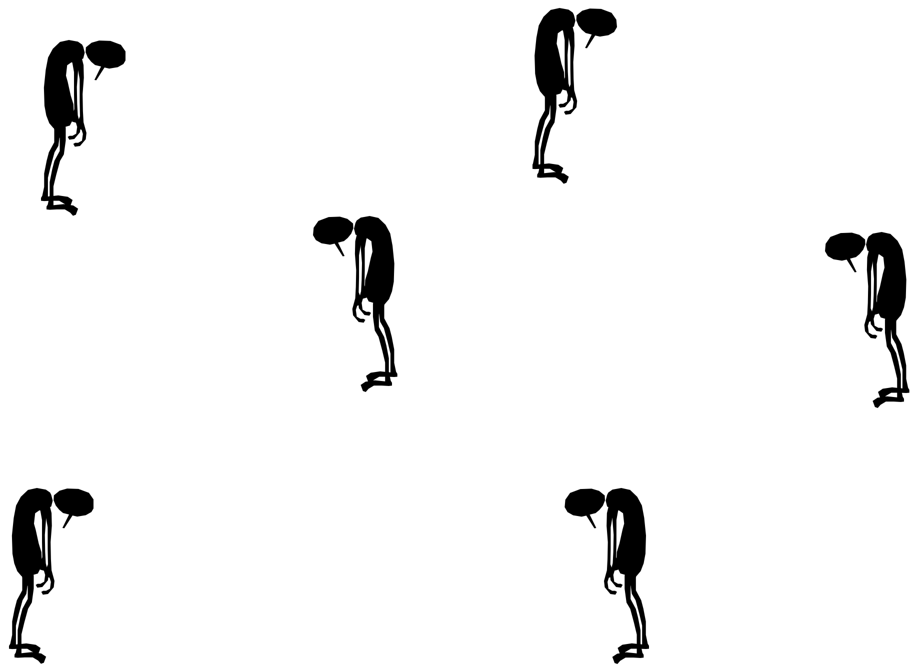
Introduction to the Gathering Problem

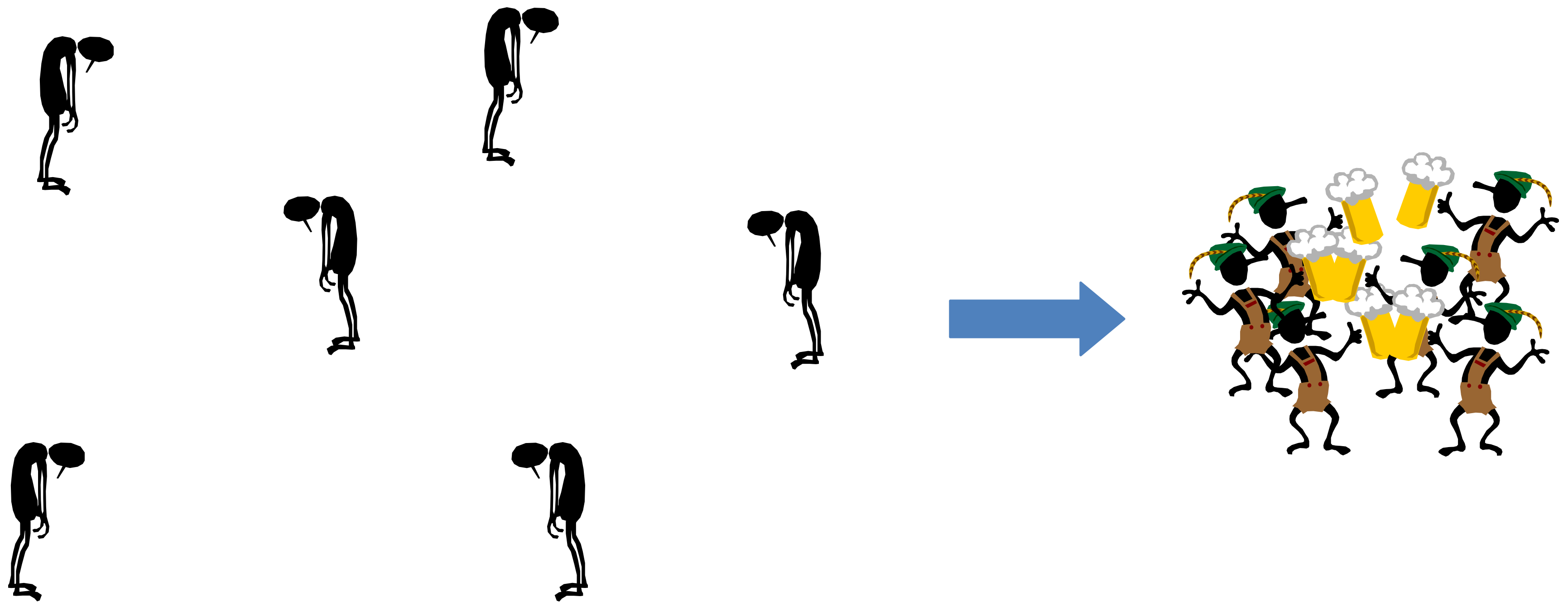
Gathering - Unlimited Visibility

Gathering - Unlimited Visibility



Initially the robots are in arbitrary distinct positions.

Gathering - Unlimited Visibility



Initially the robots are in arbitrary distinct positions.

In finite time, they gather in the same place.

Gathering
Unlimited Visibility - SYm

Ando, Oasa, Suzuki, Yamashita
Siam Journal Of Computing, 1999

- Istantaneous activities
- n=2, the problem is unsolvable

Gathering, n=2
Unlimited Visibility - SYm

In fact, since the robots have no dimension…

…and cannot bump...

…moving them towards each other is not useful…

Gathering, n=2
Unlimited Visibility - SYm

In fact, since the robots have no dimension…

…and cannot bump...

…moving them towards each other is not useful…

…but it works if they can *bump!*

Gathering

Unlimited Visibility - SYm

Ando, Oasa, Suzuki, Yamashita
Siam Journal Of Computing, 1999

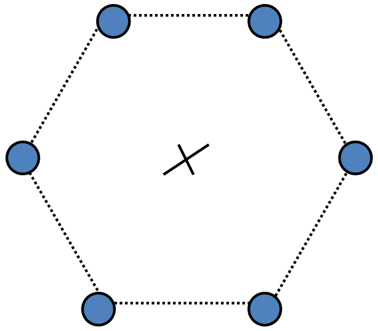
- Istantaneous activities
- n=2, the problem is unsolvable
- n>2, they provide an oblivious algorithm that let the robots gather in finite time

Gathering, ASYNC

- In spite of its apparent simplicity, this problem has been tackled in several studies
- In fact, several factors render this problem difficult to solve
  - Major problems arise from symmetric configurations....

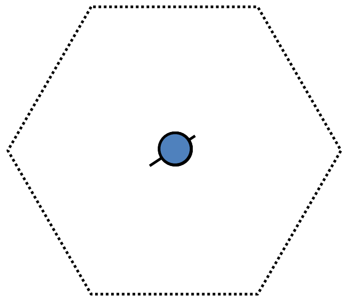
Difficulties

If at the beginning....



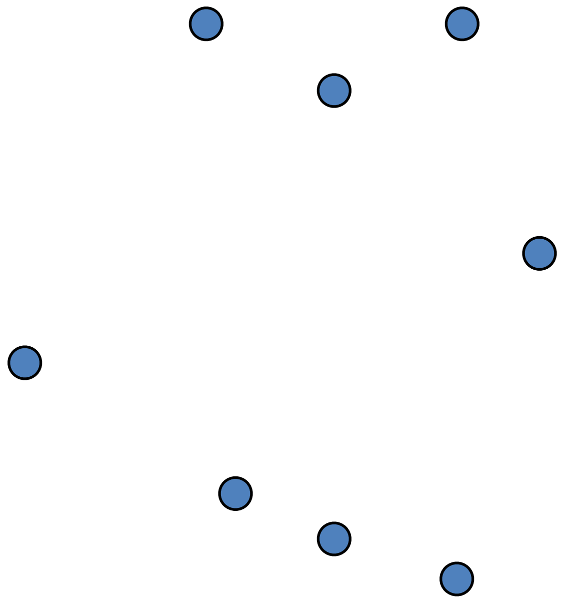
Difficulties

If at the beginning....



