Gathering robots on meeting-points: feasibility and optimality

Serafino Cicerone[1] Gabriele Di Stefano[1] Alfredo Navarra[2]

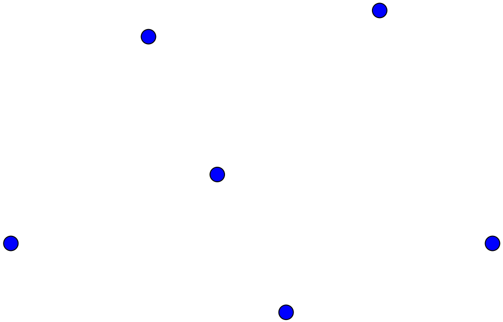
[1]Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica,
Università degli Studi dell'Aquila, Italy.

[2]Dipartimento di Matematica e Informatica,
Università degli Studi di Perugia, Italy.

5th workshop on Moving And Computing (MAC)
7th workshop on GRAph Searching, Theory and Applications (GRASTA)
– October 19-23, 2015 Montreal, Canada –

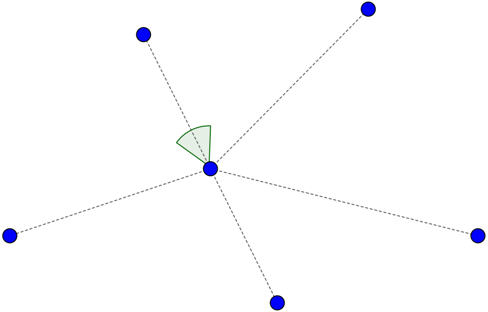
Funded by MIUR PRIN & INδAM - GNCS

Gathering problem



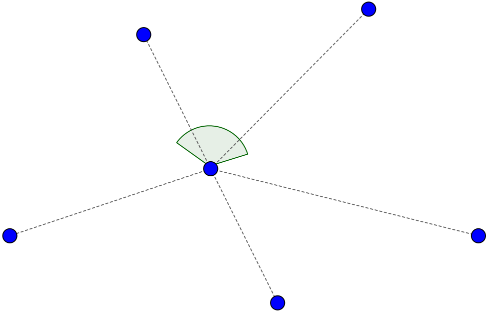
- A configuration of anonymous & autonomous robots on the plane ...
- ... have to agree to meet at some location and remain in there

Gathering problem



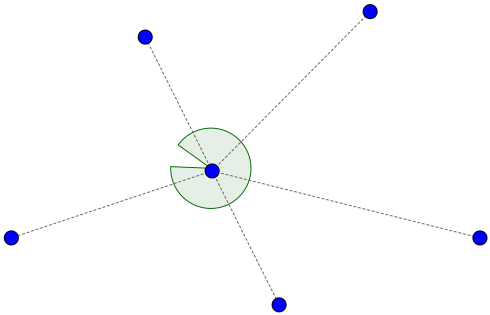
- **sensing** the positions of other robots in its surrounding, ...

Gathering problem



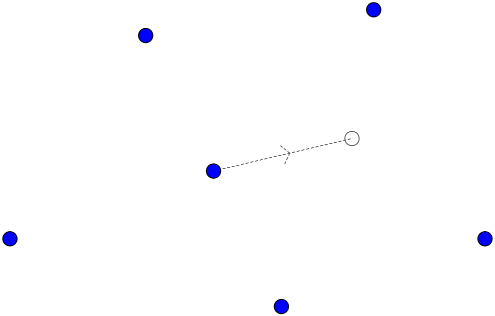
- **sensing** the positions of other robots in its surrounding, ...

Gathering problem



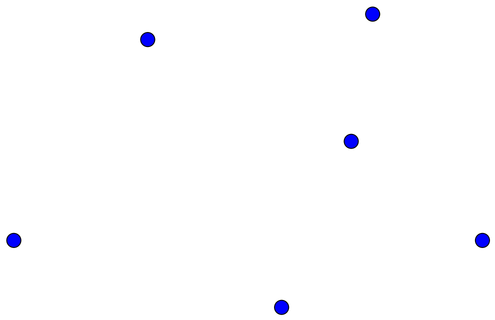
- **sensing** the positions of other robots in its surrounding, ...

Gathering problem



- **computing** a new position, ...

Gathering problem



- **moving** toward it accordingly, ...
- ...thus creating a **new** configuration of robots

Gathering problem

- **AIM**: all robots reach the same place, eventually, and do not move anymore

What is a robot?

Each robot is a **computational unit** that repeatedly cycles through 4 states:

- **Wait**: the robot is idle – a robot cannot stay indefinitely idle – initially, all robots are waiting
- **Look**: the robot observes the world using its sensors which return a configuration (set of points) of the relative positions of all other robots (*configuration view*)
- **Compute**: the robot performs a local computation according to a deterministic algorithm, which is the same for all robots – the result of this phase is a destination point
- **Move**: Non-rigid movement, i.e. there exists δ such that the robot is guaranteed to move of at least of δ unless it wants to move less

What is a robot?

