The Minimum Algorithm Size of k-Grouping by Silent Oblivious Robots

Paola Flocchini[1][0000−0003−3584−5727], Debasish Pattanayak[2][0000−0003−2862−2795], Nicola Santoro[2][0000−0002−7954−3918], and Masafumi Yamashita[3]

[1] University of Ottawa, Canada
[2] Carleton University, Canada
[3] Kyushu University, Japan

Abstract. Consider a set of mobile computational elements, called robots, that are viewed as points, and operate in the Euclidean plane in synchronous rounds. The robots are oblivious (they forget all computations performed in previous rounds), silent (unable of direct communication), and anonymous (indistinguishable from the outside). Each robot is provided with a private coordinate system, and can determine the position of the other robots (performing a Look operation); it has an algorithm, which it executes (performing a Compute operation) to determine a destination point; and it can move towards the destination (performing a Move operation).
The k-`Grouping` problem requires the robots, starting from an arbitrary initial configuration in the plane, to gather at k distinct locations, not chosen in advance, by performing Look-Compute-Move cycles, and no longer move. This simple problem is however unsolvable if all the robots execute the same algorithm. It has been recently shown that, were different subgroups of the robots to execute different algorithms, the problem remains still unsolvable if the number of the algorithms is less than k.
In this paper we prove that this number is minimum: we design k distinct algorithms and prove that, if each is executed by an arbitrary non-empty subset of the robots, they will collectively be able to solve the problem; furthermore, they are able to do so under a weak assumption on the level of agreement among the local coordinate systems. We further prove that, without any agreement, the problem becomes unsolvable even with $k+1$ different algorithms. However, if an unbounded number of algorithms are allowed such that each robot has a unique algorithm, then we can solve k-`Grouping` without any agreement.

Keywords: Autonomous mobile robot, Minimum algorithm size, Scattering, Gathering, Pattern formation.

1 Introduction

1.1 Framework, Background, and Motivation

Consider a collection $\mathcal{R}$ of mobile computational entities, called *robots*, viewed as points, operating in synchronous rounds in the Euclidean plane. Time is di-

vided into discrete intervals, called *rounds*; in each round a non-empty subset of robots is activated. All activated robots perform a *Look-Compute-Move* (LCM) cycle, consisting of three phases, in perfect synchrony: in the *Look* phase, each robot takes a snapshot of the environment, returning the positions of the other robots in its own local coordinate system; in the *Compute* phase, it executes its pre-programmed algorithm to compute a destination point; in the *Move* phase, it moves towards the destination. The decision of which robots are activated in a given round is under the control of an adversarial *scheduler*. Under the weakest adversarial scheduler, called *fully-synchronous* ($\mathcal{FSYNC}$), the entire set of robots is activated in every round. Under the strong adversary, called *semi-synchronous* ($\mathcal{SSYNC}$), the activated subset is arbitrary (provided that every robot is activated infinitely often).

Since its introduction in the distributed computing community [16], this setting has been extensively investigated, and the primary direction of research has been on determining the minimal assumptions on the robots' capabilities under which they are collectively capable to solve a given problem or to successfully perform a given task.

In the weakest (de-facto standard) model, $\mathcal{OBLOT}$, the robots are *oblivious* (i.e., have no memory of previous cycles), *silent* (i.e., have no means of direct communication), and *disoriented* (i.e., do not necessarily agree on a common coordinate system). They are also generally assumed to be *anonymous* (i.e. do not have distinct identifiers or visible markers) and *homogeneous* (i.e., execute the same algorithm – compute the same function) [**?**].

Within this model, the research has been focused on the computability and complexity aspects of basic fundamental tasks, in particular the class of `Pattern Formation` problems, requiring the robots to move from the configuration of their initial (arbitrary) positions in the plane to one congruent with a geometric shape (the pattern) given in input (e.g., see [1,4,5,6,9,11,10,16,17], as well as [13] and chapters therein). An extensive amount of research has been dedicated on identifying the impact that specific factors have on the solvability of these basic problems. These are, for instance, the studies on the impact of capabilities such as multiplicity detection (i.e. ability in the *Look* phase to determine the number of robots at the same point at the same time), some agreement on the coordinate system (e.g., compass, chirality, ...), rigidity (e.g., ability in the *Move* phase to always reach the computed destination), etc.