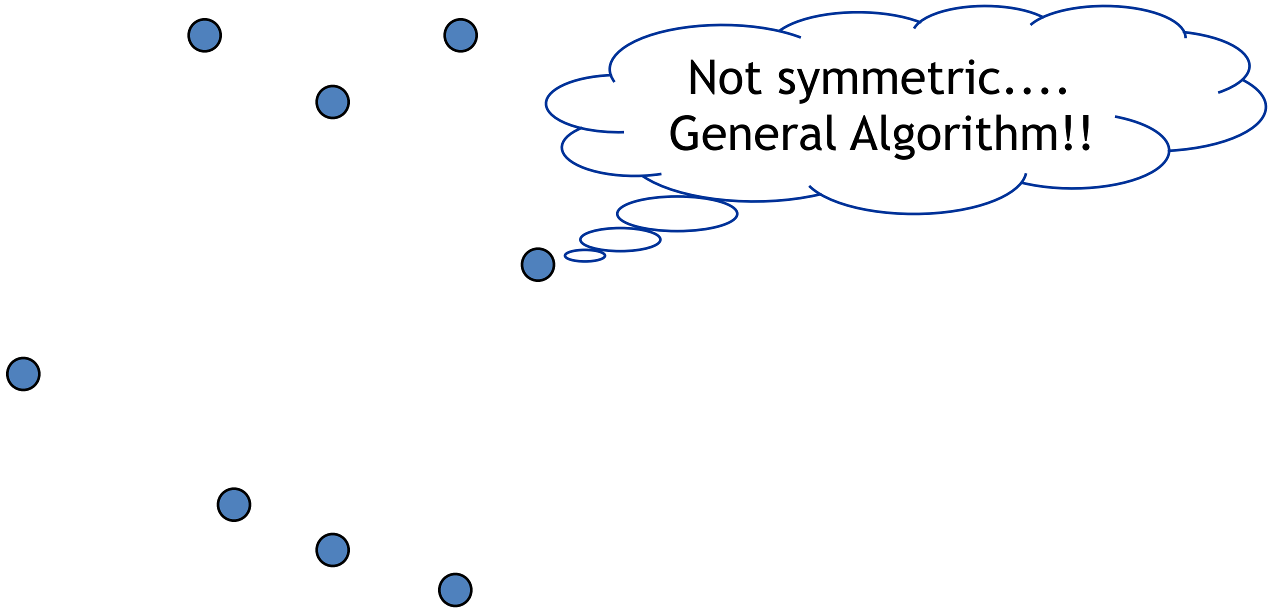
Difficulties

If at the beginning....



Difficulties

If at the beginning....

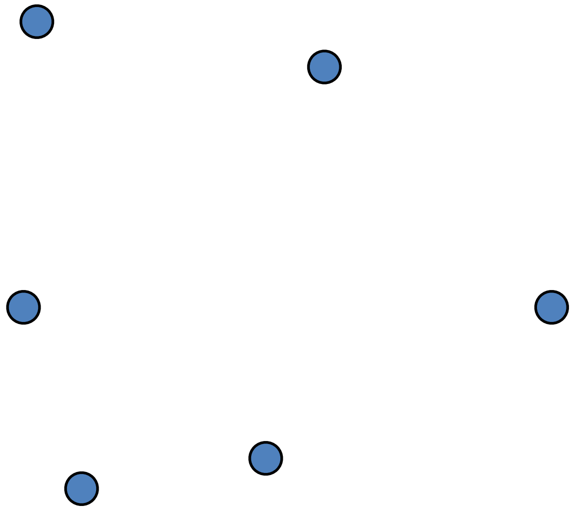


Difficulties

Difficulties

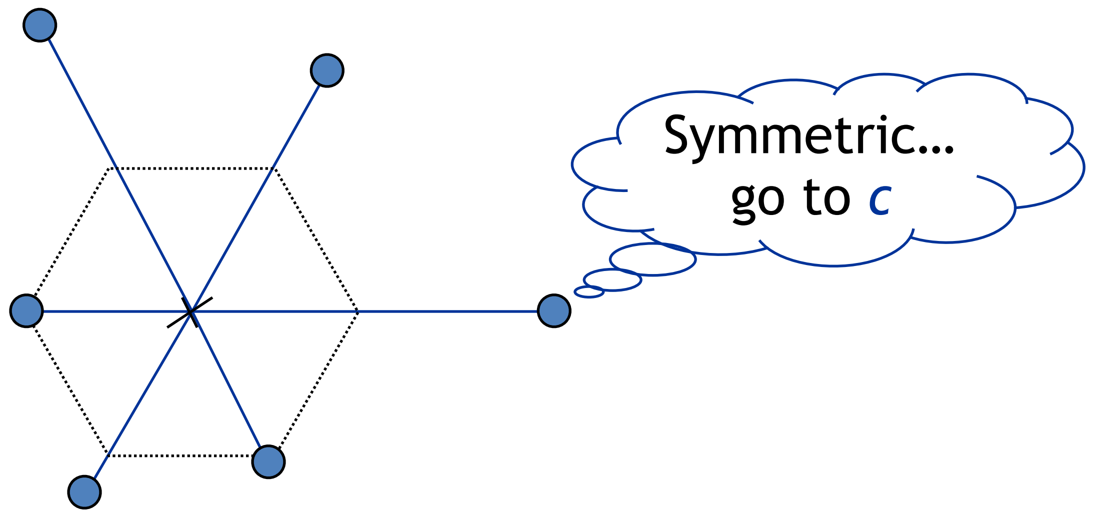
....after a while....

