Gathering of Robots on Meeting-Points

Serafino Cicerone[1], Gabriele Di Stefano[1], and Alfredo Navarra[2(✉)]

[1] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Via Vetoio, Coppito, 67100 L'Aquila, Italy
{serafino.cicerone, gabriele.distefano}@univaq.it
[2] Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Via Vanvitelli 1, 06123 Perugia, Italy
alfredo.navarra@unipg.it

Abstract. We consider the gathering problem of oblivious and asynchronous robots moving in the plane. When $n > 2$ robots are free to gather anywhere in the plane, the problem has been solved in [Cieliebak et al., *SIAM J. on Comput., 41(4), 2012*]. We propose a new natural and challenging model that requires robots to gather only at some predetermined points in the plane, herein referred to as *meeting-points*.

Robots operate in standard Look-Compute-Move cycles. In one cycle, a robot perceives the robots' positions and the meeting-points (Look) according to its own coordinate system, decides whether to move toward some direction (Compute), and in the positive case it moves (Move). Cycles are performed asynchronously for each robot. Robots are anonymous and execute the same distributed and deterministic algorithm.

In the new proposed model, we fully characterize when gathering can be accomplished. We design an algorithm that solves the problem for all configurations with $n > 0$ robots but those identified as ungatherable.

1 Introduction

The gathering task is a basic primitive in robot-based computing systems. It has been extensively studied in the literature under different assumptions. The problem asks to design a distributed algorithm that allows a team of robots to meet at some common place. Varying on the capabilities of the robots as well as on the environment where they move, very different and challenging aspects must be faced (see, e.g. [2,7,9–11,14,15], and references therein).

In this paper we consider a very minimal setting. We are interested in robots placed in $\mathbb{R}^2$ where they can freely move but they must meet at some predetermined points, herein called *meeting-points*. We call this new problem the *Gathering on Meeting-Points* problem, shortly GMP.

The work has been partially supported by the Italian Ministry of Education, University, and Research (MIUR) under national research projects: PRIN 2010N5K7EB "ARS TechnoMedia – Algoritmica per le Reti Sociali Tecno-Mediate" and PRIN 2012C4E3KT "AMANDA – Algorithmics for MAssive and Networked DAta", and by the National Group for Scientific Computation (GNCS-INdAM).

P. Bose et al. (Eds.): ALGOSENSORS 2015, LNCS 9536, pp. 183–195, 2015.
DOI: 10.1007/978-3-319-28472-9_14

Initially, no robots occupy the same location, and they are assumed to be: *Dimensionless*: modeled as geometric points in the plane; *Anonymous*: no unique identifiers; *Autonomous*: no centralized control; *Oblivious*: no memory of past events; *Homogeneous*: they all execute the same deterministic algorithm; *Asynchronous*: there is no global clock that synchronize their actions; *Silent*: no direct way of communicating; *Unoriented*: no common coordinate system, no compass, no chirality. Robots are equipped with sensors and motion actuators, and operate in *Look-Compute-Move* cycles (see, e.g. [11]). In one cycle a robot takes a snapshot of the current global configuration (Look) in terms of relative robots and meeting-points positions, according to its own coordinate system. Successively, in the Compute phase it decides whether to move toward a specific direction or not, and in the positive case it moves (Move).

During the Look phase, robots are assumed to perceive *multiplicities*, that is, whether a same point is occupied by one or more robots, but not the exact number. In the literature, this is called *global-weak multiplicity detection* [7,11,12]. Herein we simply call it *multiplicity detection*. Note that robots always detect whether a meeting-point and one or more robots occupy the same location.

Cycles are performed asynchronously, i.e., the time between Look, Compute, and Move phases is finite but unbounded, and it is decided by an adversary for each robot. Moreover, during the Look phase, a robot does not perceive whether other robots are moving or not. Hence, robots may move based on outdated perceptions. In fact, due to asynchrony, by the time a robot takes a snapshot of the configuration, this might have drastically changed when it starts moving. The scheduler determining the Look-Compute-Move cycles timing is assumed to be fair, that is, each robot performs its cycle within finite time and infinitely often. In the literature, this kind of scheduler is called Asynchronous (ASYNCH). Different options for the scheduler are: Fully-synchronous (FSYNCH), where all robots are awake and run their Look-Compute-Move cycle concurrently and each phase of the cycle has exactly the same duration for all robots; Semi-synchronous (SSYNCH), that coincides with the FSYNCH model with the only difference that not all robots are necessarily activated during a cycle.

The distance traveled within a move is neither infinite nor infinitesimally small. More precisely, the adversary has also the power to stop a moving robot before it reaches its destination, but there exists an unknown constant $\delta > 0$ such that if the destination point is closer than δ, the robot will reach it, otherwise the robot will be closer to it of at least δ. Note that, without this assumption, an adversary would make it impossible for any robot to ever reach its destination.

Considering the model without meeting-points, the problem has been solved in [5] for any number of robots $n > 2$. Adding meeting-points can sometimes help in designing a gathering algorithm while sometimes can play for the adversary. In fact, meeting-points are like anchors in the plane that never move, and hence if they are "favorably" placed, they may suggest the final gathering point. Contrary, when the placement of the meeting-points induces nasty symmetries, then they can be completely useless in terms of orientation, and it might be a real trouble for the robots to agree on a common meeting-point where to gather.

