Gathering Autonomous Mobile Robots

MARK CIELIEBAK
ETH Zurich, Switzerland

GIUSEPPE PRENCIPE
Università di Pisa, Italy

Abstract

We study the problem of coordinating a set of autonomous mobile robots that can freely move in a two-dimensional plane; in particular, we want them to gather at a point not fixed in advance (GATHERING PROBLEM). We introduce a model of weak robots (decentralized, asynchronous, no common knowledge, no identities, no central coordination, no direct communication, oblivious) which can observe the set of all points in the plane which are occupied by other robots. Based on this observation, a robot uses a deterministic algorithm to compute a destination, and moves there. We prove that these robots are too weak to gather at a point in finite time. Therefore, we strengthen them with the ability to detect whether more than one robot is at a point (*multiplicity*). We analyze the GATHERING PROBLEM for these stronger robots. We show that the problem is still unsolvable if there are only two robots in the system. For 3 and 4 robots, we give algorithms that solve the GATHERING PROBLEM. For more than 4 robots, we present an algorithm that gathers the robots in finite time if they are not in a specific symmetric configuration at the beginning (*biangular configuration*). We show how to solve such initial configurations separately. However, the general solution of the GATHERING PROBLEM remains an open problem.

Keywords

Autonomous Mobile Robots, Gathering, Distributed Coordination.

1 Introduction

Consider a distributed system whose entities are autonomous mobile robots modeled as devices with computational capabilities that are able to freely move in a two-dimensional plane. We study the problem of coordinating these robots. The coordination mechanism is totally *decentralized*, i.e., the robots are completely *autonomous* and no central control is used. The objective is to gather all robots at one point. This point is not fixed in advance.

The GATHERING PROBLEM is one of the basic interaction primitives studied in a system populated by a set of autonomous mobile robots [7]. The problem of coordinating a collection of autonomous mobile robots has been studied in robotics and in artificial intelligence [2, 6, 7]. Mostly, the problem is approached from an experimental point of view: algorithms are designed using mainly heuristics, and then tested either by means of computer simulations or with real robots. Neither proofs of correctness of the algorithms, nor any analysis of the relationship between the problem to be solved, the capabilities of the robots employed, and the robots' knowledge of the environment are given. Recently, concerns on computability and complexity of the coordination problem have motivated *algorithmic* investigations, and the problem has also been approached from a *computational* point of view [1, 4, 5, 8, 9, 10]. In [1] and [10], any action of the robots, including moving, is *instantaneous*, while in [4, 5, 8, 9], as well as in this paper, there is no such assumption.

We consider a very weak model of robots: The robots are anonymous, have no common knowledge, no central coordination, and no means of direct communication. Initially, they are in a waiting state. They wake up asynchronously, observe the other robots' positions, compute a point in the plane, move towards this points (but may not reach it[1]) and become waiting again. Each step takes an unpredictable amount of time. The model is described in detail in the next section.

We can show that these robots cannot gather if they have no additional abilities. One such ability is to detect multiplicities, i.e., to be able to detect whether there is more than one robot at the same point. This ability is used as follows: first let at least two robots meet at a point, so that there is a point in the plane with multiplicity greater than one; then, all the remaining robots move to this point (thereby, they avoid to generate another point with multiplicity greater than one).

For less than five robots, there exist simple algorithms to gather the robots at a point. The problem becomes challenging for $n \geq 5$ robots: How can we gather a large set of robots? An easy solution would be to gather the robots at the Weber point (*WP*): the unique point in the plane that minimizes the sum of the distances between itself and all the points in a given set of points [11]. An interesting property of the Weber point is that it does not change when moving any point towards it. Hence, the robots can simply gather in *WP*. Unfortunately, the Weber point is not computable [3]. Therefore, we need a different strategy. Assume that the initial configuration of the robots is a regular n-gon. This is a *totally symmetric* configuration, and either all or no robots can be selected to move. In this case, the center of the n-gon can be used as a gathering point. Even in the more general case of biangular configurations with center c (i.e., there exist two angles α and β such that all angles between two adjacent robots w.r.t. c are either α or β, and they alternate, see Figure 2.a), the robots can meet in c. On the other hand, we can

[1] That is, a robot can stop before reaching its destination point, e.g. because of limits to the robot's motion energy.

design an (already rather sophisticated) algorithm for the GATHERING PROBLEM (see Section 4), if the initial configuration is not biangular. The remaining challenge is to combine these two cases, i.e., to design an algorithm which solves the GATHERING PROBLEM for any initial configuration. This seems to be a hard problem, and we conjecture that it is unsolvable. For the case $n \geq 5$, in this paper we focus on describing an algorithm that gather the robots when they are not in a biangular configuration at the beginning.