Each robot is a **computational unit** that repeatedly cycles through 4 states:

- **Wait**: the robot is idle – a robot cannot stay indefinitely idle – initially, all robots are waiting
- **Look**: the robot observes the world using its sensors which return a configuration (set of points) of the relative positions of all other robots (*configuration view*)
- **Compute**: the robot performs a local computation according to a deterministic algorithm, which is the same for all robots – the result of this phase is a destination point
- **Move**: Non-rigid movement, i.e. there exists δ such that the robot is guaranteed to move of at least of δ unless it wants to move less

What is a robot?

Each robot is a **computational unit** that repeatedly cycles through 4 states:

- **Wait**: the robot is idle – a robot cannot stay indefinitely idle – initially, all robots are waiting
- **Look**: the robot observes the world using its sensors which return a configuration (set of points) of the relative positions of all other robots (*configuration view*)
- **Compute**: the robot performs a local computation according to a deterministic algorithm, which is the same for all robots – the result of this phase is a destination point
- **Move**: Non-rigid movement, i.e. there exists δ such that the robot is guaranteed to move of at least of δ unless it wants to move less

What is a robot?

Each robot is a **computational unit** that repeatedly cycles through 4 states:

- **Wait**: the robot is idle – a robot cannot stay indefinitely idle – initially, all robots are waiting
- **Look**: the robot observes the world using its sensors which return a configuration (set of points) of the relative positions of all other robots (*configuration view*)
- **Compute**: the robot performs a local computation according to a deterministic algorithm, which is the same for all robots – the result of this phase is a destination point
- **Move**: Non-rigid movement, i.e. there exists δ such that the robot is guaranteed to move of at least of δ unless it wants to move less

What is a robot?

Each robot is a **computational unit** that repeatedly cycles through 4 states:

- **Wait**: the robot is idle – a robot cannot stay indefinitely idle – initially, all robots are waiting
- **Look**: the robot observes the world using its sensors which return a configuration (set of points) of the relative positions of all other robots (*configuration view*)
- **Compute**: the robot performs a local computation according to a deterministic algorithm, which is the same for all robots – the result of this phase is a destination point
- **Move**: Non-rigid movement, i.e. there exists δ such that the robot is guaranteed to move of at least of δ unless it wants to move less

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Robots are

- **Dimensionless** – robots are modeled as geometric points in the plane
- **Anonymous** – no unique identifiers
- **Homogeneous** – all the robots execute the same algorithm
- **Autonomous** – no centralized control
- **Oblivious** – no memory of past events
- **Silent** – no explicit way of communicating – the only mean is to move and let others observe
- **Asynchronous** – there is no global clock ...
  - each phase may have any finite duration, and different robots executions are completely independent
  - fair scheduling: every robot wakes up within finite time, infinitely often
- **Unoriented** – robots do not share a common coordinate system
  - no common compass
  - no common knowledge

Gathering problem: known results

Impossibility of gathering [P'07]

In the asynchronous setting, there exists no deterministic algorithm that solves the gathering problem in finite time, for a set of $n \geq 2$ oblivious robots.

To solve it, we need to add some additional capabilities to robots...

Recent positive result [CFPS'12]

In the asynchronous setting, there exists a deterministic algorithm that solves the gathering problem in finite time, for a set of $n > 2$ oblivious robots with **multiplicity detection**.

[P'07] Prencipe: *Impossibility of gathering by a set of autonomous mobile robots*, Theoret. Comput. Sci., 384 (2007)

[CFPS'12] Cieliebak, Flocchini, Prencipe, Santoro: *Distributed computing by mobile robots: Gathering*. SIAM J. on Comp., 41 (2012)

Gathering problem: known results

Impossibility of gathering [P'07]

In the asynchronous setting, there exists no deterministic algorithm that solves the gathering problem in finite time, for a set of $n \geq 2$ oblivious robots.

To solve it, we need to add some additional capabilities to robots...

Recent positive result [CFPS'12]

In the asynchronous setting, there exists a deterministic algorithm that solves the gathering problem in finite time, for a set of $n > 2$ oblivious robots with **multiplicity detection**.

[P'07] Prencipe: *Impossibility of gathering by a set of autonomous mobile robots*, Theoret. Comput. Sci., 384 (2007)

[CFPS'12] Cieliebak, Flocchini, Prencipe, Santoro: *Distributed computing by mobile robots: Gathering*. SIAM J. on Comp., 41 (2012)

Optimal Gathering problem

- We want to add optimality constraints
- E.g.: minimize the total distance covered by all robots to finalize gathering
- In 2D Euclidean space, it equals to compute the so called **Weber-point**, and move the robots toward it
- **Good news**: there exists one unique Weber-point (unless robots are all collinear)
- **Bad news**: the Weber-point is computationally intractable (already for 5 robots!)