The rationale behind the choice of introducing meeting-points is twofold. From the one hand, we believe the model is theoretically interesting, as it is a hybrid scenario in between the classical environment where robots freely move in the plane (see, e.g., [1,5]), and the more structured one where robots must move on the vertices of a graphs (see, e.g., [8,13]), implemented here by the set of meeting-points. On the other hand, meeting-points for gathering purposes might be a practical choice when robots move in specific environments where not all places can be candidate to serve as gathering points.

Optimization issues have been addressed in [3,4]. The same strategies cannot be applied here since a wider set of configurations must be now considered.

Our Results. The first contribution is that of introducing the so called meeting-points for the well-know gathering problem under the Look-Compute-Move model. Although the new formulation of the gathering problem seems to be rather close to the original one [5], it turns out to require challenging strategies. In fact, there exist ungatherable configurations, characterized by some symmetries, regardless the number of robots, and the most of configurations have been approached with new stigmergy methodologies since the previous techniques cannot be applied, even those proposed in [3,4].

We fully characterize when GMP can be solved. We exploit the ungatherability results from [3], holding also in the stronger FSYNCH setting. Then, for all other configurations, we design a distributed algorithm that solves the problem for any number $n > 0$ robots. The new algorithm works in the weakest ASYNCH setting.

2 Definitions and General Ungatherability Results

In this section we formally define the GMP problem, and then we recall from [3]: the view of a configuration, relations between symmetries and the view, ungatherability results, and the concept of Weber-points of a configuration.

Problem Definition. The system is composed of n mobile *robots*. At any time, the multiset $R = \{r_1, r_2, \ldots, r_n\}$, with $r_i \in \mathbb{R}^2$, contains the *positions* of all the robots. The set $U(R) = \{x \mid x \in R\}$ contains the *unique* robots' positions. M is a finite set of fixed *meeting-points* in the plane representing the only locations in which robots can be gathered. The pair $\mathcal{C} = (R, M)$ represents a system *configuration*. A configuration $\mathcal{C}$ is *initial* at time t if at that time all robots have distinct positions (i.e., $|U(R)| = n$). A configuration $\mathcal{C}$ is *final* at time t if (i) at that time each robot computes or performs a null movement and (ii) there exists a point $m \in M$ such that $r_i = m$ for each $r_i \in R$; in this case we say that the robots have gathered on point m at time t.

We study the GATHERING ON MEETING-POINTS problem (shortly, GMP), that asks to transform an initial configuration into a final one. A *gathering algorithm* for GMP is a deterministic distributed algorithm that brings the robots in the system to a final configuration in a finite number of cycles from any given initial configuration, regardless the adversary. We say that an initial configuration $\mathcal{C}$ is *ungatherable* if there are no gathering algorithms for GMP with respect to $\mathcal{C}$.

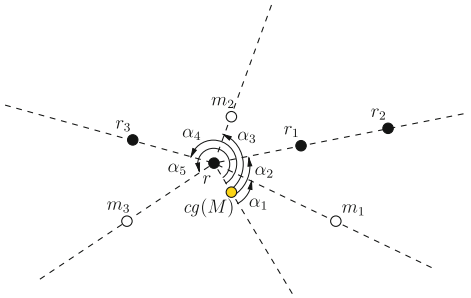

Fig. 1. The counter-clockwise order in which a robot perceives the configuration from r is (r, m_1, r_1, r_2, m_2, r_3, m_3) and $\mathcal{V}^-(r) = (0°, d(r, cg(M)), \mathtt{r}, \alpha_1, d(r, m_1), \mathtt{m}, \alpha_2, d(r, r_1), \mathtt{r}, \alpha_2, d(r, r_2), \mathtt{r}, \alpha_3, d(r, m_2), \mathtt{m}, \alpha_4, (r, r_3), \mathtt{r}, \alpha_5, d(r, m_3), \mathtt{m})$.

Configuration View and Symmetries. Given two distinct points u and v in the plane, let $d(u, v)$ denote their distance, $line(u, v)$ denote the straight line passing through these points, and (u, v) (resp. $[u, v]$) denote the open (resp. closed) segment containing all points in this line that lie between u and v. The half-line starting at point u (but excluding the point u) and passing through v is denoted by $hline(u, v)$. Given two lines $line(c, u)$ and $line(c, v)$, we denote by $\sphericalangle(u, c, v)$ the angle (ranging from zero to less than 360°) centered in c and with sides $hline(c, u)$ and $hline(c, v)$.