Difficulties



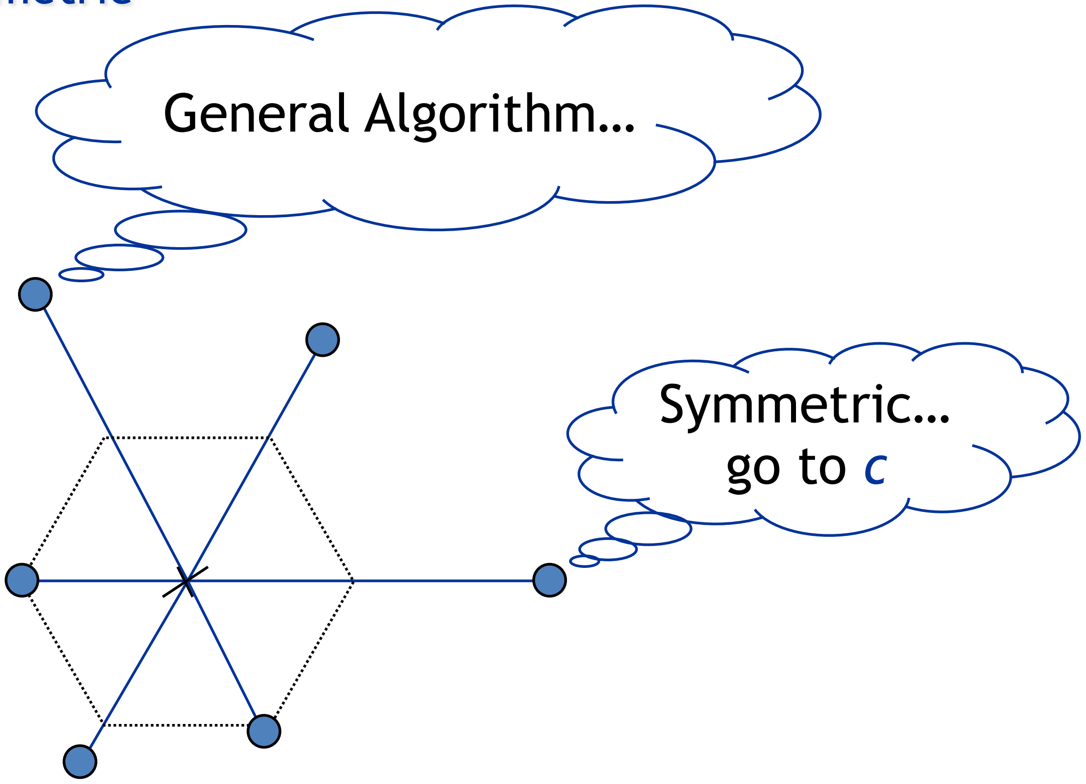
Difficulties

Symmetric



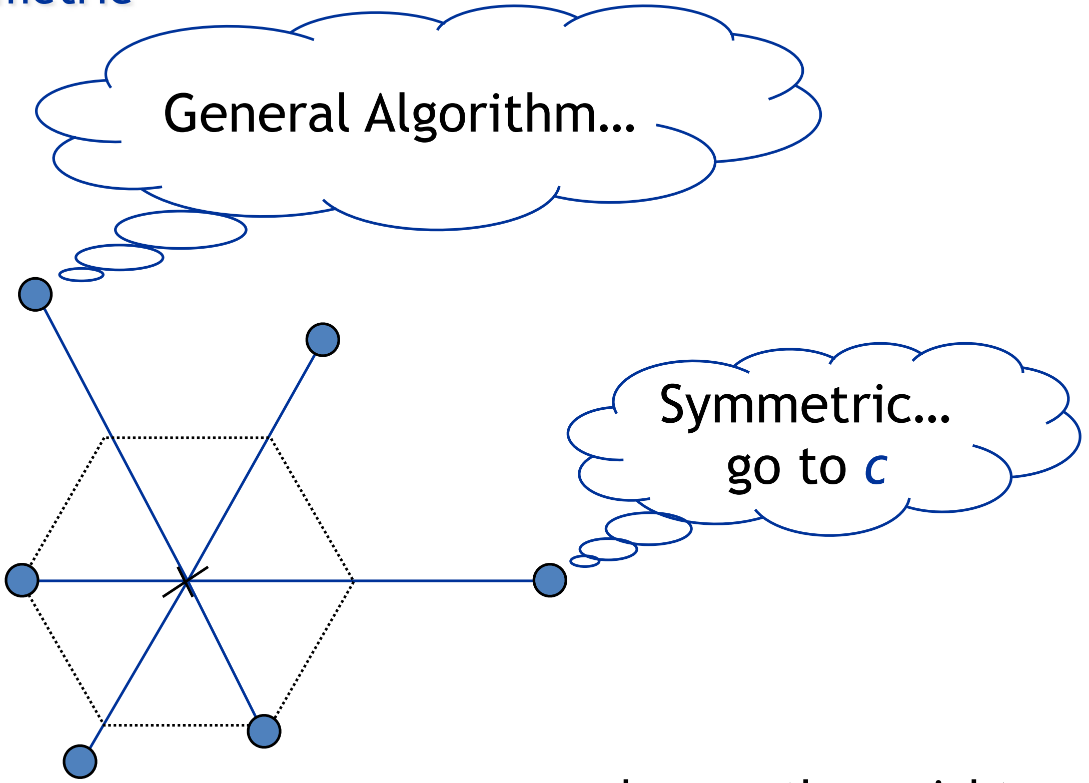
Difficulties

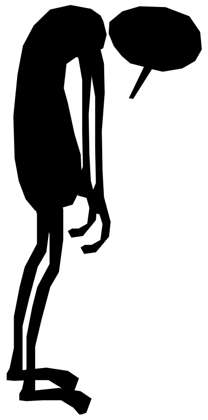
Symmetric



Difficulties

Symmetric





....hence, they might never gather!!!!

Gathering—easy solution

Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, ‘36)!

Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, ‘36)!

Given $r_1,\ldots,r_n$:

Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, '36)!

Given $r_1,\ldots,r_n$:

$$WP = \arg\min_{p \in \Re^2} \sum_i \operatorname{dist}(p, r_i)$$

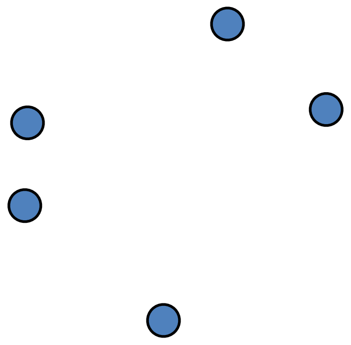
Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, '36)!

Given $r_1,\ldots,r_n$:

$$WP = \arg\min_{p\in\Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, '36)



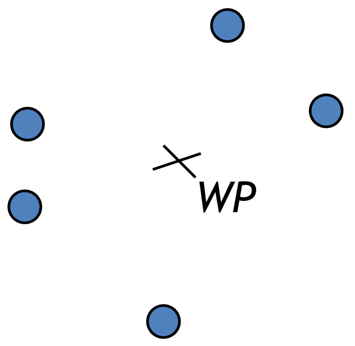
Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, ‘36)!

Given $r_1,\ldots,r_n$:

$$WP = \arg\min_{p\in\Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, ‘36)



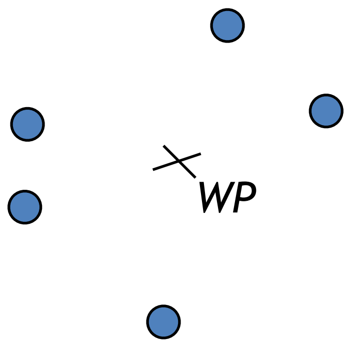
Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, '36)!

Given $r_1, \ldots, r_n$:

$$WP = \arg\min_{p \in \Re^2} \sum_i \text{dist}(p, r_i)$$

1. It is unique (Weiszfeld, '36)



Gathering—easy solution

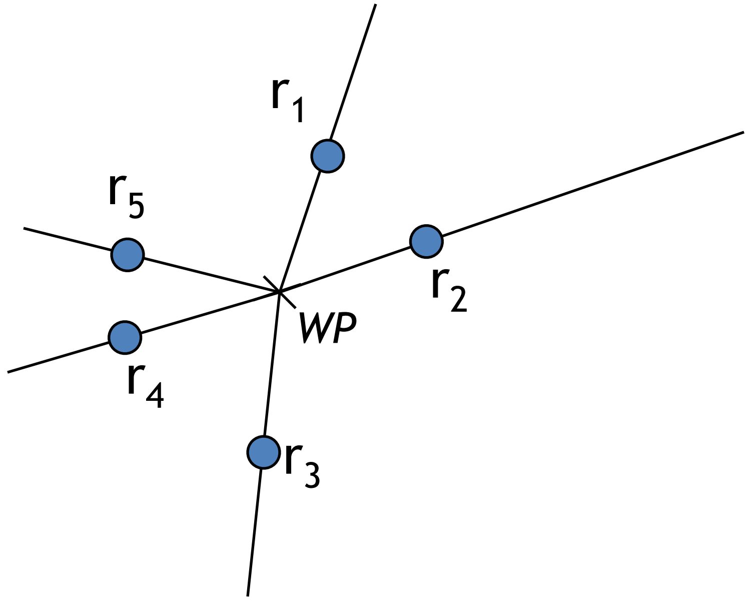
Easy Solution: Weber Point (Weiszfeld, ‘36)!

Given $r_1,\ldots,r_n$:

$$WP = \arg\min_{p\in\Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, ‘36)

2. WP is Weber Point of points on $[r_i, WP]$ (Weiszfeld, ‘36)



Gathering—easy solution

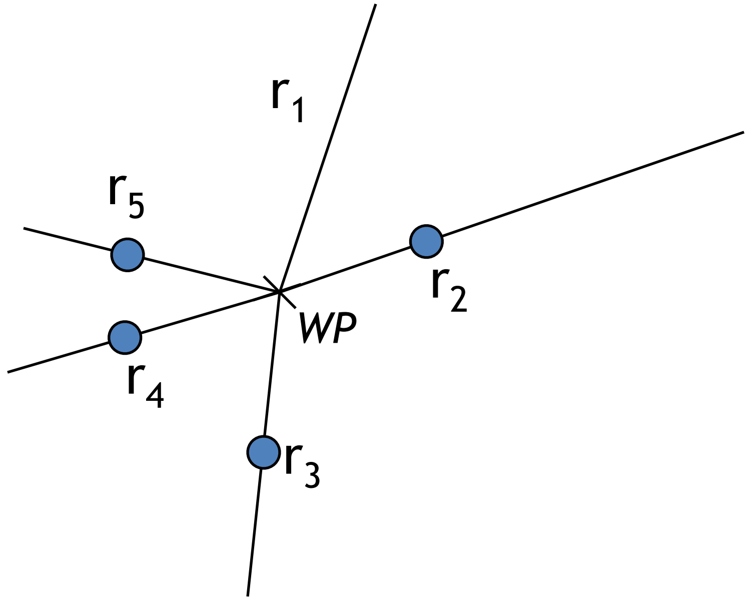
Easy Solution: Weber Point (Weiszfeld, '36)!