The rest of the paper is organized as follows. In the following two sections, we define the model of robots we are using, the problems to solve, and some notations. In Section 3, we provide solutions for the GATHERING PROBLEM for $n = 2, 3, 4$. In Section 4, we present a solution for the GATHERING PROBLEM for arbitrary n, when the robots are not in a biangular configuration at the beginning. Conclusions are drawn in Section 5. Proofs are mostly omitted due to space limitations, and can be found in [9].

2 Model and Definitions

2.1 Autonomous Mobile Robots

Each robot is viewed as a point, and it is equipped with sensors that let it observe the set of all points in the plane which are occupied by at least one other robot, and form its *local view* of the world. Note that a robot only knows *whether* there are other robots at a specific point, but it has no knowledge about their *number*. The local view of each robot includes a unit of length, an origin (which we will assume w.l.o.g. to be the position of the robot in its current observation), and a coordinate system (e.g. Cartesian). We do not assume any kind of agreement among the robots on the unit of length, the origin, or the local coordinate systems.

A robot is initially in a *waiting* state (*Wait*). Asynchronously and independently from the other robots, it *observes* the environment (*Look*) by activating its sensors. The sensors return a snapshot of the world, i.e. the set of all points which are occupied by at least one other robot, with respect to the local coordinate system. The robot then *calculates* its destination point (*Compute*) according to its deterministic algorithm, based only on its local view of the world. Each robot executes the same deterministic algorithm. It then *moves* towards the destination point (*Move*); if the destination point is the current location, the robot stays still. After an unpredictable time (during or after the move), the robot returns to the waiting state. Therefore, it may or may not reach its destination point during the move. The sequence *Wait* - *Look* - *Compute* - *Move* forms a *cycle* of a robot.

The robots are *fully asynchronous*, that is the amount of time spent in each phase of a cycle is finite but otherwise unpredictable. In particular, the robots do not have a common notion of time. As a result, robots can be seen by the other robots while moving, and thus computations can be made based on obso-

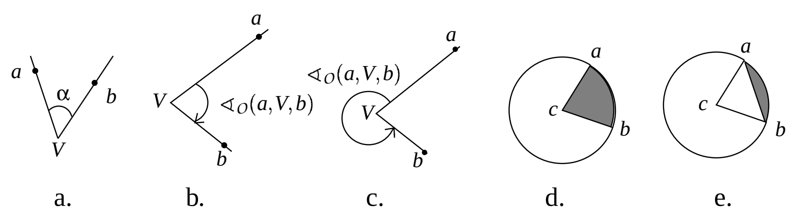

Figure 1: (a) Convex angle $\sphericalangle(a,V,b)$. (b), (c) Angles $\sphericalangle_O(a,V,b)$ with clockwise and counterclockwise orientation, respectively. (d) Arc (thick line) and sector (grey part) defined by $\sphericalangle(a,c,b)$. (e) Grey area represents $segm(\sphericalangle(a,c,b))$.

lete observations. The robots are *oblivious*, meaning that they do not remember any previous observations nor computations performed in any previous step. The robots are *anonymous*, meaning that they are a priori indistinguishable by their appearance, and they do not have any kind of identifiers that can be used during the computation. The robots have *no means of direct communication*: any communication occurs in a totally implicit manner, by observing the other robots' positions.

There are two limiting assumptions concerning *infinity*: **(A1)** The amount of time required by a robot to complete a cycle is not infinite, nor infinitesimally small. **(A2)** The distance traveled by a robot in a cycle is not infinite, nor infinitesimally small (unless it brings the robot to the destination point). As no other assumptions on space exists, the distance traveled by a robot in a cycle is unpredictable.

2.2 The Gathering Problem

The GATHERING PROBLEM is defined as follows.

> Given n robots $r_1, \dots, r_n$, arbitrarily placed in the plane, with no two robots in the same position, make them gather at one point in a finite number of cycles.

A relaxed variant of this problem would be to make the robots only move "very close" to each other. This variant is rather easy to solve: each robot computes the center of gravity[2] of all robots, and moves towards it. However, in the GATHERING PROBLEM, we want the robots to meet *exactly* at one point.

Theorem 1 *There exists no deterministic oblivious algorithm that solves the* GATHERING PROBLEM *in a finite number of cycles for a set of $n \geq 2$ robots with no additional abilities.*

[2]For n points $x_1, \dots, x_n$ in the plane, the center of gravity is $c := \frac{1}{n}\sum_{i=1}^{n} x_i$.

