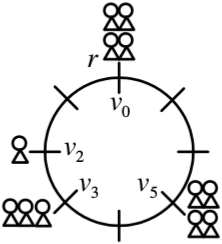
- We consider n nodes and k robots in an unoriented ring

Gathering problem :

- We consider n nodes and k robots in an unoriented ring
- For any configuration C we not M(C) the maximum number of robots on one node

Gathering problem :

- We consider n nodes and k robots in an unoriented ring
- For any configuration C we not M(C) the maximum number of robots on one node



Idea :

- If there is only once M(C)-node, then the robots "know" where to go
- If there is multiple, the idea is to try to make them move one by one so that a tower node "wins the fight". We must find a why to elect a candidate.
- If there are multiple candidates, find a way to make, in expectation, exactly one of them move

Idea :

- If there is only once M(C)-node, then the robots "know" where to go
- If there is multiple, the idea is to try to make them move one by one so that a tower node "wins the fight". We must find a why to elect a candidate.
- If there are multiple candidates, find a way to make, in expectation, exactly one of them move
- **Take care !** The scheduler is an enemy and will activate the robots in the worst way.

Let's consider the M(C) nodes :

Let's consider the M(C) nodes :

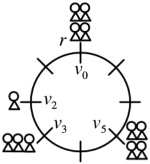
Case 1 There is only one such node : the tower can be identified by the robots and they can get closer to the tower node.

Let's consider the M(C) nodes :

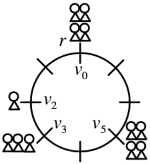
Case 1 There is only one such node : the tower can be identified by the robots and they can get closer to the tower node.

- **The scheduler is an enemy !**
- Less than M(C) nodes should move in the same direction !

Case 2 There are multiple such nodes :

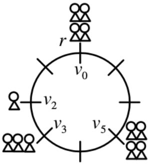


Case 2 There are multiple such nodes :



Take h_{min} the minimal distance between a M(C)-robot node and a neighboring robot node. Take V the set of nodes at distance h_{min} of a M(C)-node and R the robots on these nodes.

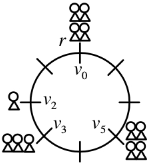
Case 2 There are multiple such nodes :



Take h_{min} the minimal distance between a M(C)-robot node and a neighboring robot node. Take V the set of nodes at distance h_{min} of a M(C)-node and R the robots on these nodes.

Cas 2.1 $|R| = 1$ - This robot gets to his closer M(C)-robot node.

Case 2 There are multiple such nodes :

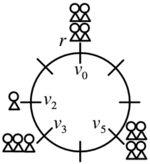


Take h_{min} the minimal distance between a M(C)-robot node and a neighboring robot node. Take V the set of nodes at distance h_{min} of a M(C)-node and R the robots on these nodes.

Cas 2.1 $|R| = 1$ - This robot gets to his closer M(C)-robot node.

Cas 2.2 $|R| > 1$ - The robots move to their close M(C)-robot node with probability $\frac{1}{2|R|}$.

Case 2 There are multiple such nodes :



Take h_{min} the minimal distance between a M(C)-robot node and a neighboring robot node. Take V the set of nodes at distance h_{min} of a M(C)-node and R the robots on these nodes.

Cas 2.1 $|R| = 1$ - This robot gets to his closer M(C)-robot node.

Cas 2.2 $|R| > 1$ - The robots move to their close M(C)-robot node with probability $\frac{1}{2|R|}$.

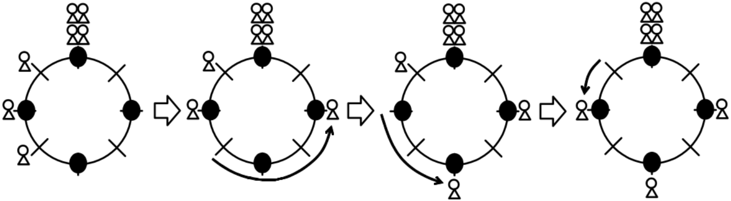
Complexity : $O(n \log k)$ rounds and $O(kn)$ moves.

We can now apply the two previous algorithms to solve the **set formation problem**.

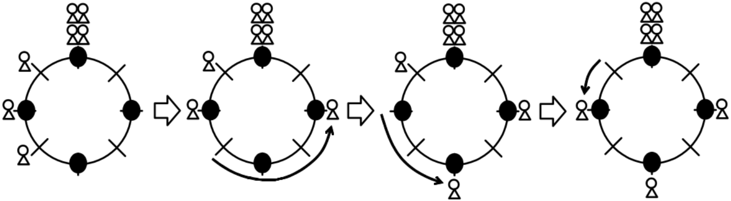
- Solve the orientation algorithm for $l = |SET| - 1$,

- Solve the orientation algorithm for $l = |SET| - 1$,
- Now that the ring is oriented, move the robots one by one

- Solve the orientation algorithm for $l = |SET| - 1$,
- Now that the ring is oriented, move the robots one by one



- Solve the orientation algorithm for $l = |SET| - 1$,
- Now that the ring is oriented, move the robots one by one



Complexity : $O((\log\ k + |SET|)n)$ rounds and $O(kn)$ moves.

Conclusion :

Conclusion :

- Our strong assumptions on the system are mandatory

Conclusion :

- Our strong assumptions on the system are mandatory
- Solving the gathering and orientation issues is very important and leads to tons of other problems solved

Conclusion :

- Our strong assumptions on the system are mandatory
- Solving the gathering and orientation issues is very important and leads to tons of other problems solved

In order to go further we could :

Conclusion :

- Our strong assumptions on the system are mandatory
- Solving the gathering and orientation issues is very important and leads to tons of other problems solved

In order to go further we could :

- Find the problems we can solve with weaker hypotheses,

Conclusion :

- Our strong assumptions on the system are mandatory
- Solving the gathering and orientation issues is very important and leads to tons of other problems solved

In order to go further we could :

- Find the problems we can solve with weaker hypotheses,
- Work with a weaker scheduler, like an oblivious one,

Conclusion :

- Our strong assumptions on the system are mandatory
- Solving the gathering and orientation issues is very important and leads to tons of other problems solved

In order to go further we could :

- Find the problems we can solve with weaker hypotheses,
- Work with a weaker scheduler, like an oblivious one,
- Work with a more complex graph than a ring.