Given a configuration $\mathcal{C} = (R, M)$, $cg(M)$ is the *center of gravity* of points in M, that is the point whose coordinates are the mean values of the coordinates of the points of the set. In [3] it has been defined a data structure called *view* and computable by each robot r (according to its local coordinate system) for any point $p \in R \cup M$. Essentially, a robot r that needs to evaluate the view of a point p, first computes $cg(M)$ and then, starting from the direction given by $hline(p, cg(M))$ and looking around from p (in clockwise and counter-clockwise manner), it determines the order $(p = p_0, p_1, \ldots, p_{|U(R)|+|M|})$, $p_i \in R \cup M$, in which all robots and meeting-points appear. From such order of points, the configuration's view is produced by replacing each point p_i with a triple α_i, d_i, x_i formed by, in order and for $i > 0$, $\alpha_i = \sphericalangle(p_0, p, p_i)$, $d_i = d(p, p_i)$, and $x_i \in \{\mathtt{r}, \mathtt{m}, \mathtt{x}\}$ according whether p_i is a robot position, a meeting-point, or a position where a multiplicity occurs (cf. Fig. 1). The triple associated to p_0 represents the point p where d_0 is equal to $d(p, cg(M))$. Finally, by considering $\mathtt{r} < \mathtt{m} < \mathtt{x}$, it is possible to order the two strings $\mathcal{V}^-(p)$ and $\mathcal{V}^+(p)$ associated to the view of each point p according to clockwise or counter-clockwise look. So, the view of p is $\mathcal{V}(p) = \min\{\mathcal{V}^+(p), \mathcal{V}^-(p)\}$, and then $\mathcal{V}(\mathcal{C}) = \bigcup_{p \in U(R) \cup M} \{\mathcal{V}(p)\}$. Notice that, even if robots do not have a common understanding of the handedness (chirality), by computing $\mathcal{V}(\mathcal{C})$ they all get the same information.

Robots can use $\mathcal{V}(\mathcal{C})$ not only to share a common view about $\mathcal{C}$ but also to determine whether a configuration is "symmetric" or not. Let $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ a map from points to points in the plane. It is called an *isometry* or distance preserving if for any $a, b \in \mathbb{R}^2$ one has $d(\varphi(a), \varphi(b)) = d(a, b)$. Examples of isometries in the

plane are *translations*, *rotations* and *reflections*. An isometry φ is a translation if there exists no point x such that $\varphi(x) = x$; it is a rotation if there exists a unique point x such that $\varphi(x) = x$ (and x is called *center of rotation*); it is a reflection if there exists a line ℓ such that $\varphi(x) = x$ for each point $x \in \ell$ (and ℓ is called *axis of reflection*).

An *isometry of an initial configuration* $\mathcal{C} = (R, M)$ is an isometry in the plane that maps robots to robots (i.e., points of R into R) and meeting-points to meeting-points (i.e., points of M into M). Isometries for $\mathcal{C}$ do not include translations as the sets R and M are finite.

If $\mathcal{C}$ admits only the identity isometry, then $\mathcal{C}$ is said *asymmetric*, otherwise it is said *symmetric* (i.e., $\mathcal{C}$ admits rotations or reflections). If $\mathcal{C}$ is symmetric due to an isometry φ, a robot cannot distinguish its position at $r \in R$ from $r' = \varphi(r)$. As a consequence, two robots (e.g., one on r and one on $\varphi(r)$) can decide to move simultaneously, as any algorithm is unable to distinguish between them. In such a case, there might be a so called *pending move*, that is, wlog r performs its entire Look-Compute-Move cycle while r' does not terminate the Move phase, i.e. its move is pending. Clearly, all the other robots performing their cycles are not aware whether there is a pending move, that is they cannot deduce the global status from their view. This fact greatly increases the difficulty to devise a gathering algorithm for symmetric configurations.

The following results states that each robot can use the view $\mathcal{V}(\mathcal{C})$ to determine whether $\mathcal{C}$ is symmetric or not.

Lemma 1 [3]. *An initial configuration* $\mathcal{C} = (R, M)$, $|M| > 1$, *admits a reflection (rotation, resp.) if and only if there exist* $p, q \in R \cup M$, *such that* $\mathcal{V}^+(p) = \mathcal{V}^-(q)$ *with* p *and* q *not necessarily distinct* ($\mathcal{V}^+(p) = \mathcal{V}^+(q)$, *with* $p \neq q$, *resp.).*

From this results we get that, for an asymmetric configuration $\mathcal{C}$, it is unique the point (robot or meeting-point) having the minimum view.

Lemma 2. *Let* $\mathcal{C} = (R, M)$ *be a non-rotational initial configuration, and* ℓ *be a line passing through* $cg(M)$. *If the line perpendicular to* ℓ *at* $cg(M)$ *is not a reflection axis for* $\mathcal{C}$, *then* ℓ *admits a North-South orientation.*

The above lemma implies that, under certain conditions, all robots can agree about the North of a line ℓ passing through $cg(M)$, and in case about the "northernmost" robot or meeting-point on ℓ.

Ungatherability Results. In this section we recall a sufficient condition for a configuration to be ungatherable: if this applies then GMP is not solvable. Note that the results hold also for the case of the synchronous environments FSYNCH.

Corollary 1 [3]. *An initial configuration* $\mathcal{C} = (R, M)$ *is ungatherable even in* FSYNCH *if it admits a rotation with center* c *and* $c \notin R \cup M$ *or it admits a reflection with axis* ℓ *and* $\ell \cap (R \cup M) = \emptyset$.