Optimal Gathering problem

- We want to add optimality constraints
- E.g.: minimize the total distance covered by all robots to finalize gathering
- In 2D Euclidean space, it equals to compute the so called **Weber-point**, and move the robots toward it
- **Good news**: there exists one unique Weber-point (unless robots are all collinear)
- **Bad news**: the Weber-point is computationally intractable (already for 5 robots!)

Optimal Gathering problem

- We want to add optimality constraints
- E.g.: minimize the total distance covered by all robots to finalize gathering
- In 2D Euclidean space, it equals to compute the so called **Weber-point**, and move the robots toward it
- **Good news**: there exists one unique Weber-point (unless robots are all collinear)
- **Bad news**: the Weber-point is computationally intractable (already for 5 robots!)

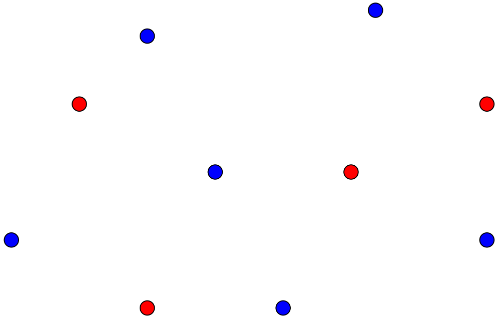
Optimal Gathering problem

- We want to add optimality constraints
- E.g.: minimize the total distance covered by all robots to finalize gathering
- In 2D Euclidean space, it equals to compute the so called **Weber-point**, and move the robots toward it
- **Good news**: there exists one unique Weber-point (unless robots are all collinear)
- **Bad news**: the Weber-point is computationally intractable (already for 5 robots!)

Optimal Gathering problem

- We want to add optimality constraints
- E.g.: minimize the total distance covered by all robots to finalize gathering
- In 2D Euclidean space, it equals to compute the so called **Weber-point**, and move the robots toward it
- **Good news**: there exists one unique Weber-point (unless robots are all collinear)
- **Bad news**: the Weber-point is computationally intractable (already for 5 robots!)

A new challenge...

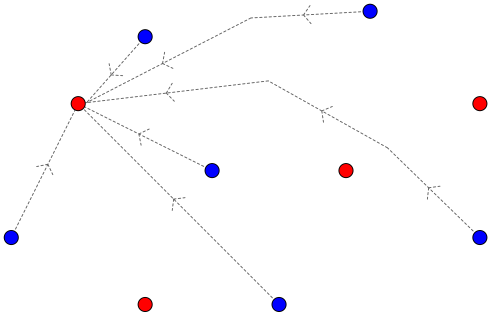


A configuration $\mathcal{C}$ consisting of ...

- a set R of anonymous robots on the plane
- a set M of fixed meeting points

As for the classical gathering, initial configurations are assumed to not contain multiplicities

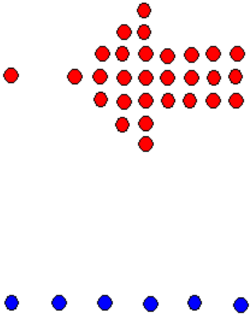
A new challenge...



Problem: GATHERING OVER MEETING POINTS (GMP)

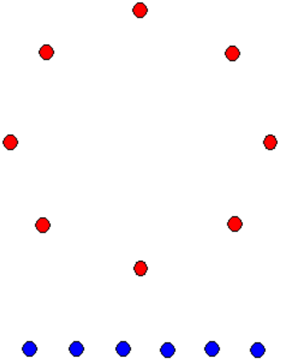
- design an algorithm able to gather all robots on a meeting point in M (**Algosensors'15**)

Dealing with Meeting points



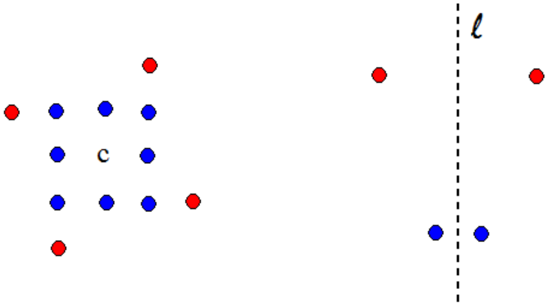
- Meeting points can sometimes help in designing a gathering algorithm...
- ...while sometimes they can play for the adversary.

Dealing with Meeting points



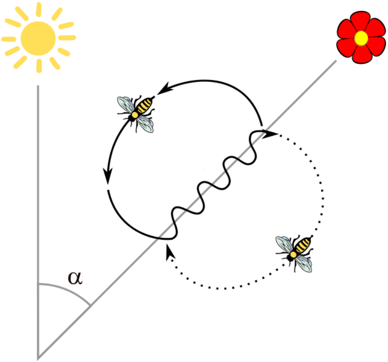
- Meeting points can sometimes help in designing a gathering algorithm...
- ...while sometimes they can play for the adversary.