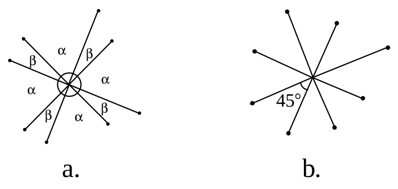

Figure 2: (a) Biangular and (b) equiangular configuration of 8 points.

Proof. *(sketch)* Gathering two robots with no additional abilities is impossible (see [10]). Assume there are $n > 2$ robots. If they meet at a point, then at some time - immediately before they meet - they may occupy exactly two different positions, say x and y (this can be achieved using an adversary which makes them become waiting at the right moments). If the robots have no additional abilities, this looks exactly the same as if there are only two robots in the plane, in positions x and y. Hence, they cannot gather. For a detailed proof, see [9]. $\square$

Because of Theorem 1, we strengthen our robots by *multiplicity detection*: the robots can distinguish whether there is zero, one, or more than one robot at a specific point in the plane. If the multiplicity at a point p is greater than one, we say that there is *strict multiplicity* at p.

We will heavily exploit the multiplicity detection in our algorithms by always gathering the robots at the only position where strict multiplicity occurs.

2.3 Notation

In the rest of the paper, the following notation will be used (refer to Figure 1). Given two distinct points a and b in the plane, $[a,b)$ denotes the half-line which starts in a and passes through b, and $[a,b]$ denotes the line segment between a and b. We denote by $dist(a,b)$ the Euclidean distance between the two points. Given two half-lines $[V,a)$ and $[V,b)$, we denote by $\sphericalangle(a,V,b)$ the convex angle (i.e., the angle which is at most 180°) centered in V and with sides $[V,a)$ and $[V,b)$. Given an orientation O (clockwise or counterclockwise), we denote by $\sphericalangle_O(a,V,b)$ the oriented angle centered in V, i.e., the angle from $[V,a)$ to $[V,b)$ according to the orientation O.

Given a circle $\mathcal{C}$, the intersection between the circumference of $\mathcal{C}$ and an angle α at the center of $\mathcal{C}$ is denoted by $arc(\alpha)$; the intersection between α and $\mathcal{C}$ is denoted by $sector(\alpha)$; and the area of the circle delimited by $[a,b]$ and $arc(\alpha)$ is denoted by $segm(\alpha)$, with a and b being the endpoints of $arc(\alpha)$.

Given points a, b and c, the triangle with these three points as vertices is denoted by $\triangle(a,b,c)$. We use $p \in \triangle(a,b,c)$ to indicate that p is inside the triangle or on its border.

We say that n distinct points in the plane are in *biangular* configuration (refer

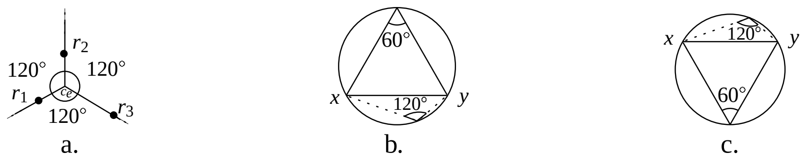

Figure 3: (a) Center of equiangularity of r_1, r_2 and r_3. (b)–(c) The two 120°-circles defined by x and y.

to Figure 2.a) if there exists a point c (the *center*), an ordering of the points, and two angles α and β such that each two adjacent points form an angle α or β w.r.t. c, and the angles alternate. If the points are in biangular position, then the center c is unique, can be computed in finite time, and is invariant under movement in its direction; that is, it does not change if any of the points moves towards c. If $\alpha = \beta$, we say that the points are in *equiangular* configuration (see Figure 2.b).

3 Solving the Gathering Problem for $n \leq 4$

In the following, we will provide solutions for the GATHERING PROBLEM separately for $n = 2, 3$ and 4 robots. The general idea of the algorithms is to let the robots reach a configuration where there is exactly one point q in the plane with strict multiplicity (recall that there is no strict multiplicity in the initial configuration, and that the robots can detect multiplicities). When such a configuration is reached, all the robots move towards q avoiding collisions (i.e., q remains the only point with strict multiplicity). This is accomplished by the routine `move_to`(q) that moves robot r towards q in the plane as follows: if r is already on q, it does not move at all; we use `do_nothing()` to indicate that a robot does not move. If no robot is on the segment $[r, q]$, then r moves towards q. Otherwise, let r' be the robot on $[r, q]$ closest to r; then r is moved to a point at distance at most $d > 0$ from r'. In this way, collisions are avoided.