So, if a configuration admits a reflection (rotation, resp.) then the gathering is possible only if on the axis (center, resp.) there are robots or meeting-points.

Weber-Points. Let $\mathcal{C} = (R, M)$ be an initial configuration. We define the *Weber-distance* of $\mathcal{C}$ as the value $\Delta(\mathcal{C}) = \min_{m \in M} \sum_{r \in R} d(r, m)$. The name of *Weber-distance* is due to the following remark: given a set of points $T \subseteq \mathbb{R}^2$, the *Weber-point* of T is a well known concept and corresponds to a point p such that $p = \text{argmin}_{p' \in \mathbb{R}^2} \sum_{t \in T} d(t, p')$. It is well known that (i) if the points in T are not on a line, then the Weber-point of T is unique [16], and (ii) the Weber-point of T is not computable in general [6]. The *Weber-distance* of a point $m \in M$ in $\mathcal{C}$ is denoted as $wd(\mathcal{C}, m)$ and is defined as $wd(\mathcal{C}, m) = \sum_{r \in R} d(r, m)$. Hence, a point $m \in M$ is called *Weber-point* of $\mathcal{C}$ if $wd(\mathcal{C}, m)$ is minimum, that is $wd(\mathcal{C}, m) = \Delta(\mathcal{C})$. Symbol $wp(\mathcal{C})$ is used to denote the set containing all the Weber-points of $\mathcal{C}$ (notice that the $wp(\mathcal{C})$ may contain more that one point and that such points can be easily computed since M is finite).

We recall now a characterization about the set of Weber-points after the move of a robot toward a Weber-point. From now on, we use the sentence "robot r moves toward a meeting-point m" to mean that r *performs a straight move toward* m *and the final position of* r *lies on the interval* $(r, m]$.

Lemma 3 [3]**.** *Let $\mathcal{C} = (R, M)$ be a configuration with $m \in wp(\mathcal{C})$ and $r \in R$. If $\mathcal{C}' = (R', M)$ is the configuration obtained after r moved toward m, then all the Weber-points in $wp(\mathcal{C}')$ lie on hline(r, m) and $m \in wp(\mathcal{C}') \subseteq wp(\mathcal{C})$.*

3 Gathering for GMP

In this section we provide a solution for the GMP problem. We start by providing a partition of the set $\mathcal{I}$ containing all the possible initial configurations for GMP. According to Corollary 1 there are configurations in $\mathcal{I}$ that are ungatherable. The class of such configurations is denoted by $\mathcal{U}$ and contains any initial configuration $\mathcal{C}$ fulfilling one of the following conditions:

- $\mathcal{C}$ admits a rotation with center c and $c \notin R \cup M$;
- $\mathcal{C}$ admits a reflection with axis ℓ and $\ell \cap (R \cup M) = \emptyset$.

In the remaining of this section we provide a gathering algorithm for the GMP problem in the most general ASYNCH setting when the input configuration (R, M) is restricted to $\mathcal{I} \setminus \mathcal{U}$. Moreover we assume $|R| > 1$ and $|M| > 1$, as otherwise the solution is straightforward: in fact, if $|R| = 1$ it is sufficient that the only robot reaches a meeting-point and if $|M| = 1$ all the robots can move toward the only meeting-point.

Before starting the description of the algorithm we introduce some additional concepts and notation. Given a configuration $\mathcal{C}$, let $O_1, O_2, \ldots, O_t$, $t \geq 1$, be all the circles centered in $cg(M)$ such that $O_i \cap R \neq \emptyset$ for each $1 \leq i \leq t$. Moreover, ρ_i represents the radius of O_i, and we assume $\rho_i < \rho_j$ if $i < j$. If a robot is on $cg(M)$, then $\rho_1 = 0$. Let O_M be the smallest circle that is centered in $cg(M)$ and contains all points in M, and let ρ_M be its radius.

All the initial configurations processed by the algorithm, along with those with one multiplicity created during the execution, are partitioned as follows:

```
   Procedure: Compute
   Input: Configuration C = (R, M)
 1 Let c = cg(M), d = max{ρ_{t-1}, ρ_M}, d' = max{ρ_t, ρ_M};
 2 if C ∈ S_1 then move toward the meeting-point with unique multiplicity;
 3 if C ∈ S_2 then move toward cg(M);
 4 if C ∈ S_3 then
 5     if r on cg(M) then move at distance ρ_2/2 from cg(M) in any direction;
 6 if C ∈ S_4 then
 7     if r on O_1 then move at distance ρ_2/2 from cg(M) on hline(cg(M), r);
 8 if C ∈ S_5 then
 9     NumGuards = 0;
10     Call Guards()                    /* Guards modifies NumGuards */ ;
11     if NumGuards ≠ 0 then
12         Call MakeMultiplicity();
13 if C ∈ S_6 then Call AtMost3Bots();
```