Because of the stringent limitations imposed by the model on the computational power of the robots, several problems remain unsolvable even in presence of some of these additional capabilities. For example, it is impossible to form an asymmetric pattern starting from a symmetric initial configuration, even if the robots are rigid, have chirality and multiplicity detection, and the scheduler is fully synchronous [16].

Consider the case where there is no single algorithm A that would allow the robots to solve a given problem $\mathcal{P}$ under a given set of conditions. It might be however possible to solve the problem by designing two (or more) distinct algorithms, say A_1 and A_2, programming some robots $\mathcal{R}_1 \subset \mathcal{R}$ with A_1 while

the others $\mathcal{R}_2 = \mathcal{R} \setminus \mathcal{R}_1$ with A_2, and having each robot execute its algorithm unaware of the composition of the two sets.

This observation, recently made in [2], leads to the following interesting research question:

> *Given a problem $\mathcal{P}$ in a given setting, what is the minimum number of distinct algorithms that would allow a set $\mathcal{R}$ of robots to solve it?*

We shall denote such a measure, called the *minimum algorithm size* in [2], by $\Phi(\mathcal{P})$, and investigate it for a basic family of `Pattern Formation` problem.

1.2 Problem and Contributions

The class of problems we are investigating, called k-`Grouping`, requires the robots to gather within finite time, at k distinct points, not chosen in advance, and no longer move; this has to be done regardless of the number of robots (provided $k \leq |\mathcal{R}|$) and of their initial configuration $C(0)$ (i.e., location of the robots at time $t = 0$).

In other words, for every $\mathcal{R}$ with $|\mathcal{R}| \geq k$, any solution algorithm must satisfy the following temporal geometric predicate in every possible execution:

$$k\text{-}\texttt{Grouping} \equiv \{\exists \hat{t} \geq 0 \ \{(|C(\hat{t})| = k) \textbf{ and } (\forall t' \geq \hat{t}, \ (C(t') = C(\hat{t}))\}\}$$

where $|C(t)|$ denotes the number of distinct points occupied by the robots at time t.

Observe that the special case `1-Grouping` is precisely one of the most important and investigated problem in the field, called `Gathering` (or `Rendezvous`) (e.g., see [1,3,8,12,14,15]). Another special case of k-`Grouping` is the subclass of the `Scattering` family of problems, called k-`Scattering`, which assumes that, in the initial configuration $C(0)$, all robots are co-located in the same point (e.g., see [2,7]).

An interesting aspect of k-`Grouping` is that, in spite of its apparent simplicity, it is generally *unsolvable* in $\mathcal{OBLOT}$ (i.e., with a single algorithm). This impossibility is very easy to verify; e.g., consider forming two groups starting from an initial configuration $C(0)$ where all robots are co-located in the same point. Furthermore, this impossibility holds true even if the robots are very powerful and the adversary very weak; in particular, it holds also when the robots share a common orientation and direction of the axes (i.e., they have a compass), have strong multiplicity detection, movements are rigid, and the scheduler is fully synchronous $\mathcal{FSYNC}$.

Hence the need to determine $\Phi(k\text{-}\texttt{Grouping})$, i.e., the *minimum algorithm size* of k-`Grouping`. A recent result of [2] implies that

$$\Phi(k\text{-}\texttt{Grouping}) \geq k$$

even if the robots are very powerful and the adversary is very weak (see above).

The following questions thus naturally arise: is this bound tight? can it be achieved under less stringent conditions (i.e., with weaker robots and under a stronger adversary)?

In this paper, we first of all provide an affirmative answer to both questions. We prove that the lower-bound is tight, and indeed k distinct algorithms are sufficient; the proof is constructive. This result holds even if the robots agree only on the direction of a single axis, have no multiplicity detection, movements are non-rigid (i.e., they can be stopped in their movement) and the adversarial scheduler is the stronger $\mathcal{SSYNC}$.

We then investigate whether such a result can be achieved in an even weaker setting: specifically, without any agreement among the local coordinate systems. We prove that the answer is negative: in this case, k-`Grouping` is unsolvable even if the robots had $k+1$ algorithms, and the other conditions were the best possible (strong multiplicity detection, rigid movements, and fully synchronous scheduler).