Given $r_1,\ldots,r_n$:

$$WP = \arg\min_{p \in \Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, '36)

2. WP is Weber Point of points on $[r_i, WP]$ (Weiszfeld, '36)



Gathering—easy solution

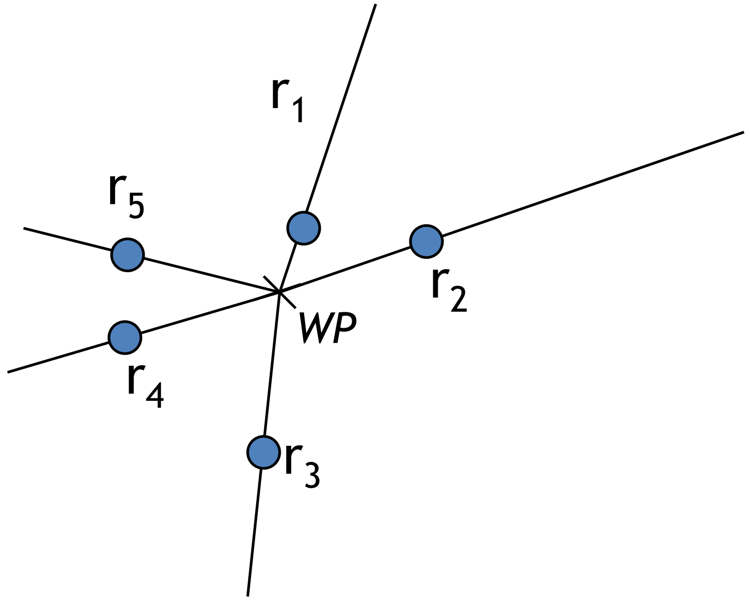
Easy Solution: Weber Point (Weiszfeld, '36)!

Given $r_1, \ldots, r_n$:

$$WP = \arg\min_{p \in \Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, '36)

2. WP is Weber Point of points on $[r_i, WP]$ (Weiszfeld, '36)



Gathering—easy solution

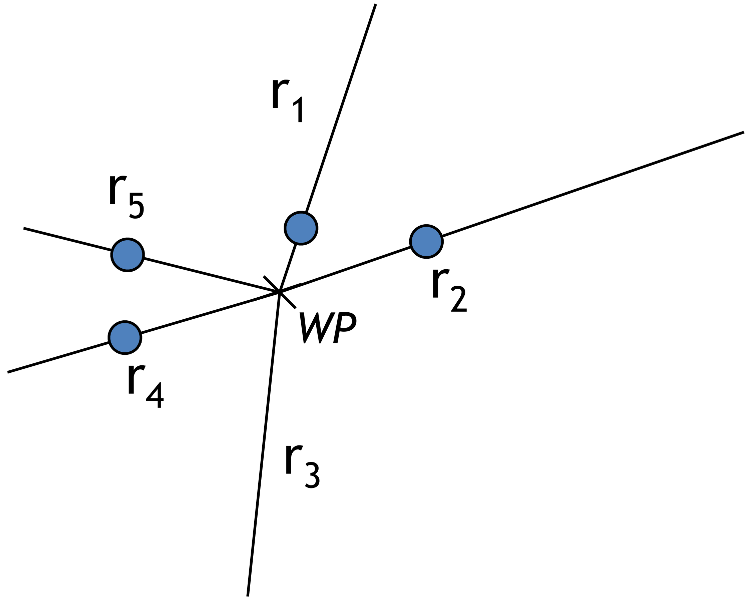
Easy Solution: Weber Point (Weiszfeld, ‘36)!

Given $r_1,\ldots,r_n$:

$$WP = \arg\min_{p\in\Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, ‘36)

2. WP is Weber Point of points on $[r_i, WP]$ (Weiszfeld, ‘36)



Gathering—easy solution

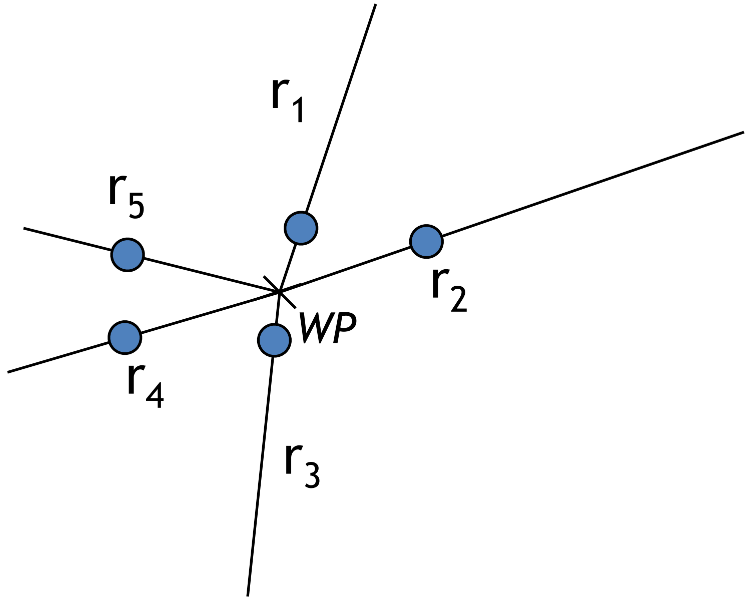
Easy Solution: Weber Point (Weiszfeld, '36)!

Given $r_1,\ldots,r_n$:

$$WP = \arg\min_{p\in\Re^2} \sum_i \text{dist}(p, r_i)$$

1. It is unique (Weiszfeld, '36)

2. WP is Weber Point of points on $[r_i, WP]$ (Weiszfeld, '36)



Gathering—easy solution

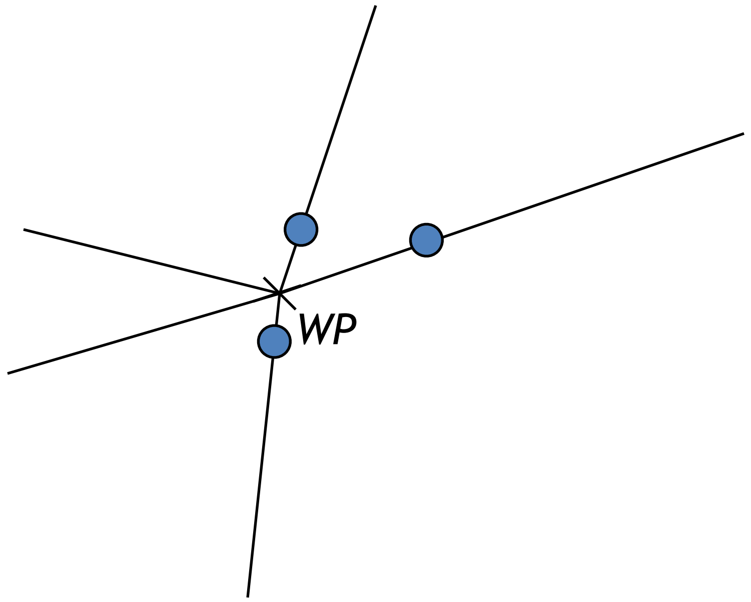
Easy Solution: Weber Point (Weiszfeld, ‘36)!

Given $r_1, \ldots, r_n$:

$$WP = \arg\min_{p \in \Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, ‘36)

2. WP is Weber Point of points on $[r_i, WP]$ (Weiszfeld, ‘36)



Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, '36)!