$\mathcal{S}_0$: any final configuration $\mathcal{C}$ where all the robots form one multiplicity on some meeting-point;
$\mathcal{S}_1$: any configuration $\mathcal{C} \notin \mathcal{S}_0$ with only one multiplicity on some meeting-point;
$\mathcal{S}_2$: any $\mathcal{C} = (R, M) \notin \bigcup_{i=0}^{1} \mathcal{S}_i$, with $cg(M) \in M$;
$\mathcal{S}_3$: any $\mathcal{C} \notin \bigcup_{i=0}^{2} \mathcal{S}_i$ admitting a rotation;
$\mathcal{S}_4$: any $\mathcal{C} = (R, M) \notin \bigcup_{i=0}^{2} \mathcal{S}_i$ with one robot r on O_1 such that $0 < \rho_1 < \rho_2/2$ and $(R \setminus \{r\}, M) \in \mathcal{S}_3$.
$\mathcal{S}_5$: any $\mathcal{C} \notin \bigcup_{i=0}^{4} \mathcal{S}_i$, with more than 3 robots.
$\mathcal{S}_6$: any $\mathcal{C} \notin \bigcup_{i=0}^{4} \mathcal{S}_i$, with at most 3 robots.

It easily follows that $\{\mathcal{U}, \mathcal{S}_2, \mathcal{S}_3, \ldots, \mathcal{S}_6\}$ is a partition of $\mathcal{I}$. Note that configurations in classes $\mathcal{S}_0$ and $\mathcal{S}_1$ are the only non-initial ones handled by the algorithm.

The general algorithm, executed by each robot, is represented by Procedure Compute. It is divided into six parts, according to the subdivision of the non-final configurations. The procedure and the sub-procedures represent what a generic robot r executes during the Compute phase once it has detected the class which the perceived configuration belongs to.

The general strategy of the algorithm is to transform each initial configuration $\mathcal{C} \in \bigcup_{i=2}^{6} \mathcal{S}_i$ into a configuration $\mathcal{C}' \in \mathcal{S}_1$ by moving robots to create a multiplicity on some meeting-point m. Once this occurs, all robots can always detect m and the gathering can be easily finalized. This approach is easy to obtain when there is a meeting-point m on $cg(M)$ (i.e., configurations in $\mathcal{S}_2$) since m is always recognizable by all robots. Whereas, it is not applied in rotational (or quasi-rotational) configurations with a robot on $cg(M)$ (i.e., configurations in $\mathcal{S}_3$ or $\mathcal{S}_4$) that are transformed in configurations in $\mathcal{S}_5$ or $\mathcal{S}_6$.

Then, the core of the algorithm is given for cases $\mathcal{S}_5$ and $\mathcal{S}_6$ where some sub-procedures later defined are invoked. For handling configurations in such classes, the strategy of our algorithm is composed of four phases:

1. select one or two robots, denoted as guard(s);
2. place suitably the guard(s), if required;

3. crate a multiplicity by means of robots not designated as guard(s);
4. finalize the gathering on the created multiplicity.

The selection of the guard(s) is done among the robots furthest from $cg(M)$. Due to the limited number of symmetries that a configuration can admit, it is always possible to select one or two robots that are moved away from $cg(M)$ so that they are always recognizable. The algorithm selects two guards, which are equivalent, only for reflexive configurations where a single guard cannot be pointed out. As final positioning for the guards, we chose a sufficiently large distance from $cg(M)$ that depends on the radius of the current O_{t-1}. Once guards are correctly placed, all other robots cooperate in order to create a multiplicity on a meeting-point. In practice, stigmergy paradigms are applied, that is guards are used in place of a compass so that all robots get oriented by observing their positions. From this orientation they deduce what will be the final gathering point and hence move there. Such a point is a meeting-point m that maintains its peculiarity while robots move toward it. For instance, if we chose among the meeting-points the northernmost of minimum Weber distance, then as soon as robots start moving toward it, its Weber distance decreases. Eventually, the meeting-point m will remain the only one of minimum Weber distance, according to Lemma 3.

According to the assumed multiplicity detection, once a multiplicity is created, robots are no longer able to compute the Weber distance accurately. Hence, our strategy assures to create the first multiplicity over m, and once this happens all robots move toward it without creating other multiplicities. Moves are always computed without creating undesired multiplicities. Clearly, the movement of the guards (during the above phase 2) cannot create any multiplicity since they are moved from O_t to the outer space. Afterward, since robots move straightly toward m, then two robots meet only at the final destination point, unless they move along the same direction. In such a case, we make robots move without overtaking each other. In particular if a robot r is moving toward a point p and there is another robot r' in the open segment (r, p), then r moves toward a point p' on (r, p) such that $d(r, p') = \frac{d(r,r')}{2}$. In this way, undesired multiplicities are never created. Once a multiplicity is created on m, it is then easy to move all other robots (including the guards) toward it, by exploiting the multiplicity detection. Hence the gathering is easily finalized.

The next theorem provides the correctness of our algorithm.

Theorem 1. *There exists a gathering algorithm that solves the* GMP *problem for an initial configuration* $\mathcal{C}$ *if and only if* $\mathcal{C} \in \mathcal{I} \setminus \mathcal{U}$.