Finally we show that the problem is always solvable, even without any agreement among the local coordinate systems, if the number of algorithms is unbounded in number; indeed it suffices with a distinct algorithm for each robot.

See Table 1 for a summary of the results.

k-`Grouping`	**Algorithms**	**Axis**	**Rigidity**	**Scheduler**	**Reference**
unsolvable	$< k$	2	Yes	$\mathcal{FSYNC}$	[2]
solvable	k	1	No	$\mathcal{SSYNC}$	this paper
unsolvable	$k+1$	No	Yes	$\mathcal{FSYNC}$	this paper
solvable	$\lvert\mathcal{R}\rvert$	No	No	$\mathcal{SSYNC}$	this paper

Table 1. Summary of results

1.3 Organization

The paper is organized as follows. In Section 2, we describe the model under consideration. In Section 3.1, we show that the bound on the number of algorithms is tight with partial agreement on axes, and in Section 3.2 show impossibility without agreement even with a collective algorithm of size $k+1$. Further, in Section 3.2, we show a possibility if the algorithm size is unbounded before concluding in Section 4. Due to space limitations, some proofs and details are omitted.

2 Model and Terminology

The system consists of a set $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ of mobile computational entities, called robots, modeled as geometric points, that operate in $\mathbb{R}^2$. The robots

are autonomous without a central control. Each robot has its own local coordinate system, and is equipped with devices that allow it to observe the positions of the other robots in its local coordinate system.

The robots operate in *Look-Compute-Move* (LCM) cycles. When activated, a robot executes a cycle by performing the following three operations:

1. *Look:* The robot obtains a snapshot of the positions occupied by robots expressed with respect to its own coordinate system; this operation is assumed to be instantaneous.
2. *Compute:* The robot executes the algorithm using the snapshot as input; the result of the computation is a destination point.
3. *Move:* The robot moves towards the computed destination. If the destination is the current location, the robot stays still.

The robots are *silent*: they have no explicit means of communication; furthermore they are *oblivious*: at the start of a cycle, a robot has no memory of observations and computations performed in previous cycles.

The system is *synchronous*; that is, time is divided into discrete intervals, called *rounds*. In each round a robot is either active or inactive. The robots active in a round perform their LCM cycle in perfect synchronization; if not active, the robot is idle in that round. All robots are initially idle.

The decision of which robots are activated in a given round is under the control of an adversarial *scheduler*. The weakest adversarial scheduler, called *fully-synchronous* ($\mathcal{FSYNC}$), activates the entire set of robots in every round. Under the strong adversary, called *semi-synchronous* ($\mathcal{SSYNC}$), the activated subset is arbitrary; however, every robot is activated infinitely often. In the following, we use round and time interchangeably.

Let $r_i(t)$ denote the location of robot r_i at time t in some global coordinate system (possibly unknown to the robots); let $C(t) = \{p_1, p_2, \ldots, p_n\}$, called *configuration* at time t, be the multi-set of robot positions $p_i = r_i(t)$, and let $Q = \{q_1, q_2, \ldots, q_m\}$ be the corresponding set of points occupied by the robots. Observe that $|Q| = m \leq n$ since some robots might be at the same locations, called multiplicity points. The robots are said to have strong/weak/no *multiplicity detection* capability if they are able to detect in a given point the exact number of robots, only whether or not there is more than one robot, or have no such a capability, respectively

Movements are said to be *rigid* if the robots always reach their destination. They are said to be *non-rigid* if they may be unpredictably stopped by an adversary whose only limitation is the existence of $\delta > 0$, unknown to the robots, such that if the destination is at distance at most δ the robot will reach it, else it will move at least δ towards the destination.

As for their private coordinate systems, the robots are said: to have *complete agreement* if they all share the same orientation and direction of both axes; to have *one-axis agreement* if they all share the orientation and direction of just one axes (say the Y axis); to have *no agreement* (or to be disoriented) if they might disagree on the orientation and direction of the axes. In all cases, there might to be agreement on the unit of distance.