3.1 Two Robots

Suzuki *et al.* proved in [10] that the GATHERING PROBLEM is unsolvable for two robots if the robots have no additional abilities. On the other hand, if the robots can detect that they “run into each other”, i.e. they stop immediately when they move on the same line in opposite directions and they meet, then the problem can be solved easily: each one of the two robots simply starts moving towards the other robot. Note that this ability differs from multiplicity detection, since here we assume that they recognize the fact that they “run into each other” *while* they move, and that they stop immediately, whereas multiplicity detection is only used

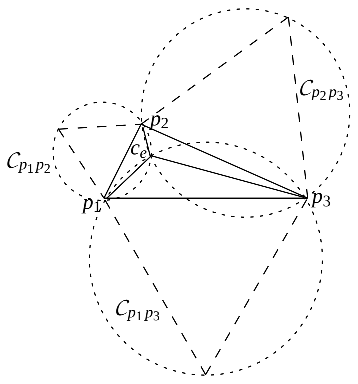

Figure 4: Center of equiangularity c_e lies on the intersection of the three 120°-circles $C_{p_1p_2}$, $C_{p_2p_3}$ and $C_{p_1p_3}$.

during the computation step.

Result 1 *The* GATHERING PROBLEM *is unsolvable for two robots without additional abilities. If the robots can detect when they "run into each other", there exists an algorithm which solves the* GATHERING PROBLEM.

3.2 Three Robots

In order to solve the GATHERING PROBLEM for $n = 3$ robots, we first present some geometric basics, which we will use in the algorithm (Algorithm 1). Given three distinct points p_1, p_2 and p_3, we say that a point c_e is the *center of equiangularity* of p_1, p_2 and p_3, if $\sphericalangle(p_1, c_e, p_2) = \sphericalangle(p_1, c_e, p_3) = \sphericalangle(p_2, c_e, p_3) = 120°$ (see Figure 3.a). If the center of equiangularity exists, then it is unique. If c_e is the center of equiangularity of p_1, p_2 and p_3, then c_e is the center of equiangularity of p'_1, p'_2 and p'_3 for all $p'_i \in [p_i, c_e), 1 \leq i \leq 3$. Thus, moving the points towards c_e does not change the center of equiangularity.

Given two distinct points x and y, we say that circle $\mathcal{C}_{xy}$ with center c is a *120°-circle* if $\sphericalangle(x, p, y) = 120°$ for all $p \in arc(\sphericalangle(x, c, y))$. There exist two 120°-circles for x and y, depending on the orientation (refer to Figure 3.b–c).

If the center of equiangularity of p_1, p_2 and p_3 exists, then it lies on the intersection of three 120°-circles $C_{p_1p_2}$, $C_{p_2p_3}$ and $C_{p_1p_3}$ (refer to Figure 4).

Lemma 1 *Consider three distinct points p_1, p_2 and p_3 such that $\triangle(p_1, p_2, p_3)$ does not contain an angle greater than or equal to* 120°. *Then the center of equiangularity of p_1, p_2 and p_3 exists and can be computed in finite time.*

Given three robots r_1, r_2, r_3, we distinguish three main cases to solve the

GATHERING PROBLEM (refer to Figure 5.a–c): If the robots are on a line, we move the two outer robots towards the median robot. If the triangle $\triangle(r_1, r_2, r_3)$ contains an angle of at least 120° in vertex r_i, then we move the other two robots towards r_i. Otherwise, the triangle $\triangle(r_1, r_2, r_3)$ contains no angle greater than or equal to 120°. In this case, we move all robots towards c_e, with c_e the center of equiangularity of r_1, r_2 and r_3. This is shown in Algorithm 1.

For the correctness of Algorithm 1, observe that the configuration of the first two cases (robots on a line, and triangle with an angle of at least 120°) remains invariant until at least one robot reaches its destination (the median robot in the case of Line 3, and r_i in the case of Line 6). Then there is a unique point with strict multiplicity, and all robots gather at this point. In the third case, the center of equiangularity does not change until one or more robots reach c_e. If more than one robot reach c_e at the same time, then we have a point with strict multiplicity. If only one robot reaches c_e, then we are in the second case (angle greater than or equal to 120°), with r_i being the robot in c_e. Thus, the other two robots will continue moving towards c_e.