3.1 Classes $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$, *and* $\mathcal{S}_4$

In this section we describe how configurations in the first four classes are handled by our algorithm. Concerning classes $\mathcal{S}_1$ and $\mathcal{S}_2$, robots can move concurrently toward the unique multiplicity, or $cg(M)$, respectively.

Lemma 4. *Given a configuration* $\mathcal{C}$ *in class* $\mathcal{S}_1$ *or* $\mathcal{S}_2$, *Procedure* COMPUTE *leads to a configuration* $\mathcal{C}'$ *in class* $\mathcal{S}_0$, *eventually.*

Consider now the case where the initial configuration $\mathcal{C}$ admits a rotation ($\mathcal{C} \in \mathcal{S}_3$) or it "almost" admits a rotation ($\mathcal{C} \in \mathcal{S}_4$). Since in case of rotations it is generally impossible to identify specific robots suitable for the role of guards, when $\mathcal{C}$ is in $\mathcal{S}_3$ the algorithm first breaks this symmetry by moving the robot in the center of the rotation c. Notice that a robot must be on c as otherwise either $\mathcal{C} \in \mathcal{S}_2$ or $\mathcal{C} \in \mathcal{U}$. Class $\mathcal{S}_4$ has been introduced to assure that the robot moving from c reaches a target (a distance $\rho_2/2$ from c) and stops there before the positioning of the guard(s) starts.

Lemma 5. *Given an initial configuration $\mathcal{C} = (R, M)$ in class $\mathcal{S}_3$ or $\mathcal{S}_4$, Procedure* Compute *leads to a configuration $\mathcal{C}'$ in class $\mathcal{S}_5$ or $\mathcal{S}_6$, eventually.*

3.2 Class $\mathcal{S}_5$

In this section, configurations of class $\mathcal{S}_5$ are addressed. As described at the beginning of the section, our algorithm makes use of a stigmergy paradigm, that is some robots (namely, the "guards") are used in place of a compass so that all robots get oriented by observing their positions. We now formally define the guards of a configuration.

Definition 1. *Let $\mathcal{C} = (R, M)$ be an initial configuration with $\rho_t \geq d = 3 \cdot \max\{\rho_{t-1}, \rho_M\}$. If $O_t \cap R = \{r\}$ then r is a* guard. *Assume $O_t \cap R = \{r_1, r_2\}$ and r_1, r_2 are symmetric with respect to an axis ℓ of M. If $\mathcal{C}$ is reflexive according to ℓ or there exists a meeting-point which is unique according to some property, then r_1 and r_2 are two* guards.

According to the notion of guards, in order to finalize the gathering, the configuration evolves in four different stages: the first two stages are devoted to (1) select one or two robots that are guards or that can be moved until they become guards, and (2) suitably move the guards, if necessary. These first two stages are realized by means of Procedure Guards. The third stage concerns (3) making a multiplicity by means of robots that are not guards. Procedure MakeMultiplicity realizes it. The last stage requires (4) finalizing the gathering on the created multiplicity according to Lemma 4.

We now shortly introduce Procedure Guards applied by robots in order to detect whether the guards exist or if they have to be created. Procedure Guards is invoked when it is recognized that the initial configuration $\mathcal{C}$ taken as input belongs to $\mathcal{S}_5$. Procedure Guards checks how many robots reside on O_t and then calls the corresponding subroutine. Note that, according to Definition 1, guards are at distance at least $3d$ from the center $cg(M)$, while all the remaining robots and all the meeting-points are at distance at most d from $cg(M)$. In this way, once guards are recognized, the other robots start moving toward a specific meeting-point and they will never exceed distance d from $cg(M)$.

Conversely, when there are no guards, the configuration must be transformed so that a new configuration with one or two guards is created. During the transformation, one or two robots must be selected and moved far from $cg(M)$ until

```
Procedure: Guards
Input: Configuration C = (R, M) with circles O_i, i = 1, 2, ..., t
1 if O_t ∩ R = {r_1} then Call CheckOne(C, r_1);
2 if O_t ∩ R = {r_1, r_2} then Call CheckTwo(C, r_1, r_2);
3 if |O_t ∩ R| > 2 then Call CheckMany(C);
```

```
Procedure: CheckOne
Input: Configuration C = (R, M) and robot r_1 on O_t
1 if ρ_t ≥ 3d then NumGuards = 1; exit;
2 Let x = hline(c, r_1) ∩ O_{t−1};
3 Let C' be the configuration obtainable from C by assuming r_1 on x;
4 if ρ_t − ρ_{t−1} ≤ d and C' is reflexive with axis ℓ ≠ line(c, x) and C' ∈ S_5 then
5 |   Let r_2 be the robot symmetric to r_1 in C' with respect to ℓ;
6 |   if r = r_2 then
7 |   |   move r on hline(c, r) at distance ρ_t from c;        /* Possible pending move */
8 else
9 |   if r = r_1 then move r on hline(c, x) at distance 3d from c;
```

reaching a position compatible with the definition of guards. In particular, if two guards must be created, then the designed robots must move at distance $3d$ from $cg(M)$ in two steps: first they are moved both at distance $2d$, and afterward $3d$. This double step is done so as the difference between the distances of the two guards (that might move asynchronously) from $cg(M)$ is always kept below d in order to distinguish the case where only one guard must be created. During the phase in which robots are moved to create guards, d is initially defined as the maximum between ρ_{t-1} and ρ_M. Sometimes we make use of $d' = \max\{\rho_t, \rho_M\}$ instead of d; this happens when the guard(s) have not started moving yet, and O_t is occupied also by robots that will not become guards.