As discussed in Section 1.2, the class of k-`Grouping` problems requires the robots to gather within finite time, at k distinct points, not chosen in advance, and no longer move; this has to be done regardless of the number of robots (provided $k \leq |\mathcal{R}|$) and of their initial configuration $C(0)$ (i.e., location of the robots at time $t = 0$).

A *collective solution algorithm* for k-`Grouping` of *size* $s \geq k$ is any set $\mathcal{A} = \{A_1, ..., A_s\}$ of $s \geq k$ distinct algorithms such that: for any set of robots $\mathcal{R}$ with $|\mathcal{R}| \geq k$ and any partition $P(\mathcal{R}) =< R_1, ..., R_s >$ of $\mathcal{R}$ in s non-empty subsets, if each element of R_i is programmed with and executes A_i, then, without knowing which algorithm is being executed by the other robots and starting from any initial configuration, within finite time all robots gather at k distinct points, and no longer change position.

The smallest value of s among all collective solution algorithms is called the *minimum algorithm size* and denoted by $\Phi(k$-`Grouping`$)$.

3 k-`Grouping`

In this section, first, we determine $\Phi(k$-`Grouping`$)$ constructively by providing a set of k algorithms that solve k-`Grouping` when the robots have agreement on the orientation of one axis. Then we investigate the solvability of k-`Grouping` in absence of any agreement on the coordinate system; we prove that, without any agreement on the coordinate system, no set of $k + 1$ algorithms can solve k-`Grouping`. On the other hand, the problem becomes solvable if every robot has a different algorithm.

3.1 Partial Agreement.

First, we consider robots with agreement in the orientation of one-axis. They may not agree on the chirality. In other words, without loss of generality all robots agree on the direction of positive y-axis, but they may disagree on the direction of positive x-axis. The algorithm presented here is a general strategy that works for any $k \geq 2$. For $k = 1$, we refer the readers to the paper by Bhagat et al. [3], where they present an algorithm for 1-`grouping` (also known as `Gathering`) of robots with one-axis agreement.

General Strategy.

At a high level, the proposed collective algorithm $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$, for $k > 1$, has each robot determine in its Look phase the *smallest enclosing rectangle* (SER) of the observed configuration, where the sides of the rectangle are parallel to the axes, and use it as an input in the execution of its algorithm.

Informally, the general strategy of the collective algorithm $\mathcal{A}$ consists on having all the robots that execute the same algorithm A_i to eventually move to the same line L_i perpendicular to the y axis, and eventually gather at a single point M_i on L_i. The location of the destination lines is totally dynamic and may change several times during the executions of the robots' algorithms. Eventually,

as we will show, the top line L_1 and the bottom line L_k will no longer change, so neither will the height of the SER. Once this occurs (let us stress that the robots might be unaware of this fact), also the location of line L_i $(1 < i < k)$ is determined as being at distance $d_i = (i-1)h/(k-1)$ from L_1, where h is the height of the SER, and no longer change. Then the robots executing algorithms A_i eventually form a point M_i on L_i at distance d_i from M_1.

The algorithm terminates once the observed configuration is composed of k equidistant points on a vertical axis.

In what follows, we describe the algorithm in detail; the pseudocode is presented in Algorithm 1.

Basic Concepts and Terminology.

Before starting the description, let us introduce some important concepts and terminology. In the following (including the algorithm description) all the geometric objects are expressed in terms of the local coordinate system of the observing robot. Let us stress that, since we are considering the weak setting where the local coordinate systems might disagree on the direction of the x-axis, the notion of "left" and "right" of this robot is possibly opposite that of other robots.

Consider the configuration $C(t)$ observed by a robot active at time t; let $Q = \{q_1, ..., q_m\}$ be the distinct positions occupied by the robots in that configuration, where $q_j = (x_j, y_j)$ $(1 \leq j \leq m)$.

Given Q, let $l_x = \min_j(x_j)$, $r_x = \max_j(x_j)$, $b_y = \min_j(y_j)$, and $t_y = \max_j(y_j)$, where $1 \leq j \leq m \leq n$. Then the smallest enclosing rectangle $SER(Q)$ is uniquely determined by the four corners (l_x, b_y), (l_x, t_y), (r_x, t_y), and (r_x, b_y) (ref. Fig. 1).