Algorithm 1 Gathering for $n = 3$

```
    If There Is One Point $q$ With Multiplicity $> 1$ Then move_to($q$).
    If All 3 Robots Are On A Line Then
        $r :=$ The Median Robot On The Line;
        move_to($r$).
5:  If $\exists i \in \{1,2,3\} | \sphericalangle(x, r_i, y) \geq 120°$, With $x, y \neq r_i$ The Other Two Robots Then
        move_to($r_i$).
    $c_e :=$ Center Of Equiangularity Of $r_1, r_2, r_3$;
    move_to($c_e$).
```

Result 2 *Three robots that can detect multiplicity can always gather at a point in a finite number of cycles.*

3.3 Four Robots

Given four robots $r_1, \ldots, r_4$ in the plane, the only possible configurations are depicted in Figure 5.d–g. Algorithm 2 shows how to take distinct actions according to these initial configuration of the robots, similarly to the case of three robots. Observe that in the last case, if one robot reaches point q (computed in Line 14), we end up in the configuration depicted in Figure 5.e, and q is the position of the median robot.

Result 3 *Four robots that can detect multiplicity can always gather at a point in a finite number of cycles.*

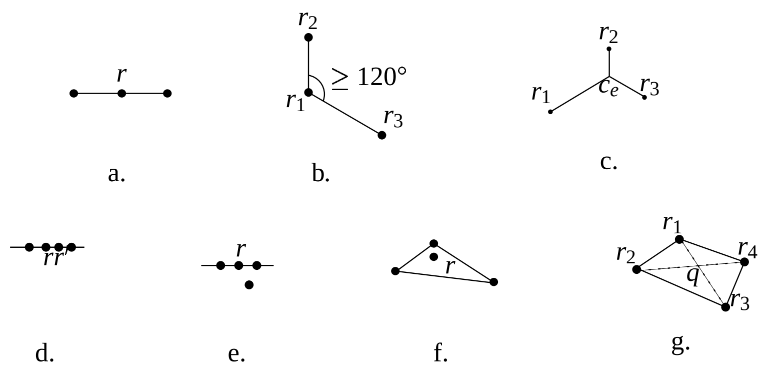

Figure 5: (a)–(c) Configurations of 3 points: (a) all on a line; (b) one angle in $\triangle(r_1, r_2, r_3)$ greater than or equal to 120°; (c) center of equiangularity c_e inside $\triangle(r_1, r_2, r_3)$. (d)–(g) Configurations of 4 points: (d) all on a line; (e) three points on a line; (f) one point inside the convex hull; (g) all on the convex hull.

4 The Gathering Problem for $n \geq 5$

Solving the GATHERING PROBLEM for more than four robots seems to be a hard problem: Assume that the initial configuration of n robots is a regular n-gon. In this case, the configuration is totally symmetric, and no subset of the robots can be deterministically elected to move. Thus, all robots will be allowed to move. An n-gon is a special case of a biangular configuration with equal angles $\alpha = \beta$ and equal distances from the center of biangularity c. Since the center c of biangularity is invariant under movement towards c, moving all robots towards c solves the GATHERING PROBLEM if the initial configuration is biangular (and if it is an n-gon). Hence, we have a strategy that solves the problem if the initial configuration is biangular.

In the following, we focus on presenting an algorithm (Algorithm 3) which solves the GATHERING PROBLEM for more than four robots if the initial configuration is not biangular. The sections concludes with a discussion why Algorithm 3 in combination with the strategy given above does not solve the entire GATHERING PROBLEM for $n \geq 5$.

4.1 Definitions

Smallest Enclosing Circle. Given n distinct points in the plane, we will denote by *SEC* the *smallest enclosing circle* of the points. This circle passes either through two of the points that are on the same diameter (opposite points), or through at least three of the points. The smallest enclosing circle of a set of n

Algorithm 2 Gathering for $n = 4$

```
If There Is One Point $q$ With Multiplicity $> 1$ Then move_to($q$).
If All 4 Robots Are On A Line Then
    $r, r'$ := The Two Median Robots On The Line;
    $c$ := Center Of The Two Outer Robots On The Line;
    If I Am $r$ or $r'$ Then move_to($c$) Else do_nothing().
If Three Robots Are On A Line Then
    $r$ := The Median Robot On The Line;
    move_to($r$).
$CH$ := Convex Hull Of $\{r_1, r_2, r_3, r_4\}$;
If One Robot Is Strictly Inside $CH$ Then
    $r$ := Robot Inside $CH$;
    move_to($r$).
If No Robot Is Strictly Inside $CH$ Then
    $q$ := Intersection Point Of The Two Diagonals Of $CH$;
    move_to($q$).
```

points is unique and can be computed in $O(n \log n)$ time ([12]).