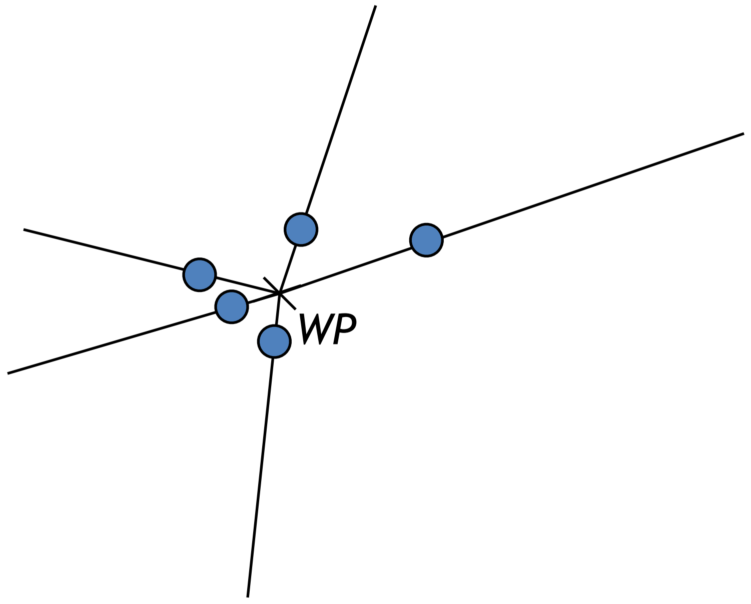
Given $r_1,\dots,r_n$:

$$WP = \arg\min_{p\in\Re^2} \sum_i \mathrm{dist}(p, r_i)$$

1. It is unique (Weiszfeld, '36)
2. WP is Weber Point of points on $[r_i, WP]$ (Weiszfeld, '36)

$\Downarrow$

WP Invariant Under Movement!



Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, ‘36)!

Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, '36)!

Algorithm:

1. Compute WP
2. Move Towards WP

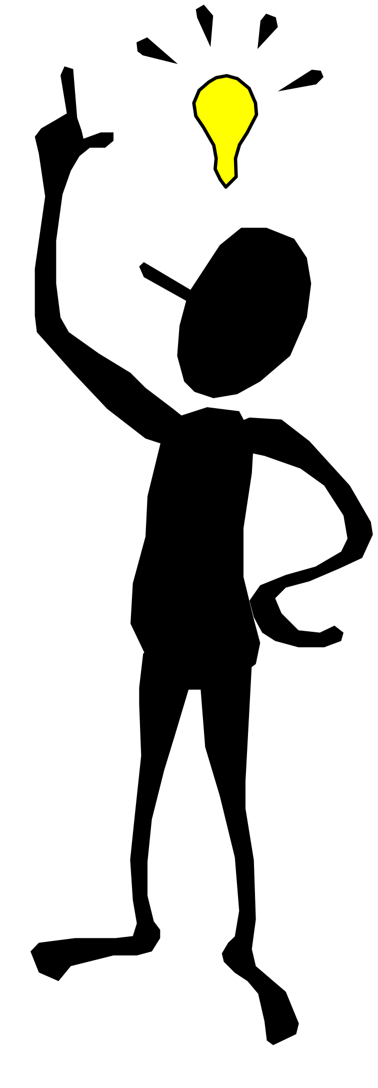
Gathering—easy solution

Easy Solution: Weber Point (Weiszfeld, ‘36)!

Algorithm:

1. Compute WP
2. Move Towards WP

Unfortunately, WP is not computable!



Gathering

(Unlimited Visibility, no agreement)

n=2: Unsolvable (by Suzuki *et al.*),
unless they can *bump into each other*!

Gathering
(Unlimited Visibility, no agreement)

n=2: **Unsolvable (by Suzuki *et al.*)**,
unless they can *bump into each other*!

Gathering
(Unlimited Visibility, no agreement)

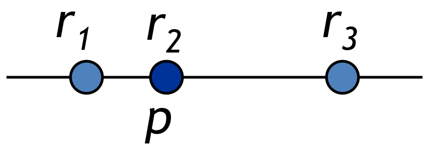
n=2: Unsolvable (by Suzuki *et al.*),
unless they can *bump into each other*!

N=3 (4): Always solvable!

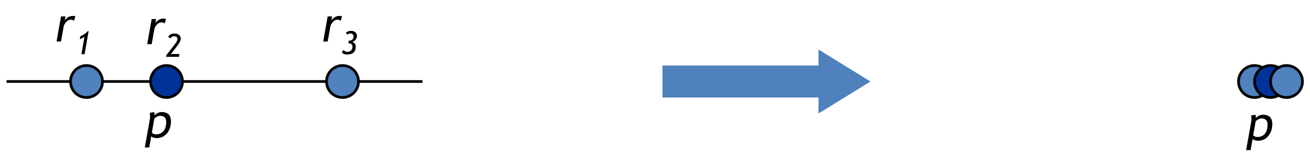
Gathering, *n=3*
(Unlimited Visibility, no agreement)

Gathering, *n=3*

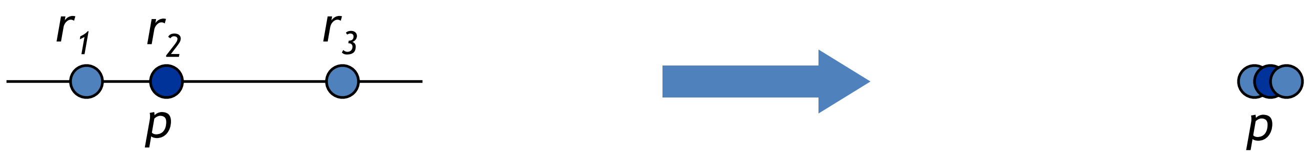
(Unlimited Visibility, no agreement)

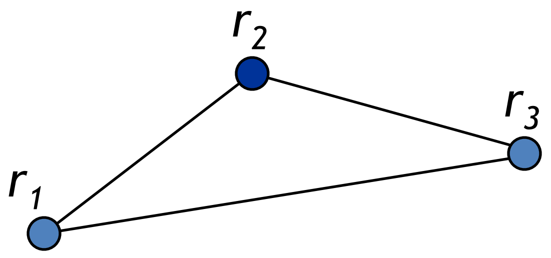


Gathering, *n=3*
(Unlimited Visibility, no agreement)

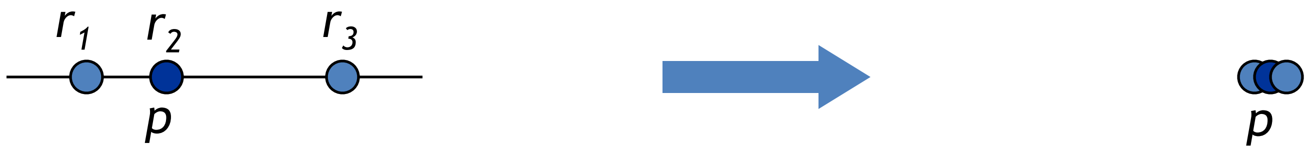


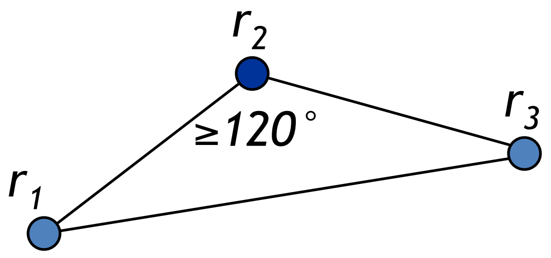
Gathering, *n=3*
(Unlimited Visibility, no agreement)



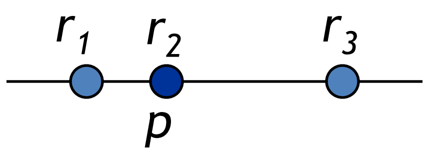


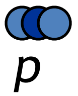
Gathering, *n*=3
(Unlimited Visibility, no agreement)