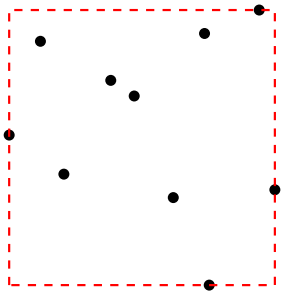

Fig. 1. Smallest Enclosing Rectangle of a configuration $C(t)$

The rectangle $SER(Q)$ is said to be *improper* if it is a horizontal line or a point (i.e., if $t_y = b_y$), *proper* otherwise. Let h be the height of a proper $SER(Q)$, i.e., $h = t_y - b_y$.

In a proper $SER(Q)$, the parallel lines L_i $(1 \leq i \leq k)$ are defined as follows: L_1 is the line passing through (l_x, t_y) and (r_x, t_y); L_k is the line passing through (l_x, b_y) and (r_x, b_y); L_j $(1 < j < k)$ is the line passing through (l_x, l_j) and (r_x, l_j), where $l_j = ((j-1)b_y + (k-j)t_y)/(k-1)$. Notice that the distance between two successive lines is $h/(k-1)$. Let M_1 be the midpoint of the segment of L_1 between (l_x, t_y) and (r_x, t_y).

The *final configuration* C_f is the configuration where all the robots are located on a line parallel to the y-axis, $|Q| = k$, and the distance between consecutive points in Q is $h/(k-1)$.

Algorithm 1: k-grouping

Input: k, set of points Q, robot position (p_x, p_y)
Output: Destination of the robot

Algorithm A_1**:**
  if $Q = C_f$ **then**
    destination $\leftarrow (p_x, p_y)$
    terminate
  else
    compute $SER(Q)$
    if $t_y = b_y$ **then**
      destination $\leftarrow (p_x, p_y + 1)$
    else
      if *if there is only one robot position* (q_x, q_y) *on* L_1 **then**
        destination $\leftarrow (q_x, q_y)$
      else
        if $p_y = t_y$ **then**
          destination $\leftarrow ((l_x + r_x)/2, p_y)$
        else
          destination $\leftarrow (p_x, t_y)$

Algorithm A_i**: where** $1 < i \leq k$
  if $Q = C_f$ **then**
    destination $\leftarrow (p_x, p_y)$
    terminate
  else
    compute $SER(Q)$
    if $t_y = b_y$ **then**
      destination $\leftarrow (p_x, p_y)$
    else
      Let $l_i = ((i-1)b_y + (k-i)t_y)/(k-1)$
      if *if there is only one robot position* (q_x, q_y) *on* L_1 **then**
        if $p_y = m_i$ **then**
          destination $\leftarrow (q_x, p_y)$
        else
          destination $\leftarrow (p_x, l_i)$
      else
        destination $\leftarrow (p_x, l_i)$

Detailed Description.

Let us now describe the algorithm in some details. In each round t, a robot activated by the scheduler, in the Look operation, observes the current configuration $C(t)$ as the set of positions Q. Based on this information, the robot, located at (p_x, p_y), proceeds as follows.

(1) If $Q = C_f$, then it does not move, and terminates.

(2) If $Q \neq C_f$, it computes the smallest enclosing rectangle $SER(Q)$ of Q. What happens next depends on whether $SER(Q)$ is proper or not.
(3) Let $SER(Q)$ be proper; then both L_1 and L_k are defined. What does the robot do depends on its algorithm.
   - If the robot is executing A_1 and there is only one point (q_x, q_y) on L_1, then that point becomes its destination M_1 in this round. If instead there is more than one point on L_1, the robot moves to $M_1 = ((l_x + r_x)/2, p_y)$ if it is on L_1, otherwise it moves parallel to y-axis to reach L_1, specifically, to the point (p_x, t_y).
   - If the robot is executing algorithm $A_i, i \neq 1$, first it computes the location of line L_i perpendicular to the y-axis. If the robot is not on L_i, then it moves to L_i by moving parallel to y-axis. If the robot is already located on L_i and there exists exactly one point $q = (q_x, q_y)$ on L_1, then the robot moves to the point (q_x, p_y).
(4) Finally, let $SER(Q)$ be improper; that is, $SER(Q)$ is a horizontal line or a point. In this case, if the active robot is executing A_1, it moves one unit distance[4] towards positive y-axis, otherwise it does nothing.

