Minimum Algorithm Sizes for Self-stabilizing Gathering and Related Problems of Autonomous Mobile Robots (Extended Abstract)*

Yuichi Asahiro[1]** and Masafumi Yamashita[2]

[1] Kyushu Sangyo University, Fukuoka, Japan (asahiro@is.kyusan-u.ac.jp)
[2] Kyushu University, Fukuoka, Japan (masafumi.yamashita@gmail.com)

Abstract. We investigate swarms of autonomous mobile robots in the Euclidean plane. Each robot has a target function to determine a destination point from the robots' positions. All robots in a swarm conventionally take the same target function. We allow the robots in a swarm to take different target functions, and investigate the effects of the number of distinct target functions on the problem-solving ability. Specifically, we are interested in how many distinct target functions are necessary and sufficient to solve some well-known problems which are not solvable when all robots take the same target function, regarding target function as a resource, like time and message, to solve a problem. The number of distinct target functions necessary and sufficient to solve a problem Π is called the *minimum algorithm size* (MAS) for Π. (The MAS is ∞, if Π is not solvable even for the robots with unique target functions.) We establish the MASs for solving the gathering and related problems from any initial configuration, i.e., in a self-stabilizing manner. Our results include: There is a family of the scattering problems cSCT ($1 \leq c \leq n$) such that the MAS for the cSCAT is c, where n is the size of the swarm. The MAS for the gathering problem is 2. It is 3, for the problem of gathering **all non-faulty** robots at a single point, regardless of the number ($< n$) of crash failures. It is however ∞, for the problem of gathering **all robots** at a single point, in the presence of at most one crash failure.

Keywords: Autonomous mobile robot · Minimum algorithm size · Scattering · Gathering · Pattern formation · Crash failure.

1 Introduction

Swarms of anonymous oblivious mobile robots have been attracting many researchers over four decades, e.g., [1, 3, 9–12, 15, 23, 24, 29, 30]. An anonymous oblivious mobile robot, which is represented by a point in the Euclidean space, looks identical and indistinguishable, lacks identifiers and communication devices, and operates in Look-Compute-Move cycles: When a robot starts a cycle,

* Due to space limitation, some proofs and contributions are deferred to full version [7].
** Corresponding author.

it identifies the multiset of the robots' positions, computes a destination point using a function called *target function* based only on the multiset identified, and then moves to the destination point. All papers listed above investigate swarms, provided that all robots composing a swarm take the same target function. It makes sense: Anonymous robots taking different target functions can behave, as if they had different identifiers. On the other hand, robots with different identifiers can behave, as if they took different target functions, even when they take the same one. The problems investigated cover from simple problems like the convergence and the gathering problems (e.g., [1, 24]) to hard problems like the formation problem of a sequence of patterns and the gathering problem in the presence of Byzantine failures (e.g., [12, 19]). It has turned out that a swarm of anonymous oblivious robots is powerful enough to solve sufficiently hard problems. At the same time, however, we have realized limitation of its problem-solving ability. For example, the gathering problem is, in general, not solvable even if the number of robots is 2 [29].

A promising idea to increase the problem-solving ability of a swarm is to allow robots to take different target functions. It is also natural, since almost all artificial distributed systems enjoy unique identifiers, e.g., serial numbers. This paper takes this approach, and investigates the effects of the number of distinct target functions on the problem-solving ability. Specifically, we are interested in how many distinct target functions are necessary and sufficient to solve some problems which are not solvable when all robots take the same target function.

Let $\mathcal{R}$ and Φ be a swarm of n robots, and a set of target functions such that $|\Phi| \leq n$, respectively. An *assignment* $\mathcal{A} : \mathcal{R} \to \Phi$ of target functions is a surjection from $\mathcal{R}$ to Φ, i.e., every target function is assigned to at least one robot. We call Φ an *algorithm*[3] of $\mathcal{R}$ for a problem Π, if $\mathcal{R}$ solves Π, regardless of the assignment $\mathcal{A}$ that $\mathcal{R}$ takes. (Thus, we cannot assume a particular assignment to design target functions.) The *minimum algorithm size* (MAS) for Π is the size $|\Phi|$ of algorithm Φ necessary and sufficient to solve Π. It is ∞, if Π is not solvable even for the robots with unique target functions. We then investigate the MASs of **self-stabilizing** algorithms for solving the gathering and related problems from **any** initial configuration. In what follows, an algorithm means a self-stabilizing algorithm, unless otherwise stated.