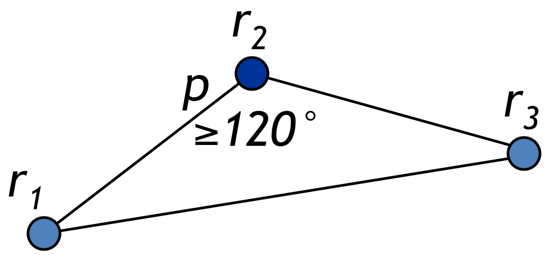




Gathering, *n*=3
(Unlimited Visibility, no agreement)

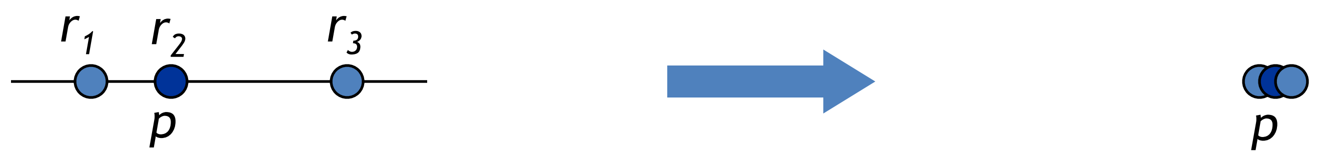


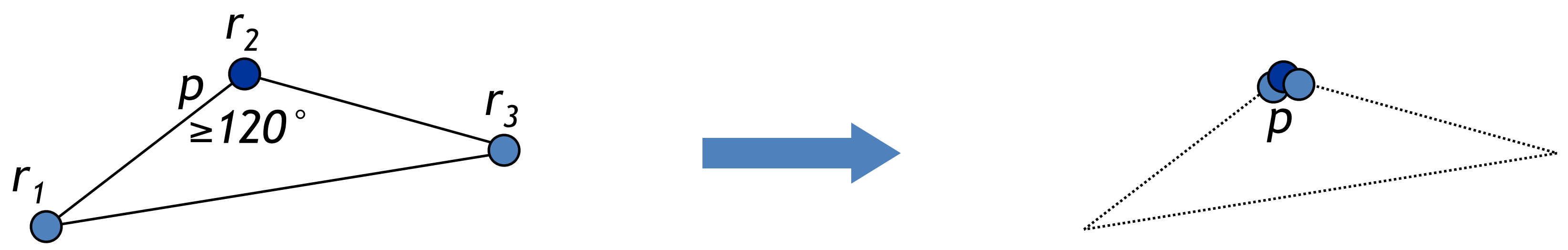




Gathering, *n=3*

(Unlimited Visibility, no agreement)

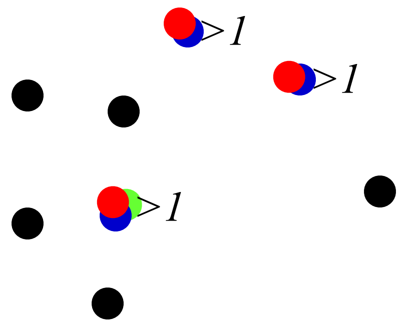




General Schema

- Use of Multiplicity Detection
  - n=3,4 (and even with the use of Weber Point)

Is there 1 or more than 1 robot at a point?



General Schema

The general idea of the solutions is based on multiplicity detection, as follows

General Schema

The general idea of the solutions is based on multiplicity detection, as follows

1. At the beginning, robots on distinct positions

General Schema

The general idea of the solutions is based on multiplicity detection, as follows

1. At the beginning, robots on distinct positions

2. Get a scenario where there is only one point *p* with multiplicity greater than one

General Schema

The general idea of the solutions is based on multiplicity detection, as follows

1. At the beginning, robots on distinct positions
2. Get a scenario where there is only one point *p* with multiplicity greater than one
3. All robots move towards *p*

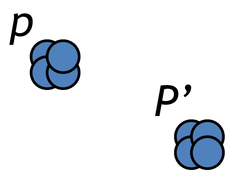
Multiplicity Detection

Multiplicity Detection

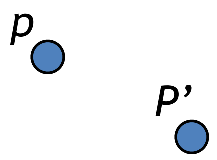
If the robots cannot detect multiplicities…

Multiplicity Detection

If the robots cannot detect multiplicities…



is as…



…the proposed solutions do not work!!!!

Multiplicity Detection

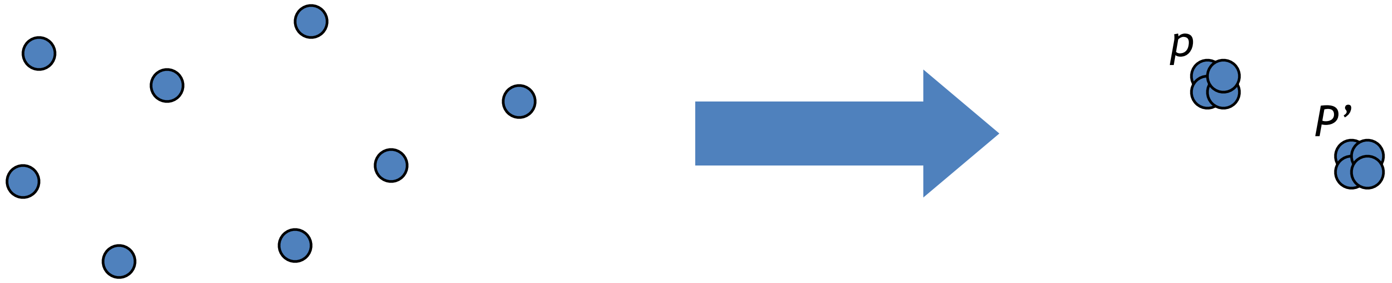
For n=2, the problem is not solvable (*Suzuki et al.*, *1999*)!

It is possible to design an adversary that lets the robots occupy two distinct positions on the plane in a finite number of cycles…

Multiplicity Detection

For n=2, the problem is not solvable (*Suzuki et al., 1999*)!

It is possible to design an adversary that lets the robots occupy two distinct positions on the plane in a finite number of cycles…



Multiplicity Detection

...hence...

Multiplicity Detection

…hence…

Problem not solvable with n=2

$+$

No multiplicity detection

$\Downarrow$

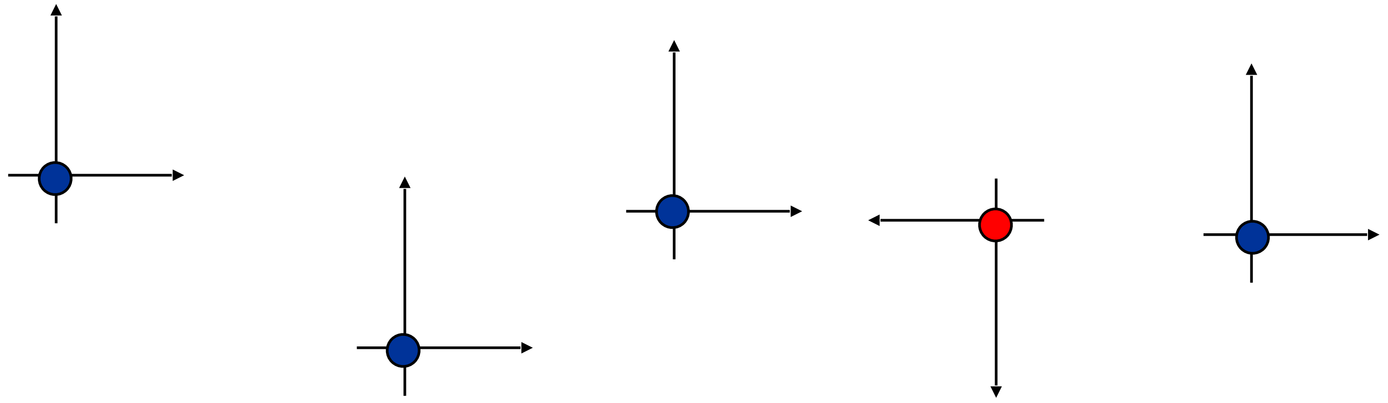
Problem not solvable for any n!