Correctness.

We now show that our algorithm solves k-`Grouping` in finite time. This is done through a sequence of Lemmas; due to space limitations, these proofs are omitted. Let $C(t_0)$ be the initial configuration. We state the following lemmata with respect to t_0.

Lemma 1. *There exists $t'_0, \geq t_0$, such that smallest enclosed rectangle $SER(Q)$ of $C(t'_0)$ is proper.*

Observe that if $SER(Q)$ is proper, for that configuration all the lines L_i are uniquely defined $1 \leq i \leq k$.

Lemma 2. *There exists $t_1 \geq t_0$ such that L_1 and L_k remain invariant there after.*

Lemma 3. *There exists $t_2 \geq t_1$ such that M_1 remains invariant there after.*

Lemma 4. *There exists $t_3 \geq t_1$ such that all robots executing A_i, for $1 < i \leq k$ remain on L_i there after.*

Lemma 5. *There exists $t_4 \geq \max(t_2, t_3)$ such that all robots executing A_i, for $1 < i \leq k$ remain at M_i there after.*

Theorem 1. *The proposed algorithm solves k-`Grouping` in finite time with one-axis agreement under a $\mathcal{SSYNC}$ scheduler with non-rigid movements.*

[4] The robots do not share a common measure of unit distance; hence, the unit distances of robots executing A_1 may be different from each other.

The finite amount of time elapsed from the start of the algorithm to the reaching of a final configuration can be easily determined. Under the $\mathcal{SSYNC}$ adversarial scheduler, time is measured in terms of *epochs*, where an epoch is a time interval over which all the robots are activated at least once. We have the following corollary.

Corollary 1. *The collective algorithm $\mathcal{A}$ solves k-*`Grouping` *in $O(1)$ epochs under a $\mathcal{SSYNC}$ scheduler if the robots have rigid movement and one-axis agreement, and in $O(h_{max}/\delta)$ epochs when the robots have non-rigid movement, where $h_{max} = \max(t_y - b_y, r_x - l_x)$.*

3.2 No Agreement.

In this section, we show that robots without agreement on axes fail to achieve k-`Grouping` even with $k+1$ distinct algorithms. We also show that it is possible to achieve k-`Grouping` with an unbounded number of algorithms, specifically with each robot executing a different algorithm.

Impossibility of k-`Grouping`.

Let O be the center of the *Smallest Enclosing Circle*(SEC) of a set of point Q. Let θ be the smallest angle such that rotating the set of points Q by θ about O results in the same configuration. We define *rotational symmetry*, $\rho(Q)$ as $2\pi/\theta$.

Theorem 2. *With no agreement on the axes, k-*`Grouping` *is not always solvable under $\mathcal{FSYNC}$ with rigid movements even with $k+1$ distinct algorithms.*

Proof. By contradiction, let there exist, for any n and k, a set of $k+1$ solution algorithms $\mathcal{A} = \{A_1, A_2, \ldots, A_k, A_{k+1}\}$ which, for any initial configuration, allows the n robots to form within finite time a configuration consisting of k points, and no longer move.

Given a configuration $C(t)$, let C_i be the subset of the robots executing algorithm A_i, and let Q_i be the set of points occupied by them.

First, we define a special class $\mathcal{C}_{k+1}$ of configurations, where $n > 2$ and $k \geq 2$. A configuration $C(t)$ belongs to $\mathcal{C}_{k+1}$, when $|C_i| = k+1$ and either (C1) or (C2) holds for each i.

(C1) If $|Q_i| = 1$, then all the robots executing A_i are located at the center of $SEC(Q)$.
(C2) If $|Q_i| > 1$, then $\rho(Q_i) = k+1$, i.e., Q_i forms a regular $k+1$-gon and the center of the polygon is the center of $SEC(Q)$.

Observe that, by definition, n must be a multiple of $k+1$.