Ungatherable configurations

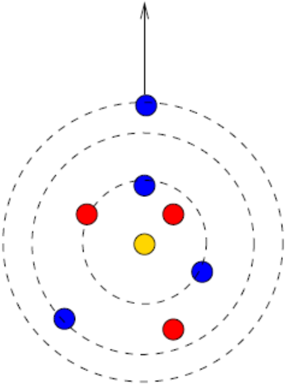


1. $\mathcal{C}$ admits a rotation with center c, and there are neither robots nor meeting-points on c;
2. $\mathcal{C}$ admits one axis of symmetry ℓ, and there are neither robots nor meeting-points on ℓ.

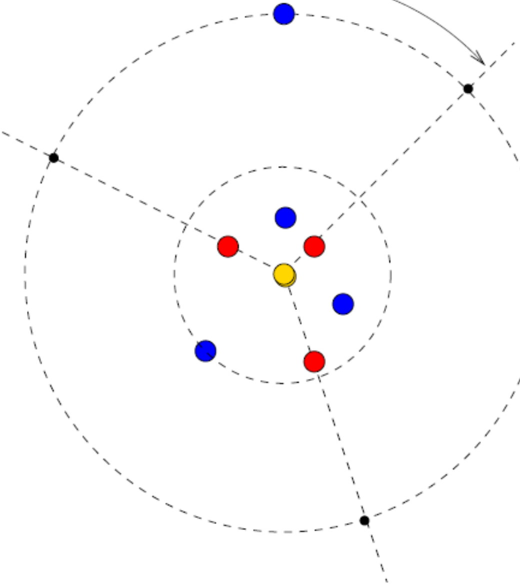
Gathering on meeting-points: stigmergy



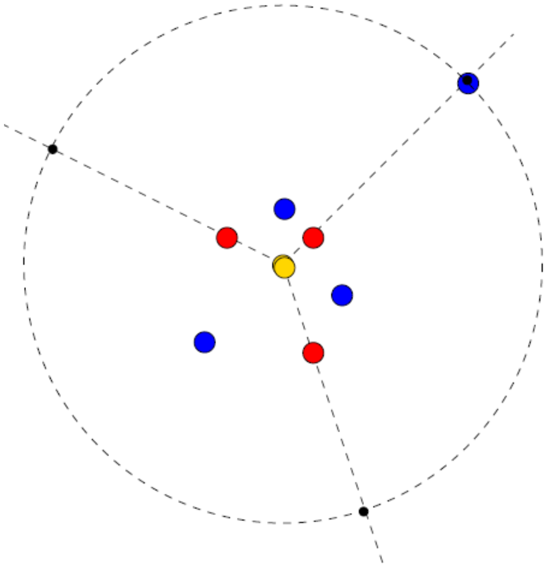
Gathering on meeting-points: stigmergy



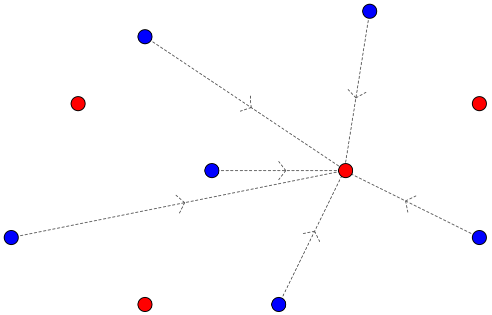
Gathering on meeting-points: stigmergy



Gathering on meeting-points: stigmergy



Optimization versions



Problem GMP is addressed with an additional optimality constraint:

1. the robots must cover the minimum total travel distance to finalize the gathering (**Algosensors'14**)
2. the maximum distance traveled by a single robot must be minimized (**CIAC'15**)

Optimal gathering for case 1

Let $\mathcal{C} = (R, M)$ be a configuration,

$m \in M$ is a **Weber-point** of $\mathcal{C}$ if m minimizes $\sum_{r \in R} d(r, m)$

Let Δ be the above quantity, Δ **is a lower bound for each gathering algorithm**

Definition

- A gathering algorithm is **optimal** if it achieves the gathering with a total distance equal to Δ.

Optimal gathering for case 1

Let $\mathcal{C} = (R, M)$ be a configuration,

$m \in M$ is a **Weber-point** of $\mathcal{C}$ if m minimizes $\sum_{r \in R} d(r, m)$

Let Δ be the above quantity, Δ **is a lower bound for each gathering algorithm**

Definition

- A gathering algorithm is **optimal** if it achieves the gathering with a total distance equal to Δ.