Let's design the adversary

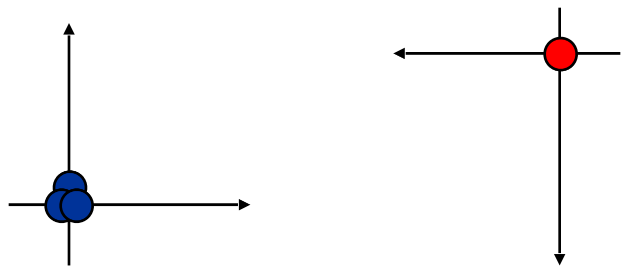
- We divide the robots in two sets
  - n-1 robots (r_1, ..., r_{n-1}) are the blue robots
  - There is one red robot (r_n) (same direction but opposite orientation)

Let’s design the adversary

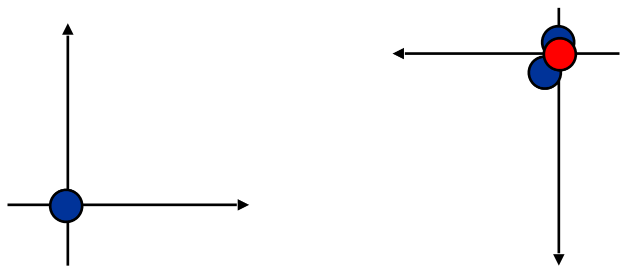
- We divide the robots in two sets
  - n-1 robots ($r_1, \ldots, r_{n-1}$) are the blue robots
  - There is one red robot (r_n) (same direction but opposite orientation)



E-configurations



E1-configuration



E2-configuration

E-configurations

<u>**LEMMA:**</u> If the robots are not gathered on the same position at the beginning, in finite time they reach an E-configuration

E-configurations

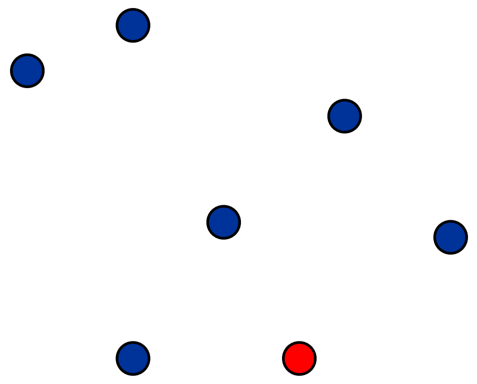
LEMMA: If the robots are not gathered on the same position at the beginning, in finite time they reach an E-configuration

We need to define a scheduler that, given any algorithm A that solves the problem, brings the robots in a E-configuration

E-configurations

Schedule:

If activating all robots at t, they do not gather on p at t+1, activate all of them at t

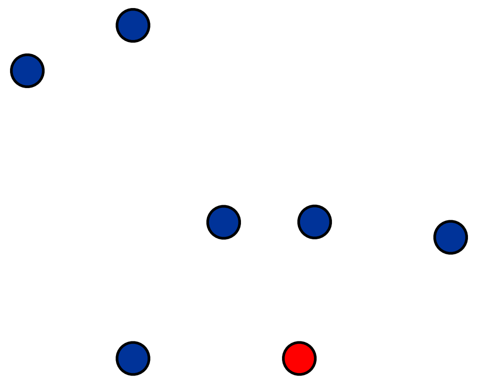


Time: 0

E-configurations

Schedule:

If activating all robots at t, they do not gather on p at t+1, activate all of them at t

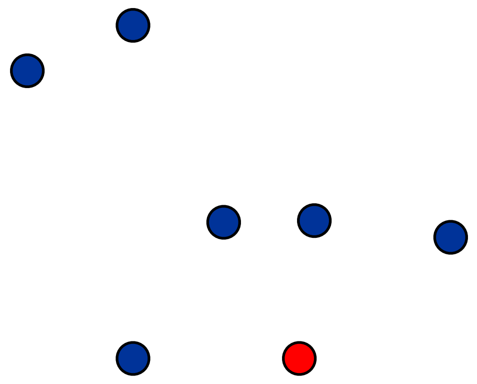


Time: 0

E-configurations

Schedule:

If activating all robots at t, they do not gather on p at t+1, activate all of them at t

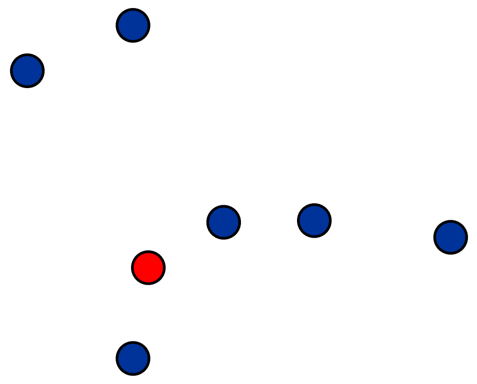


Time: 1

E-configurations

Schedule:

If activating all robots at t, they do not gather on p at t+1, activate all of them at t

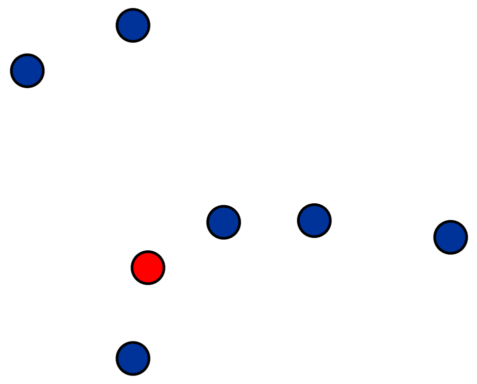


Time: 1

E-configurations

Schedule:

If activating all robots at t, they do not gather on p at t+1, activate all of them at t

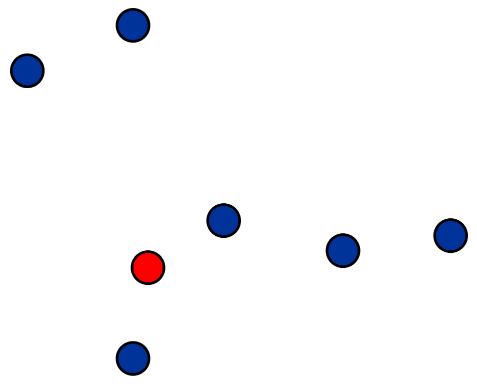


Time: 2

E-configurations

Schedule:

If activating all robots at t, they do not gather on p at t+1, activate all of them at t

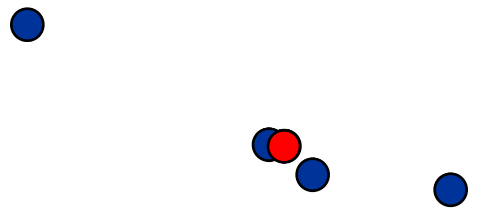


Time: ....

E-configurations

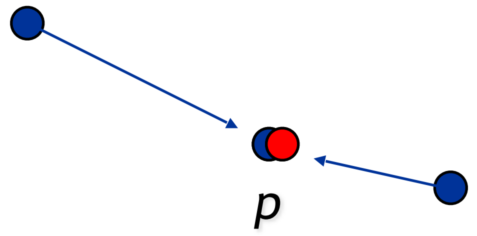
Schedule:

If activating all robots at t, they do not gather on p at t+1, activate all of them at t



Time: ....