Lemma 2 *Let SEC be the smallest circle enclosing n distinct points in the plane, and let c be its center. Let $l_1, \ldots, l_k$ be the points on the circumference of SEC (w.l.o.g they are in clockwise order). If $k \geq 4$, then there exists a point l_i, $1 \leq i \leq k$, such that $\sphericalangle(l_{i-1}, c, l_{i+1}) \leq 180°$ (all operations are modulo k), and SEC does not change by eliminating l_i from the set of points.*

String of Angles. Given n distinct points $p_1, \ldots, p_n$ in the plane, let SEC be the smallest enclosing circle of the points, and c be its center. For an arbitrary point p_k, $1 \leq k \leq n$, and an orientation O (clockwise or counterclockwise), we define the *string of angles* $SA(p_k, O)$ by the following algorithm (refer to Figure 6 for a pictorial representation of the definitions related to SA):

```
Compute_SA($p_k, O$)
    $p := p_k, i := 1$;
    While $i \neq n+1$ Do
        $p'$ := Succ($p, O$);
        $SA[i] := \sphericalangle_O(p, c, p')$;
        $p := p'$; $i := i+1$;
    End While
```

The successor of p, computed by Succ(p, O), is (refer to Figure 7)

- either the point $p_i \neq p$ on $[c, p)$ such that $dist(c, p_i)$ is minimal among all points $p_j \neq p$ on $[c, p)$ with $dist(c, p_j) > dist(c, p)$), if such a point exists; or

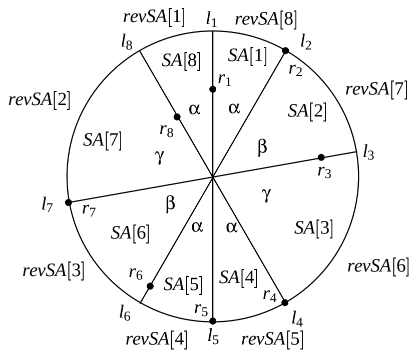

Figure 6: Example of the string of angles computed by `Compute_SA(`r_1, O`)`, with a clockwise orientation of *SEC*. With $\alpha = 25°, \beta = 60°$, and $\gamma = 70°$, we have $SA(r_1, O) =< \alpha, \beta, \gamma, \alpha, \alpha, \beta, \gamma, \alpha >=< 25°, 60°, 70°, 25°, 25°, 60°, 70°, 25° >$; $LexMinString =< \alpha, \alpha, \beta, \gamma, \alpha, \alpha, \beta, \gamma >$; $StartSet = \{4, 8\}$, and $revStartSet = \emptyset$.

- the point $p_i \neq p$ such that, according to orientation O, there is no other point inside $sector(\sphericalangle_O(p, c, p_i))$, and there is no other point on the line segment $[c, p_i]$.

Instead of $SA(p_k, O)$, we write $SA(p_k)$ if we do not refer to a specific orientation, and SA if we do not consider specific p_k and O. Given p_k and O, procedure `Succ()` defines a unique successor, and thus a unique string of angles. Given two starting points p_k and p_i, then $SA(p_k, O)$ is a cyclic shift of $SA(p_i, O)$. We associate the i-th angle in $SA(p_k, O)$ with its defining point, i.e., if $SA[i] = \sphericalangle_O(p, c, p')$, then we say that $SA[i]$ is *associated* with p. We say that SA is *general* if it does not contain any zeros; otherwise, at least two points are on a line starting in c, and we call the string of angles *degenerated*.

We define the *reverse string of angles revSA* in an analogous way, i.e., as the string of angles according to the opposite orientation of *SA*. Let *LexMinString* be the lexicographically minimal string among all strings of angles (in both orientations), i.e., $LexMinString := min(\{SA(p_i) \mid 1 \leq i \leq n\} \cup \{revSA(p_i) \mid 1 \leq i \leq n\})$. Let *StartSet* be the set of all indices where *LexMinString* starts, i.e., $StartSet := \{i | 1 \leq i \leq n, SA(p_i) = LexMinString\}$, and let *revStartSet* be the set of all indices where *LexMinString* starts with the opposite orientation.

4.2 Algorithm Sketch

In Algorithm 3, we sketch an algorithm that solves the GATHERING PROBLEM for $n \geq 5$ robots if the initial configuration is not biangular. The complete algorithm as well as the proof of correctness can be found in [9]. The algorithm guarantees

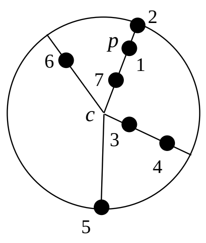

Figure 7: Routine $\texttt{Succ(}p,O\texttt{)}$ in $\texttt{Compute\_SA()}$. The circle is oriented clockwise. The points are numbered according to routine $\texttt{Succ()}$; that is $\texttt{Succ(}p,O\texttt{)}$=2, $\texttt{Succ(}2,O\texttt{)}$=3, and so on.

that the smallest enclosing circle *SEC* does not change as long as there is no strict multiplicity. Moreover, *SA* remains invariant unless a strict multiplicity is achieved, or a single robot reaches *c*, or *SA* becomes degenerated. Since the initial configuration is not biangular (and thus not totally symmetric), a strict subset of the robots can be elected which is allowed to move. During the movement, the occurrence of any totally symmetric configuration is avoided. The three cases in Lines 17–19 are sketched in the following.