Optimal gathering for case 1

Let $\mathcal{C} = (R, M)$ be a configuration,

$m \in M$ is a **Weber-point** of $\mathcal{C}$ if m minimizes $\sum_{r \in R} d(r, m)$

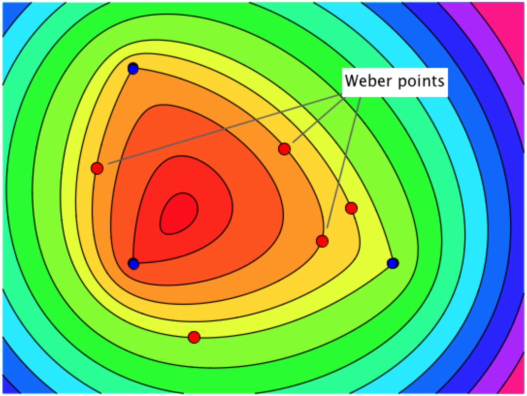
Let Δ be the above quantity. Δ **is a lower bound for each gathering algorit**

Definit

- A … h a to

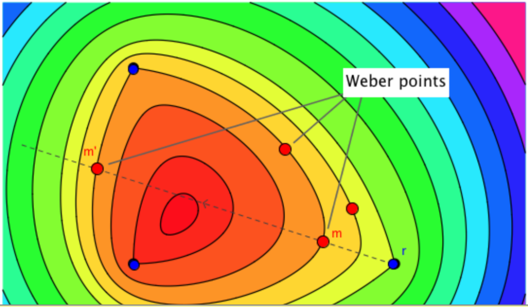
- **What we achieved**:
  1. characterize all the configurations for which an optimal algorithm exists, and ...
  2. ... define the optimal algorithm

Robots are the foci of a *k*-ellipse



- 3-ellipses with different radii
- *k*-ellipses are strictly convex curves

Robots are the foci of a k-ellipse



- **Lemma**:
  Let $\mathcal{C} = (R, M)$ be a configuration, robots in R are not collinear, $r \in R$ moves toward a Weber-point m and this move creates $\mathcal{C}' = (R', M)$. Then:
  - $\mathcal{C}'$ **contains one or two Weber-points only**: m and m' (if any) lies on $hline(r, m)$

- 3-
- k-

The strategy of the algorithm

- Select and move robots **straightly toward a Weber-point** m, so that
- ... after a certain number of moves,
- ... m remains the only Weber-point.
- Once only the Weber-point m exists, all robots move toward it!

This approach provides an optimal algorithm for some special cases:

- $\mathcal{S}_1$: conf's s.t. there is exactly one multiplicity on a meeting-point
- $\mathcal{S}_2$: conf's s.t. there is exactly one Weber-point
- $\mathcal{S}_3$: conf's s.t. a Weber-point m lies on $cg(M)$, the center of gravity of M

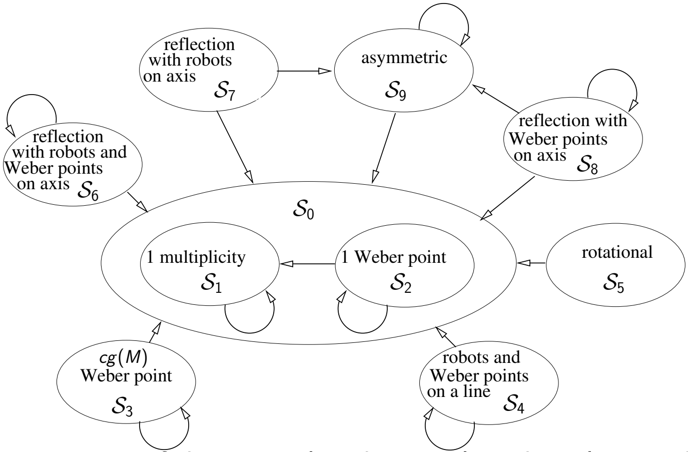
The strategy of the algorithm

- Select and move robots **straightly toward a Weber-point** m, so that
- ... after a certain number of moves,
- ... m remains the only Weber-point.
- Once only the Weber-point m exists, all robots move toward it!

This approach provides an optimal algorithm for some special cases:

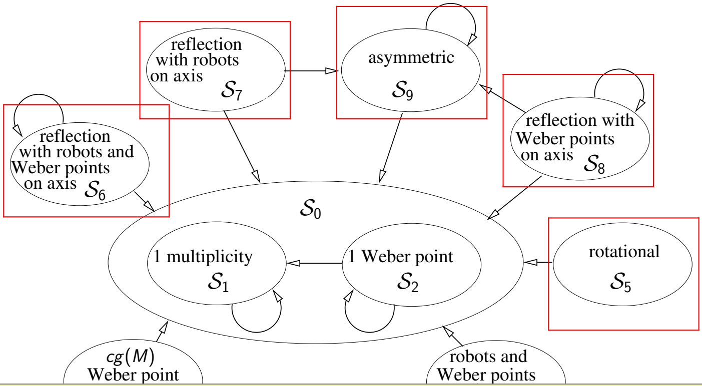
- $\mathcal{S}_1$: conf's s.t. there is exactly one multiplicity on a meeting-point
- $\mathcal{S}_2$: conf's s.t. there is exactly one Weber-point
- $\mathcal{S}_3$: conf's s.t. a Weber-point m lies on $cg(M)$, the center of gravity of M