Once Guards completed its task (and, due to the adversary, it may require several but finite many executions made by the guard robots), the variable NumGuards is set to one or two and hence MakeMultiplicity starts. Of course, also this procedure may require several executions to complete its task. Then, according to Lines 8–12 of Procedure Compute, Procedure Guards is called again each time MakeMultiplicity restarts. In such executions, Guards has only to recognize that guard(s) are settled.

So, the next lemmata can be stated.

Lemma 6. *Given a configuration $\mathcal{C} = (R, M)$ in class $\mathcal{S}_5$, Procedure* Guards *eventually leads to a configuration $\mathcal{C}' \in \mathcal{S}_5$ with 1 or 2 guards. Moreover,*

(i) if $\mathcal{C}'$ has one guard, then either $\mathcal{C}'$ is asymmetric or it admits a reflection axis with the guard on the axis;
(ii) if $\mathcal{C}'$ has two guards, then either $\mathcal{C}'$ is reflexive and the two guards are symmetric, or $\mathcal{C}'$ is asymmetric, ℓ is a reflection axis for M, and the two guards are symmetric with respect to ℓ.

Lemma 7. *If Procedure* Guards *returns a configuration $\mathcal{C}$ with 1 or 2 guards, then Procedure* MakeMultiplicity *leads to a configuration $\mathcal{C}' \in \mathcal{S}_1$, eventually.*

```
Procedure: CheckTwo
Input: Configuration C = (R, M) and robots r1 and r2 on Ot
Let c = cg(M);
if C is reflexive then
    Let ℓ be the axis of symmetry;
    if r1 and r2 are on ℓ then
        Let r' be the northernmost robot between r1 and r2;
        if r = r' then move r on hline(c, r) at distance 3d' from c;
    else
        if ρt ≥ 3d then NumGuards = 2; exit;
        if ρt ≥ 2d then
            if r ∈ {r1, r2} then move r on hline(c, r) at distance 3d from c;
        else
            if r ∈ {r1, r2} then move r on hline(c, r) at distance 2d from c;
else
    if ρt ≥ 3d then
        Let ℓ be the bisector of α = ◁(r1, c, r2);
        if ℓ is an axis of reflection for M then
            if M ∩ ℓ ≠ ∅ then NumGuards = 2; exit;
            else
                Let M' be the set of meeting-points in M ∩ OM, closest to ℓ, and with
                minimum Weber-distance;
                if |M'| = 1 then NumGuards = 2; exit;
    if r is the robot on Ot with minimum view then
        move r on hline(c, r) at distance 3d' from c
```

```
Procedure: CheckMany
Input: Configuration C = (R, M)
if C is reflexive with axis ℓ then
    Let r1 and r2 be the robots on Ot that are not on ℓ but closest to it (and the
    northernmost in case of ties);
    if r ∈ {r1, r2} then move r on hline(c, r) at distance 2d' from c;
else
    if r is the robot on Ot with minimum view then
        move r on hline(c, r) at distance 3d' from c
```

3.3 Class $\mathcal{S}_6$

Since each configuration $\mathcal{C} \in \mathcal{S}_6$ has two or three robots only, then the approach of Sect. 3.2 that makes use of guards cannot be always applied since there might be not enough remaining robots to create a multiplicity. Here we briefly summarize the strategy implemented by Procedure AtMost3Bots().

If $\mathcal{C}$ has only two robots r_1 and r_2 and is reflexive without robots on the axis ℓ, then the approach is to move both the symmetric robots by small steps toward a meeting-point m on the axis. Small steps are required in order to maintain ℓ as recognizable. During the movements, the following invariant is used: the triangles (r_1, m, h_1) and (r_2, m, h_2), where h_1 and h_2 are the projections of r_1 and r_2 on ℓ, respectively, remain similar after the movements. The small movements are repeated until the two robots reach m.

Procedure: MakeMultiplicity
Input: Configuration $\mathcal{C} = (R, M)$

if NumGuards $= 1$ **then**
  Let g be the guard and let $\ell = hline(c, g)$;
  if $M \cap \ell \neq \emptyset$ **then**
    Let $m \in M \cap \ell$ be the meeting-point closest to g;
    if $r \neq g$ **then** move r toward m;
  else
    Let $X = \bigcup_{m \in M} hline(c, m) \cap O_t$;
    if $r = g$ **then** move r on O_t toward any closest point in X;
if NumGuards $= 2$ **then**
  Let g_1, g_2 be the guards and ℓ be the bisector of $\sphericalangle(g_1, c, g_2)$;
  Let $M' = M \cap \ell$;
  if $|M'| \neq \emptyset$ **then**
    Let $M'_W \subseteq M'$ be the set of meeting-points with minimum Weber-distance;
    if $|M'_W| = 2$ **then**
      Let m be the northernmost meeting-point in M'_W;
      if $r \notin \{g_1, g_2\}$ *and* r *is not on* ℓ **then** move r toward m;
    else
      if $r \notin \{g_1, g_2\}$ **then** move r toward m, being $M'_W = \{m\}$;
  else
    Let M'' be the set of meeting-points on O_M, closest to ℓ, and with minimum Weber-distance;
    if $M'' = \{m\}$ **then**
      if $r \notin \{g_1, g_2\}$ **then** move r toward m;
    else
      if r *is the robot on* ℓ *with minimum view* **then**
        move r toward any $m \in M''$;