Motivations. We have investigated the time and the message complexities of distributed problems, considering time and message as important resources in the distributed computing. We regard target function as another resource. You will find the MASs of many problems are larger than 1 (but not ∞). The complexity of a problem is thus (at least partly) measured by its MAS. Also, an anonymous swarm with c distinct target functions can be regarded as a swarm with c distinct identifiers (and the same target function). Maintaining a large number of distinct

[3] Here, we abuse a term "algorithm." Despite that an algorithm must have a finite description conventionally, a target function (and hence a set of target functions) may not, as defined in Section 2. To compensate the abuse, when we will show the existence of an algorithm, we insist on giving a finite procedure to compute it.

identifiers not only is a centralized task, but also causes a substantial load. These motivate our theoretical work of establishing the MASs for problems.

Homonymous distributed systems. A distributed system is *homonymous*, if some processing elements (e.g., processors, processes, agents, or robots) may have the same identifier. Two extreme cases are anonymous systems and those whose identifiers are unique. The extreme cases have been investigated extensively. A relatively small number of researches on "properly" homonymous distributed system are known. Recall that we can identify an anonymous distributed system with c distinct local algorithms with a homonymous distributed system with c distinct identifiers.

Angluin [4] started investigation on anonymous computer networks in 1980, and a few researchers (e.g., [8, 22, 25]) followed her, to pursuit a purely distributed algorithm which does not rely on a central controller, in the spirit of the minimalist. Yamashita and Kameda [26] investigated the leader election problem on homonymous computer networks in 1989. Their main research topic was symmetry breaking, and they searched for a condition symmetry breaking becomes possible. A rough conclusion established is that symmetry breaking is impossible in general, but the probability that it is possible approaches to 1, as the number of processors increases, provided that the network topology is random, and identifiers, even if they are not unique, substantially increase the probability. The leader election problem on homonymous computer networks has also been investigated under several assumptions e.g., in [2, 14, 18, 28].

Other research topics on homonymous computer networks include failure detectors [5] and the Byzantine agreement problem [13]. In [13], the authors showed that the Byzantine agreement problem is solvable if and only if $\ell \geq 3f + 1$ in the synchronous case, and it is solvable only if $\ell > \frac{n+3f}{2}$ in the partially synchronous case, where ℓ is the number of distinct identifiers and f is an upperbound on the number of faulty processors. Thus, the MAS of the Byzantine agreement problem is $3f + 1$ in the synchronous case.

Only a few researchers have investigated homonymous swarms of robots: Team assembly of heterogeneous robots, each dedicated to solve a subtask, is discussed in [20], and the compatibility of target functions is discussed in [6, 11].

Contributions. We investigate the MASs for a variety of **self-stabilizing** problems, which are asked to solve problems from **any** initial configuration. The *c-scattering problem* (cSCT) is the problem of forming a configuration in which robots are distributed at least c different positions. The *scattering problem* is the nSCT. The *c-gathering problem* (cGAT) is the problem of forming a configuration in which robots are distributed at most c different positions. The *gathering problem* (GAT) is the 1GAT. The *pattern formation problem* (PF) for a given pattern G is the problem of forming a configuration P similar[4] to G.

We also investigate problems in the presence of *crash failures*: A faulty robot can stop functioning at any time, becoming permanently inactive. A faulty robot may not cause a malfunction, forever. We cannot distinguish such a robot from

[4] We say that one object is similar to another, if the latter is obtained from the former by a combination of scaling, translation, and rotation (but not by a reflection).

Table 1. For each problem Π, the MAS for Π, an algorithm for Π achieving the MAS (and the theorem/corollary/observation citation number establishing the result in parentheses) are shown.

problem Π	MAS	algorithm
cSCT ($1 \leq c \leq n$)	c	cSCTA (Thm. 1)
cGAT ($2 \leq c \leq n$)	1	2GATA (Cor. 1)
GAT	2	GATA (Thm. 3)
PF	n	PFA (Thm. 6)
fF1S ($1 \leq f \leq n-1$)	1	1SCTA (Obs. 1)
fF2S ($1 \leq f \leq n-2$)	$f+2$	$(f+2)$SCTA (Thm. 7)
$(n-1)$F2S	∞	– (Thm. 7)
fFcS ($c \geq 3$, $c+f-1 \leq n$)	$c+f-1$	$(c+f-1)$SCTA (Thm. 7)
fFcS ($c \geq 3$, $c+f-1 > n$)	∞	– (Thm. 7)
fFG ($1 \leq f \leq n-1$)	3	SGTA (Thm. 8)
fFGP ($1 \leq f \leq n-1$)	∞	– (Thm. 9)

a non-faulty one. The *f-fault tolerant c-scattering problem* (fFcS) is the problem of forming a configuration in which robots are distributed at c (or more) different positions, as long as at most f robots have crashed. The *f-fault tolerant gathering problem* (fFG) is the problem of gathering **all non-faulty robots** at a point, as long as at most f robots have crashed. The *f-fault tolerant gathering problem to f points* (fFGP) is the problem of gathering **all robots** (including faulty ones) at f (or less) points, as long as at most f robots have crashed.