Partitioning all the configurations



- Schematization of the optimal gathering algorithm along with priorities
- The general algorithm is divided into sub-procedures:
  - each sub-procedure is specific for configurations of a given class $\mathcal{S}_i$

Partitioning all the configurations



Several classes concern **isometries**:

- asymmetric configurations
- configurations with reflection
- configurations with rotation

- Sc
  pr
- Th
  - each sub-procedure is specific for configurations of a given class $\mathcal{S}_i$

Optimal gathering for problem 2

Let $\mathcal{C} = (R, M)$ be a configuration,

$m \in M$ is a **minmax-point** of $\mathcal{C}$ if m minimizes $\max_{r \in R} d(r, m)$

Let Δ be the above quantity, Δ **is a lower bound for each gathering algorithm**

Definition

- A gathering algorithm is **optimal** if it achieves the gathering by letting move each robot of at most $\Delta(\mathcal{C})$.

Optimal gathering for problem 2

Let $\mathcal{C} = (R, M)$ be a configuration,

$m \in M$ is a **minmax-point** of $\mathcal{C}$ if m minimizes $\max_{r \in R} d(r, m)$

Let Δ be the above quantity, **Δ is a lower bound for each gathering algorithm**

Definition

- A gathering algorithm is **optimal** if it achieves the gathering by letting move each robot of at most $\Delta(\mathcal{C})$.

Optimal gathering for problem 2

Let $\mathcal{C} = (R, M)$ be a configuration,

$m \in M$ is a **minmax-point** of $\mathcal{C}$ if m minimizes $\max_{r \in R} d(r, m)$

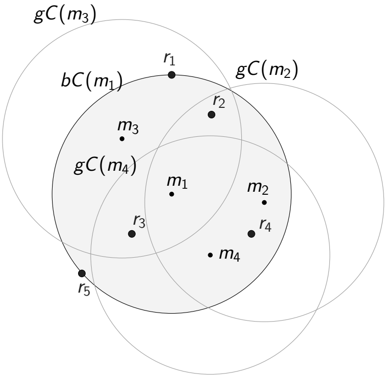
Let Δ be the above quantity. Δ **is a lower bound for each gathering algorit**

Definit

- A
  let

- **What we achieved**:
  1. characterize the configurations for which an optimal algorithm exists, and ...
  2. ... define the optimal algorithm for almost all configurations where it is possible to obtain it

Some Notation



Black-Circle (bC): circle of radius $\Delta(\mathcal{C})$ centered on a meeting-point m containing all robots, hence m is a minmax-point

- **Border-robots**: wrt $bC(m_1)$: r_1, r_5
- **Internal-robots** wrt $bC(m_1)$: r_2, r_3, r_4

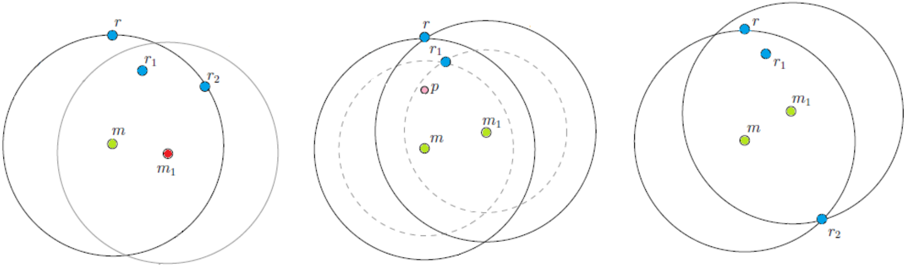
Grey-circle (gC): circle of radius $\Delta(\mathcal{C})$ centered on a meeting-point, not containing all robots

The strategy of the algorithm

- Select and move robots **straightly toward a Minmax-point** m, so that
- ...after a certain number of moves,
- ...m remains the only Minmax-point.
- Once only the Minmax-point m exists, **still robots require special strategies to "safely" move toward it!**

The strategy of the algorithm

- Select and move robots **straightly toward a Minmax-point** m, so that
- ...after a certain number of moves,
- ...m remains the only Minmax-point.
- Once only the Minmax-point m exists, **still robots require special strategies to "safely" move toward it!**



Characterizing configurations

- $mm_B(\mathcal{C}) \subseteq mm(\mathcal{C})$ is the set of minmax-points with minimum number of border-robots;
- $mm_W(\mathcal{C}) \subseteq mm_B(\mathcal{C})$ is the set of minmax-points in $mm_B(\mathcal{C})$ with minimal Weber distance;
- $mm_V(\mathcal{C}) \in mm_W(\mathcal{C})$ is the minmax-point in $mm_W(\mathcal{C})$ with minimal view.