Consider now the execution of $\mathcal{A}$ starting from a configuration C in the class $\mathcal{C}_{k+1}$. Let us stress that, since the robots are oblivious, the action of an activated robot r depends only on the observed set of points Q, its location $p = (p_x, p_y) \in Q$, and the algorithm A_i it is executing. Let u be the computed

destination point. Observe that the same action will be performed by all robots executing A_i that are in the same symmetry class of $p = (p_x, p_y)$, denoted as $\rho(p)$. We have the following cases, depending on the computed destination $u \neq p$.

(1) $u = O$: All robots in $\rho(p)$ reach the center of SEC. In fact, since $p \neq O$, by condition $(C2)$, the robots are located at the corners of the regular $k+1$-gon, they have the same view and thus the same destination O.
(2) $p = O$: Q_i becomes a regular $k+1$-gon in $C(t+1)$. In fact, because of condition $(C1)$, all robots executing A_i are located at O. Let the local coordinate system of each robot be rotated by $2\pi/(k+1)$, then every robot moves to a different point at distance d from O, forming a regular $k+1$-gon in $C(t+1)$.
(3) $p \neq O$ and $u \neq O$: A new regular $k+1$-gon Q_i is created. In fact, take the $k+1$-gon formed by the robots executing A_i. Let the local coordinate system of each of those robots be rotated by $2\pi/(k+1)$, then the destinations u and u' of any two robots on consecutive corners p and p' of the original $k+1$-gon will be also at an angle $2\pi/(k+1)$ with each other from the center, since their view were the same. Thus, the resulting Q_i in $C(t+1)$ is again a regular $k+1$-gon.

Observation 1. *The rotational symmetry of the union of concentric regular $k+1$-gons is a multiple of $k+1$.*

Since each of the actions by any of the algorithms from the collection $\mathcal{A}$ results in another configuration with rotational symmetry $j(k+1)$ for some integer $j \geq 0$, the configurations never have a set Q with $|Q| = k$, contradicting the assumed correctness of $\mathcal{A}$.

Unbounded Possibility.

Here we present a collective solution algorithm $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$, that solves k-`Grouping` for n robots under $\mathcal{SSYNC}$ scheduler with non-rigid movement.

Given a configuration $C(t)$ with the corresponding set of points Q, $SEC(Q)$ is the *Smallest enclosing circle* of the set of points. Let γ be the radius, and O be the center of the SEC. We define the *Smallest Enclosing Annulus (SEA)* of the configuration as the area between the SEC and another circle with center O and radius $\gamma/2$ (see Fig. 2).

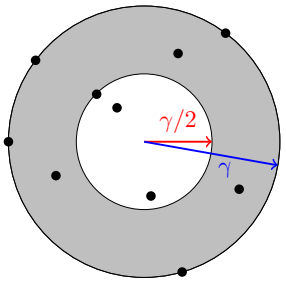

Fig. 2. Smallest Enclosing Annulus of a configuration $C(t)$

At a high level, the collective algorithm aims to form a configuration where the robots executing algorithms $A_1, \ldots, A_{k-1}$ move to the SEC (remain in the SEA, perhaps due to non-rigid movements), while the robots executing algorithms A_i, for $k \leq i \leq n$ move to

Algorithm 2: k-`Grouping with no agreement`

Input: A set of points Q, robot position p, and k
Output: Destination of the robot

Algorithm A_i**, where** $1 \leq i < k$**:**
  if $p \in SEA(Q)$ **then**
    $count \leftarrow |SEA(Q)|$, i.e., the number of robot positions in $SEA(Q)$
    if $count \geq k-1$ **then**
      destination $\leftarrow p$
    else
      $\theta \leftarrow$ smallest incident angle at p
      $\theta_i \leftarrow i\theta/2k$
      Let u be the point on $SEC(Q)$ such that $\angle pOu = \theta_i$
      destination $\leftarrow u$
  else
    $u \leftarrow$ the radial projection of p on $SEC(Q)$
    destination $\leftarrow u$
Algorithm A_i **where** $k \leq i \leq n$**:**
  $O \leftarrow$ center of $SEC(Q)$
  destination $\leftarrow O$

the center of the SEC. This achieves k-`Grouping` by having $k-1$ robot positions on the SEA and one at the center.

The pseudocode for the algorithm is present in Algorithm 2.