Case OneStartingIndex (Line 17 of Algor. 3). If $|StartSet \cup revStartSet| = 1$, then there exists exactly one starting index j and orientation O for $LexMinString$. If some robots are inside *SEC*, we move them towards *c*. If all robots are on the circumference of *SEC*, we use j and O to elect a unique robot *r* which moves towards *c* without changing *SEC* (*r* is chosen according to Lemma 2).

Case TwoStartingIndices (Line 18 of Algor. 3). If $|StartSet \cup revStartSet| = 2$, then there exists for each orientation exactly one robot r (resp. r_{rev}) such that $LexMinString$ starts in the associated index of *SA* (resp. *revSA*). We elect two neighbors *x* and *y* of *r* and r_{rev} in a unique way, such that *SEC* does not change when *x*, *y*, or both of them move towards *c* (refer to Figure 8). If no robot or no other robot except *x* and *y* is inside *SEC*, we move *x* and *y* “carefully” towards *c*, which means that none of the two is allowed to enter *c* unless the other robot is already inside *SEC*. Otherwise, there is at least one other robot inside *SEC*, and we simply move all robots which are inside *SEC* towards *c*.

Case ManyStartingIndices (Line 19 of Algor. 3). If $|StartSet \cup revStartSet| > 2$, then *SA* (and *revSA* as well) is periodic, i.e., there exist $l \geq 2$ indices $i_1, \ldots, i_l$ such that $SA(p_{i_j}) = SA(p_{i_k})$ for all $1 \leq j,k \leq l$. The string of angles can be partitioned into l equal blocks. Roughly speaking, we elect one or two robots per block which move towards *c*. We have to ensure that *SEC* remains invariant, even

Algorithm 3 Gathering for $n \geq 5$ (initial configuration not biangular)

```
If There Is One Point $q$ With Multiplicity $> 1$ Then move_to($q$).
SEC := Smallest Enclosing Circle Of All Robots;
$c$ := Center Of SEC;
If One Robot $r$ Is At Point $c$ Then
    If No Other Robot Is Inside SEC Then
        $q$ := Position Of An Arbitrary Robot On SEC;
        If I Am $r$ Then move_to($q$) Else do_nothing().
    Else % Some Other Robot Is Inside SEC Besides $r$ %
        Moving := {Robots Inside SEC};
        If I Am In Moving Then move_to($c$) Else do_nothing().
Else % No Robot Is At $c$ %
    $r$ := An Arbitrary Robot;
    $O$ := An Arbitrary Orientation on SEC;
    SA := String Of Angles w.r.t. $r$ And $O$;
    If SA Is General Then
        StartSet, revStartSet := Indices Where Lex. Minimal String Starts;
        If $|StartSet \cup revStartSet| = 1$ Then Case OneStartingIndex.
        If $|StartSet \cup revStartSet| = 2$ Then Case TwoStartingIndices.
        If $|StartSet \cup revStartSet| > 2$ Then Case ManyStartingIndices.
    Else %SA Is Degenerated%
        If Only One Robot $r$ Is Inside SEC Then
            $r'$ := Next Robot On Same Radius $[c, r)$;
            If I Am $r$ Then move_to($r'$) Else do_nothing().
        Else % More Than One Robot Are Inside SEC %
            If I Am Inside SEC Then move_to($c$).
            Else do_nothing().
```

if all these robots move together. It can be proven that such an election always exists if the configuration is not biangular. If only elected robots are inside *SEC*, we move all elected robots towards c. Again, the movement is done “carefully”, i.e., no robot is allowed to reach c unless all other elected robots are already inside *SEC*. If there are other robots inside *SEC*, then we simply move all robots inside *SEC* towards c.