$\mathcal{S}_1$: any configuration $\mathcal{C}$ such that $|mm(\mathcal{C})| = 1$;

$\mathcal{S}_2$: any configuration $\mathcal{C} \notin \mathcal{S}_1$ such that $|mm_V(\mathcal{C})| = 1$;

$\mathcal{S}_3$: initial configurations $\mathcal{C} \notin \bigcup_{1\leq i\leq 2} \mathcal{S}_i$ such that $\mathcal{C}$ admits a reflection with robots on the axis;

$\mathcal{S}_4$: initial configurations $\mathcal{C} \notin \bigcup_{1\leq i\leq 3} \mathcal{S}_i$ such that $\mathcal{C}$ admits a rotation with a robot as center;

$\mathcal{S}_5$: initial configurations $\mathcal{C} \notin \bigcup_{1\leq i\leq 4} \mathcal{S}_i$ such that $\mathcal{C}$ admits a reflection with minmax-points on the axis;

Characterizing configurations

- $mm_B(\mathcal{C}) \subseteq mm(\mathcal{C})$ is the set of minmax-points with minimum number of border-robots;
- $mm_W(\mathcal{C}) \subseteq mm_B(\mathcal{C})$ is the set of minmax-points in $mm_B(\mathcal{C})$ with minimal Weber distance;
- $mm_V(\mathcal{C}) \in mm_W(\mathcal{C})$ is the minmax-point in $mm_W(\mathcal{C})$ with minimal view.

$\mathcal{S}_1$: any configuration $\mathcal{C}$ such that $|mm(\mathcal{C})| = 1$;

$\mathcal{S}_2$: any configuration $\mathcal{C} \notin \mathcal{S}_1$ such that $|mm_V(\mathcal{C})| = 1$;

$\mathcal{S}_3$: initial configurations $\mathcal{C} \notin \bigcup_{1 \le i \le 2} \mathcal{S}_i$ such that $\mathcal{C}$ admits a reflection with robots on the axis;

$\mathcal{S}_4$: initial configurations $\mathcal{C} \notin \bigcup_{1 \le i \le 3} \mathcal{S}_i$ such that $\mathcal{C}$ admits a rotation with a robot as center;

$\mathcal{S}_5$: initial configurations $\mathcal{C} \notin \bigcup_{1 \le i \le 4} \mathcal{S}_i$ such that $\mathcal{C}$ admits a reflection with minmax-points on the axis;

Strategy for class $\mathcal{S}_1$

1. If $\mathcal{C} \in \mathcal{S}_1$ then $mm(\mathcal{C}) = \{m\}$;
2. All robots can move toward m without entering grey-circles $\rightarrow$ without creating new minmax-points
3. A robot evaluates the closest grey-circle on the direction toward m and moves by halving such a distance
4. Once all robots have moved (a *round* has completed), $\Delta(\mathcal{C})$ decreases
5. Eventually, all robots reach m as for each round we guarantee a minimum constant movement

Strategy for class $\mathcal{S}_2$

1. If $\mathcal{C} \in \mathcal{S}_2$ then $mm_V(\mathcal{C}) = \{m\}$;
2. All *border* robots move toward m without entering grey-circles
3. Once all such robots have moved $mm(\mathcal{C}) = \{m\} \rightarrow$ the configuration becomes of class $\mathcal{S}_1$

About the other classes

1. If $\mathcal{C} \in \mathcal{S}_3$ a robot on the axis is selected and moved in such a way the configuration becomes of class $\mathcal{S}_1$ or $\mathcal{S}_2$
2. If $\mathcal{C} \in \mathcal{S}_4$ the robot placed in the center of the rotation is moved in such a way the configuration becomes of class $\mathcal{S}_1$ or $\mathcal{S}_2$
3. For $\mathcal{C} \in \mathcal{S}_5$ some cases remain open. We prove that in general optimal gathering is impossible but perhaps there is a subclass where an optimal algorithm can be designed

Remark

- No *multiplicity detection* is required by the current strategy.

About the other classes

1. If $\mathcal{C} \in \mathcal{S}_3$ a robot on the axis is selected and moved in such a way the configuration becomes of class $\mathcal{S}_1$ or $\mathcal{S}_2$
2. If $\mathcal{C} \in \mathcal{S}_4$ the robot placed in the center of the rotation is moved in such a way the configuration becomes of class $\mathcal{S}_1$ or $\mathcal{S}_2$
3. For $\mathcal{C} \in \mathcal{S}_5$ some cases remain open. We prove that in general optimal gathering is impossible but perhaps there is a subclass where an optimal algorithm can be designed

Remark

- No *multiplicity detection* is required by the current strategy.