E-configurations



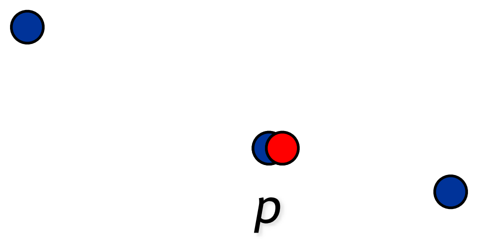
Schedule:

If activating all robots at t, they do not gather on *p* at t+1, activate all of them at t

Otherwise, at t, activate all robots but one that is not on *p* at t

Time: t

E-configurations



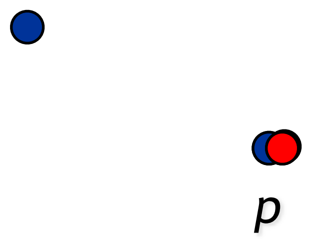
Schedule:

If activating all robots at t, they do not gather on *p* at t+1, activate all of them at t

Otherwise, at t, activate all robots but one that is not on *p* at t

Time: t

E-configurations



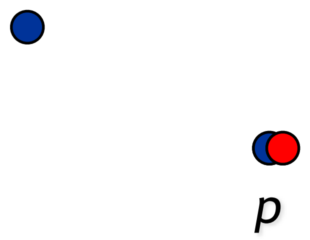
Schedule:

If activating all robots at t, they do not gather on *p* at t+1, activate all of them at t

Otherwise, at t, activate all robots but one that is not on *p* at t

Time: t

E-configurations



Schedule:

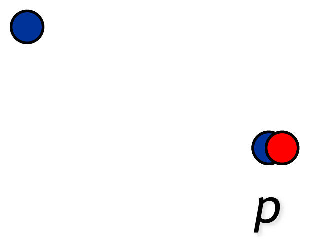
If activating all robots at t, they do not gather on *p* at t+1, activate all of them at t

Otherwise, at t, activate all robots but one that is not on *p* at t

Time: t+1

E-configurations

In the example: E2-conf



Time: t+1

Schedule:

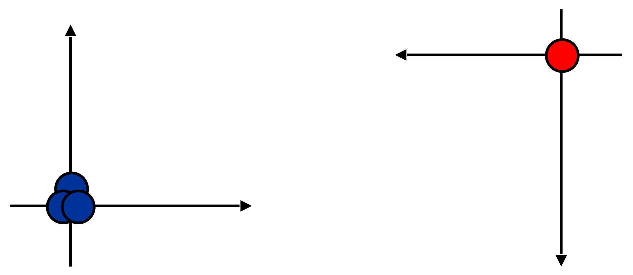
If activating all robots at t, they do not gather on *p* at t+1, activate all of them at t

Otherwise, at t, activate all robots but one that is not on *p* at t

E1-configuration

LEMMA: There exists no deterministic algorithm that, starting from an E1-configuration, solves the gathering problem

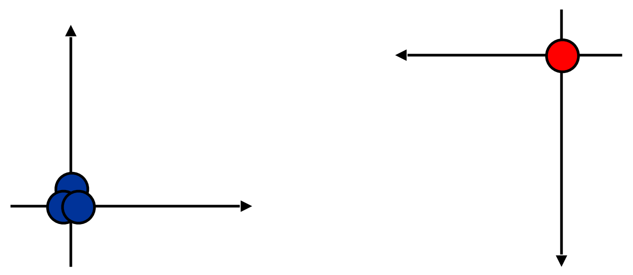
E1-configuration



NOTE: the blue robots have the same *view* of the world and cannot detect multiplicity;

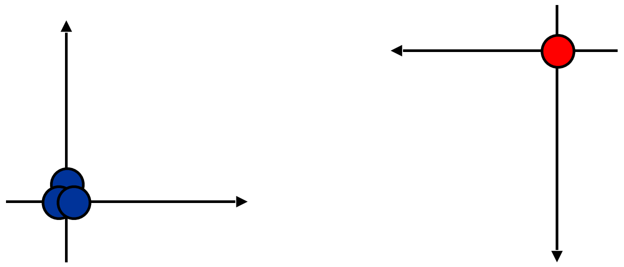
hence, if activated at a time t, they will compute the same destination point, and at time t+1 they will still be on the same position

E1-configuration



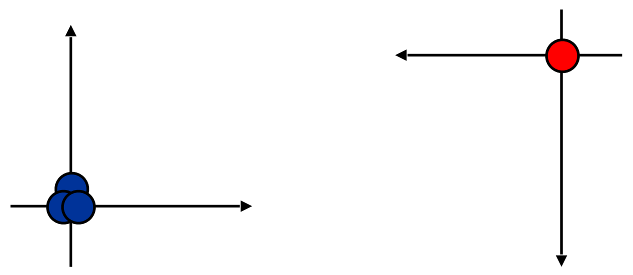
Assume exists A that solves the problem

E1-configuration



E1-configuration

Assume exists A that solves the problem



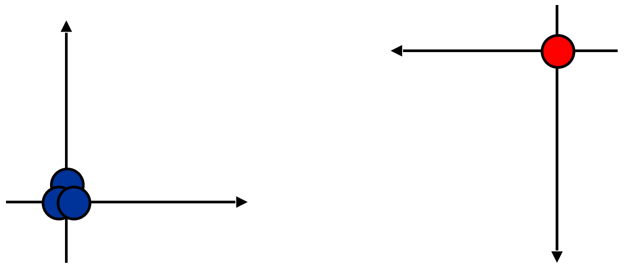
Schedule (alternates the following 2 rules):

If activating one of the blue robots, it does not reach the position occupied by the red one, activate all blue robots

If activating the red robot, it does not reach the position occupied by the blue ones, activate the red robot

Assume exists A that solves the problem

E1-configuration



Schedule (alternates the following 2 rules):

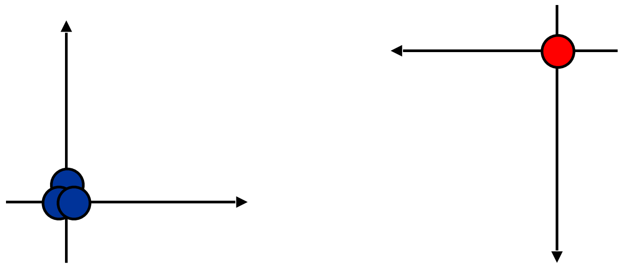
If activating one of the blue robots, it does not reach the position occupied by the red one, activate all blue robots

If activating the red robot, it does not reach the position occupied by the blue ones, activate the red robot

Therefore, for a while the configuration stays E1

Assume exists A that solves the problem

E1-configuration

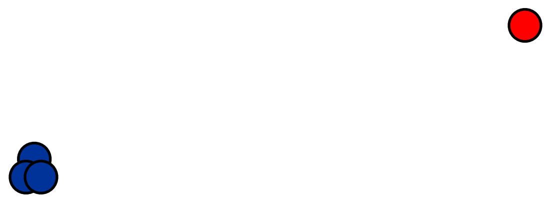


Since A solves the problem, in finite time, either the red goes on the position occupied by blue robots, or viceversa....

....at this time....

E1-configuration

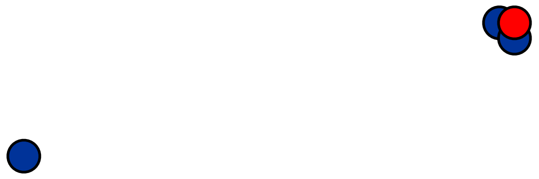
E1-configuration



Schedule:

1. Activate all but one blue robots

E1-configuration



Schedule:

1. Activate all but one blue robots

E1-configuration

Schedule:

1. Activate all but one blue robots
2. Activate the red robot (no multiplicity detection, and A deterministic)

E1-configuration

Schedule:

1. Activate all but one blue robots
2. Activate the red robot (no multiplicity detection, and A deterministic)

E1-configuration

Schedule:

1. Activate all but one blue robots
2. Activate the red robot (no multiplicity detection, and A deterministic)
3. Activate the last blue robot

E1-configuration

Schedule:

1. Activate all but one blue robots
2. Activate the red robot (no multiplicity detection, and A deterministic)
3. Activate the last blue robot

E1-configuration

Schedule:

1. Activate all but one blue robots
2. Activate the red robot (no multiplicity detection, and A deterministic)
3. Activate the last blue robot

Again in a E1 configuration!!!!

E2-configuration

LEMMA: There exists no deterministic algorithm that, starting from an E2-configuration, solves the gathering problem

Finally....

So far, we proved that

- An E-configuration can always be reached in finite time
- No deterministic algorithm solves the Gathering problem starting from an E1 or E2 configuration

Hence:

Theorem: There exists no deterministic algorithm that solves the Gathering problem

Gathering

No agreement on the local coordinate systems,
and *oblivious* robots....

Gathering

No agreement on the local coordinate systems,
and *oblivious* robots....

Necessary Condition: Multiplicity Detection!
(in SYM, hence in Corda)