4 Conclusion

In this paper, we investigated the problem of k-`Grouping` when robots are allowed to have multiple algorithms. We showed that under partial agreement of axes, such as agreeing on the direction and orientation of one-axis, k different algorithms are sufficient to solve k-`Grouping`, while it is impossible without agreement on the axes even when the robots have $k+1$ algorithms. Further, we show that given $|\mathcal{R}|$ algorithms, it is possible to solve k-`Grouping` even without agreement on axes.

We leave as an conjecture that our algorithms also work under an asynchronous scheduler, possibly even when the movement of robots are non-rigid. We also conjecture that $n-1$ algorithms are insufficient for k-`Grouping` without agreement on axes.

Observe that the problem of k-`Grouping` studied here makes no requirement on the k gathering points except that they are distinct. An interesting open problem is the variation requiring the k distinct points to satisfy a specific geometric pattern, such as a circle or convex hull.

References

1. Agmon, N., Peleg, D.: Fault-Tolerant Gathering Algorithms for Autonomous Mobile Robots. SIAM Journal on Computing **36**(1), 56–82 (Jan 2006). https://doi.org/10.1137/050645221
2. Asahiro, Y., Yamashita, M.: Minimum algorithm sizes for self-stabilizing gathering and related problems of autonomous mobile robots (extended abstract). In: 25th Int. Symp. on Stabilization Safety and Security of Distributed Systems. pp. 312–327 (2023)
3. Bhagat, S., Gan Chaudhuri, S., Mukhopadhyaya, K.: Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. Journal of Discrete Algorithms **36**, 50–62 (Jan 2016). https://doi.org/10.1016/j.jda.2015.10.005
4. Cicerone, S., Di Stefano, G., Navarra, A.: Solving the pattern formation by mobile robots with chirality. IEEE Access **9**, 88177–88204 (2021)
5. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: Gathering. SIAM Journal on Computing **41**(4), 829–879 (2012). https://doi.org/10.1137/100796534
6. Das, S., Flocchini, P., Santoro, N., Yamashita, M.: Forming sequences of geometric patterns with oblivious mobile robots. Distributed Computing **28**(2), 131–145 (2015). https://doi.org/10.1007/s00446-014-0220-9
7. Dieudonné, Y., Petit, F.: Scatter of robots. Parallel Processing Letters **19**(1), 175–184 (2009). https://doi.org/10.1142/S0129626409000146
8. Dieudonné, Y., Petit, F.: Self-stabilizing gathering with strong multiplicity detection. Theoretical Computer Science **428**, 47–57 (2012). https://doi.org/10.1016/j.tcs.2011.12.010
9. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Morgan & Claypool (2012)
10. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Arbitrary pattern formation by asynchronous oblivious robots. Theor. Comput. Sci. **407**, 412–447 (2008)
11. Flocchini, P., Prencipe, P., Santoro, N., Viglietta, G.: Distributed computing by mobile robots: uniform circle formation. Distributed Computing **30**, 413–457 (2017)
12. Flocchini, P.: Gathering. *Chapter 4 of* Distributed Computing by Mobile Entities pp. 63–82 (2019)
13. Flocchini, P., Prencipe, G., Santoro, N. (eds.): Distributed Computing by Mobile Entities, Current Research in Moving and Computing, Lecture Notes in Computer Science, vol. 11340. Springer (2019)
14. Izumi, T., Souissi, S., Katayama, Y., Inuzuka, N., Défago, X., Wada, K., Yamashita, M.: The gathering problem for two oblivious robots with unreliable compasses. SIAM Journal on Computing **41**(1), 26–46 (2012)
15. Pattanayak, D., Mondal, K., Ramesh, H., Mandal, P.: Gathering of mobile robots with weak multiplicity detection in presence of crash-faults. Journal of Parallel and Distributed Computing **123**, 145–155 (Jan 2019). https://doi.org/10.1016/j.jpdc.2018.09.015
16. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing **28**(4), 1347–1363 (1999). https://doi.org/10.1137/S009753979628292X
17. Yamashita, M., Suzuki, I.: Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theoretical Computer Science **411**(26-28), 2433–2453 (2010). https://doi.org/10.1016/j.tcs.2010.01.037