Conclusion

- Extended the classical gathering problem on the plane ...
  - Asynchronous Look-Compute-Move model
  - global weak multiplicity detection capability
- ... by introducing **restrictions on the places where to gather**
- Introduced also **optimality requirements** for the algorithm
  1. minimize the total distance covered by all robots
  2. minimize the maximum distance covered by a robot
- Provided non-gatherability characterizations
- Defined a fully characterizing algorithm for GMP
- Defined an optimal gathering algorithm for case 1 (case 2, resp.) dealing with all (almost all, resp.) possible configurations

Conclusion

- Extended the classical gathering problem on the plane ...
  - Asynchronous Look-Compute-Move model
  - global weak multiplicity detection capability
- ... by introducing **restrictions on the places where to gather**
- Introduced also **optimality requirements** for the algorithm
  1. minimize the total distance covered by all robots
  2. minimize the maximum distance covered by a robot
- Provided non-gatherability characterizations
- Defined a fully characterizing algorithm for GMP
- Defined an optimal gathering algorithm for case 1 (case 2, resp.) dealing with all (almost all, resp.) possible configurations

Conclusion

- Extended the classical gathering problem on the plane ...
  - Asynchronous Look-Compute-Move model
  - global weak multiplicity detection capability
- ... by introducing **restrictions on the places where to gather**
- Introduced also **optimality requirements** for the algorithm
  1. minimize the total distance covered by all robots
  2. minimize the maximum distance covered by a robot
- Provided non-gatherability characterizations
- Defined a fully characterizing algorithm for GMP
- Defined an optimal gathering algorithm for case 1 (case 2, resp.) dealing with all (almost all, resp.) possible configurations

Conclusion

- Extended the classical gathering problem on the plane ...
  - Asynchronous Look-Compute-Move model
  - global weak multiplicity detection capability
- ... by introducing **restrictions on the places where to gather**
- Introduced also **optimality requirements** for the algorithm
  1. minimize the total distance covered by all robots
  2. minimize the maximum distance covered by a robot
- Provided non-gatherability characterizations
- Defined a fully characterizing algorithm for GMP
- Defined an optimal gathering algorithm for case 1 (case 2, resp.) dealing with all (almost all, resp.) possible configurations

Conclusion

- Extended the classical gathering problem on the plane ...
  - Asynchronous Look-Compute-Move model
  - global weak multiplicity detection capability
- ... by introducing **restrictions on the places where to gather**
- Introduced also **optimality requirements** for the algorithm
  1. minimize the total distance covered by all robots
  2. minimize the maximum distance covered by a robot
- Provided non-gatherability characterizations
- Defined a fully characterizing algorithm for GMP
- Defined an optimal gathering algorithm for case 1 (case 2, resp.) dealing with all (almost all, resp.) possible configurations

Conclusion

- Extended the classical gathering problem on the plane ...
  - Asynchronous Look-Compute-Move model
  - global weak multiplicity detection capability
- ... by introducing **restrictions on the places where to gather**
- Introduced also **optimality requirements** for the algorithm
  1. minimize the total distance covered by all robots
  2. minimize the maximum distance covered by a robot
- Provided non-gatherability characterizations
- Defined a fully characterizing algorithm for GMP
- Defined an optimal gathering algorithm for case 1 (case 2, resp.) dealing with all (almost all, resp.) possible configurations

Future Work

- Study GMP without multiplicity detection
- Extend the analysis and the algorithms for the optimization problems to configurations where optimal gathering cannot be assured
- Use different objective functions
- Study GMP (with/without optimization) on graphs
- Study different tasks that may include meeting-points, e.g., pattern formation on specified points as in [FYOKY'15]

Future Work

- Study GMP without multiplicity detection
- Extend the analysis and the algorithms for the optimization problems to configurations where optimal gathering cannot be assured
- Use different objective functions
- Study GMP (with/without optimization) on graphs
- Study different tasks that may include meeting-points, e.g., pattern formation on specified points as in [FYOKY'15]

Future Work

- Study GMP without multiplicity detection
- Extend the analysis and the algorithms for the optimization problems to configurations where optimal gathering cannot be assured
- Use different objective functions
- Study GMP (with/without optimization) on graphs
- Study different tasks that may include meeting-points, e.g., pattern formation on specified points as in [FYOKY'15]

Future Work

- Study GMP without multiplicity detection
- Extend the analysis and the algorithms for the optimization problems to configurations where optimal gathering cannot be assured
- Use different objective functions
- Study GMP (with/without optimization) on graphs
- Study different tasks that may include meeting-points, e.g., pattern formation on specified points as in [FYOKY'15]

Future Work

- Study GMP without multiplicity detection
- Extend the analysis and the algorithms for the optimization problems to configurations where optimal gathering cannot be assured
- Use different objective functions
- Study GMP (with/without optimization) on graphs
- Study different tasks that may include meeting-points, e.g., pattern formation on specified points as in [FYOKY'15]

[FYOKY'15] Fujinaga, Yamauchi, Ono, Kijima, Yamashita: *Pattern Formation by Oblivious Asynchronous Mobile Robots.* SIAM J. on Comp., 44(3) (2015)

Future Work

- Study GMP without multiplicity detection
- Extend the analysis and the algorithms for the optimization problems to configurations where optimal gathering cannot be assured
- Use different objective functions
- Study GMP (with/without optimization) on graphs
- Study different tasks that may include meeting-points, e.g., pattern formation on specified points as in [FYOKY'15]

THANK YOU