Result 4 *Five or more robots that can detect multiplicity and that are not in a biangular configuration at the beginning can always gather at a point in a finite number of cycles.*

Algorithm 3 solves the GATHERING PROBLEM for more than four robots if the initial configuration is not biangular. For initial biangular configurations, we can gather the robots at the center of biangularity (as pointed out at the beginning of this section). Hence, we have two algorithms which together cover all possible

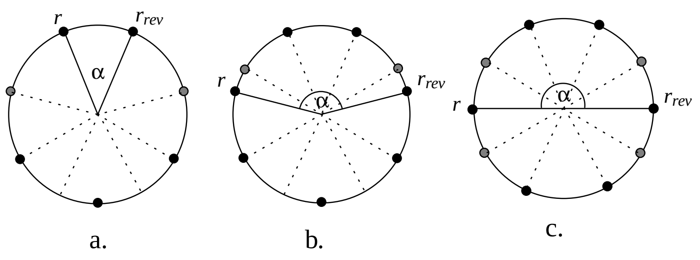

Figure 8: Algorithm 3, Case $|StartSet \cup revStartSet| = 2$. The grey dots represent the robots adjacent to r and r_{rev} that are allowed to move.

initial configurations. However, these two algorithms cannot be simply combined to solve the entire GATHERING PROBLEM: let the robots' initial configuration be non–biangular. Hence, they start executing Algorithm 3. During the run of the algorithm, it may happen that the robots form a biangular configuration. Since the robots act completely asynchronously, some of them may observe this biangular configuration, while others do not. Thus, some of the robots move to the center c of biangularity, while others still perform Algorithm 3. Hence, the gathering could never happen.

5 Conclusions

We presented a deterministic and oblivious algorithm for the GATHERING PROBLEM for $n \geq 5$ robots that works if the robots can detect multiplicities, and if the initial configuration is not biangular. Instead of multiplicity detection, other additional abilities may be considered in future work. For instance, it would be interesting to explore the relationship between memory and solvability of the assigned tasks, or to study the presence of some kind of communication among the robots.

We strongly believe that the restriction of the algorithm to non–biangular initial configurations can be relaxed by excluding totally symmetric initial configurations (all robots on a circle *and* in biangular position). Moreover, the algorithm might be modified such that, starting from a non–biangular configuration, the robots never form a biangular configuration during the run of the algorithm. However, the remaining challenge is to design an algorithm which solves the GATHERING PROBLEM with $n \geq 5$ robots for *any* initial configuration. This seems to be a hard problem, since starting from a totally symmetric configuration, all robots will be allowed to move, and an algorithm can hardly control which configurations occur during their movement. Therefore, we conjecture that the GATHERING PROBLEM is unsolvable in general.

Acknowledgements

We would like to thank Stephan Eidenbenz, Paola Flocchini, Vincenzo Gervasi, Zsuzsanna Lipták, Nicola Santoro, Konrad Schlude, and Peter Widmayer for the helpful discussions and comments.

References

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. *IEEE Transaction on Robotics and Automation*, 15(5):818–828, 1999.

[2] T. Balch and R. C. Arkin. Behavior-based Formation Control for Multi-robot Teams. *IEEE Transaction on Robotics and Automation*, 14(6), December 1998.

[3] E. J. Cockayne and Z. A. Melzak. Euclidean Constructibility in Graph-minimization Problems. *Mathematical Magazine*, 42:206–208, 1969.

[4] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots. In *ISAAC '99*, volume LNCS 1741, pages 93–102, 1999.

[5] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of Autonomous Mobile Robots With Limited Visibility. In *STACS 2001*, volume LNCS 2010, pages 247–258, 2001.

[6] D. Jung, G. Cheng, and A. Zelinsky. Experiments in Realising Cooperation between Autonomous Mobile Robots. In *ISER*, 1997.

[7] M. J. Matarić. Designing Emergent Behaviors: From Local Interactions to Collective Intelligence. In *From Animals to Animats 2: Int. Conf. on Simulation of Adaptive Behavior*, pages 423–441. The MIT Press, 1993.

[8] G. Prencipe. CORDA: Distributed Coordination of a Set of Autonomous Mobile Robots. In *ERSADS 2001*, pages 185–190, 2001.

[9] G. Prencipe. *Distributed Coordination of a Set of Autonomous Mobile Robots*. PhD thesis, Università di Pisa, 2002. http://sbrinz.di.unipi.it/˜peppe/tesi.ps.

[10] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *Siam Journal of Computing*, 28(4):1347–1363, 1999.

[11] E. Weiszfeld. Sur le Point Pour Lequel la Somme Des Distances de n Points Donnés Est Minimum. *Tohoku Mathematical*, 43:355–386, 1936.

[12] E. Welzl. Smallest Enclosing Disks (Balls and Ellipsoids). *Lecture Notes in Computer Science*, 555:359–370, 1991.

Mark Cieliebak is a Ph.D. student at the Computer Science Department of ETH Zurich. E-mail: `cielieba@inf.ethz.ch`

Giuseppe Prencipe is a research fellow at the Computer Science Department of the University of Pisa. E-mail: `prencipe@di.unipi.it`