In all the other cases one guard is created. In particular: if the there are three robots, once the guard is created the remaining robots can create a multiplicity (and hence a configuration in $\mathcal{S}_1$ is created); if there are two robots, once the guard is created then (1) the other robot can move toward the meeting-point closest to the guard, and (2) the guard can move toward the occupied meeting-point until finalizing the gathering.

4 Conclusion

We have studied a new version of the gathering problem of anonymous and oblivious robots in the plane. Robots are required to gather at some predetermined meeting-points. Robots operate in the Look-Compute-Move cycle model empowered with the multiplicity detection. We provide a new deterministic distributed gathering algorithm that solves the problem for all initial configurations but those proved to be ungatherable. Introducing meeting-points is a natural and challenging choice, and the resolution of the gathering problem within this model is of main interest in robot-based computing systems.

Revisiting other existing models for the gathering or even other problems with respect to the meeting-points represents intriguing research directions.

References

1. Bouzid, Z., Das, S., Tixeuil, S.: Gathering of mobile robots tolerating multiple crash faults. In: IEEE 33rd International Conference on Distributed Computing Systems (ICDCS), pp. 337–346 (2013)
2. Chalopin, J., Dieudonné, Y., Labourel, A., Pelc, A.: Fault-tolerant rendezvous in networks. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014, Part II. LNCS, vol. 8573, pp. 411–422. Springer, Heidelberg (2014)
3. Cicerone, S., Di Stefano, G., Navarra, A.: Minimum-traveled-distance gathering of oblivious robots over given meeting points. In: Gao, J., Efrat, A., Fekete, S.P., Zhang, Y. (eds.) ALGOSENSORS 2014, LNCS 8847. LNCS, vol. 8847, pp. 57–72. Springer, Heidelberg (2015)
4. Cicerone, S., Di Stefano, G., Navarra, A.: MinMax-distance gathering on given meeting points. In: Paschos, V.T., Widmayer, P. (eds.) CIAC 2015. LNCS, vol. 9079, pp. 127–139. Springer, Heidelberg (2015)
5. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: gathering. SIAM J. Comput. **41**(4), 829–879 (2012)
6. Cockayne, E.J., Melzak, Z.A.: Euclidean constructibility in graph-minimization problems. Math. Mag. **42**(4), 206–208 (1969)
7. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering asynchronous and oblivious robots on basic graph topologies under the look-compute-move model. In: Alpern, S., Fokkink, R., Gąsieniec, L., Lindelauf, R., Subrahmanian, V.S. (eds.) Search Theory: A Game Theoretic Perspective, pp. 197–222. Springer, New York (2013)
8. D'Angelo, G., Di Stefano, G., Navarra, A.: Gathering on rings under the look-compute-move model. Distrib. Comput. **27**(4), 255–285 (2014)
9. Degener, B., Kempkes, B., Langner, T., Meyer auf der Heide, F., Pietrzyk, P., Wattenhofer, R.: A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In: 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 139–148 (2011)
10. Farrugia, A., Gąsieniec, L., Kuszner, Ł., Pacheco, E.: Deterministic rendezvous in restricted graphs. In: Italiano, G.F., Margaria-Steffen, T., Pokorný, J., Quisquater, J.-J., Wattenhofer, R. (eds.) SOFSEM 2015-Testing. LNCS, vol. 8939, pp. 189–200. Springer, Heidelberg (2015)
11. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, San Rafael (2012)
12. Izumi, T., Izumi, T., Kamei, S., Ooshita, F.: Randomized gathering of mobile robots with local-multiplicity detection. In: Guerraoui, R., Petit, F. (eds.) SSS 2009. LNCS, vol. 5873, pp. 384–398. Springer, Heidelberg (2009)
13. Klasing, R., Markou, E., Pelc, A.: Gathering asynchronous oblivious mobile robots in a ring. Theoret. Comput. Sci. **390**, 27–39 (2008)
14. Kranakis, E., Krizanc, D., Markou, E.: The Mobile Agent Rendezvous Problem in the Ring. Morgan & Claypool, San Rafael (2010)
15. Pelc, A.: Deterministic rendezvous in networks: a comprehensive survey. Networks **59**(3), 331–347 (2012)
16. Weiszfeld, E.: Sur le point pour lequel la somme des distances de n points donnés est minimum. Tohoku Math. **43**, 355–386 (1936)