Table 1 summarizes main results.

Organization. After introducing the robot model and several measures in Sect. 2, we first establish the MAS for the cSCT in Sect. 3. Then the MASs for the cGAT and the PF are respectively investigated in Sections 4 and 5. Sections 6 and 7 consider the MASs for the fFcS, the fFG, and the fFGP. Finally, we conclude the paper by giving open problems in Sect. 8.

2 Preliminaries

The model. Consider a swarm $\mathcal{R}$ of n robots $r_1, r_2, \ldots, r_n$. Each robot r_i has its own unit of length and a local compass, which define an x-y local coordinate system Z_i: Z_i is right-handed and self-centric, i.e., the origin $(0,0)$ always shows the position of r_i. Robot r_i has the strong multiplicity detection capability, and can count the number of robots that reside at a point.

A *target function* ϕ is a function from $(R^2)^n$ to $R^2 \cup \{\perp\}$ for all $n \geq 1$ such that $\phi(P) = \perp$, if and only if $(0,0) \notin P$.[5] Here, $\perp$ is a special symbol to denote that $(0,0) \notin P$. Given a target function ϕ_i, r_i executes a Look-Compute-Move cycle when it is activated:

Look: r_i identifies the multiset P of the robots' positions in Z_i.

[5] Since Z_i is self-centric, $(0,0) \notin P$ means an error of eye sensor, which we assume will not occur.

Compute: r_i computes $\boldsymbol{x}_i = \phi_i(P)$. Since $(0,0) \in P$, $\phi_i(P) \neq \perp$. (In case ϕ_i is not computable, we simply assume that $\phi_i(P)$ is given by an oracle.)
Move: r_i moves to $\boldsymbol{x}_i$, where it always reaches $\boldsymbol{x}_i$ before this Move phase ends.

We assume a discrete time $0, 1, \ldots$. At each time $t \geq 0$, a scheduler activates some unpredictable non-empty subset (that may be all) of robots. Then activated robots execute a cycle which starts at t and ends before (not including) $t+1$, i.e., $\mathcal{R}$ is semi-synchronous ($\mathcal{SSYNC}$).

Let Z_0 be the x-y global coordinate system. It is right-handed. The coordinate transformation from Z_i to Z_0 is denoted by γ_i. We use Z_0 and γ_i just for the purpose of explanation. They are not available to any robot r_i.

The position of robot r_i at time t in Z_0 is denoted by $\boldsymbol{x}_t(r_i)$. Then $P_t = \{\boldsymbol{x}_t(r_i) : 1 \leq i \leq n\}$ is a multiset representing the positions of all robots at time t, which is called the *configuration* of $\mathcal{R}$ at t.

Given an initial configuration P_0, an assignment $\mathcal{A}$ of a target function ϕ_i to each robot r_i, and an $\mathcal{SSYNC}$ schedule, the execution is a sequence $\mathcal{E} : P_0, P_1, \ldots, P_t, \ldots$ of configurations starting from P_0. Here, for all r_i and $t \geq 0$, if r_i is not activated at t, then $\boldsymbol{x}_{t+1}(r_i) = \boldsymbol{x}_t(r_i)$. Otherwise, if it is activated, r_i identifies $Q_t^{(i)} = \gamma_i^{-1}(P_t)$ in Z_i, computes $\boldsymbol{y} = \phi_i(Q_t^{(i)})$, and moves to $\boldsymbol{y}$ in Z_i. (Since $(0,0) \in Q_t^{(i)}$, $\boldsymbol{y} \neq \perp$.) Then $\boldsymbol{x}_{t+1}(r_i) = \gamma_i(\boldsymbol{y})$. We assume that the scheduler is fair: It activates every robot infinitely many times. Throughout the paper, we regard the scheduler as an adversary.

An $\mathcal{SSYNC}$ schedule is said to be *fully synchronous* ($\mathcal{FSYNC}$), if every robot r_i is activated every time instant $t = 0, 1, 2, \ldots$. A schedule which is not $\mathcal{SSYNC}$ is said to be *asynchronous* ($\mathcal{ASYNC}$). Throughout the paper, we assume that the scheduler is fair and $\mathcal{SSYNC}$, i.e., it always produces a fair $\mathcal{SSYNC}$ schedule.

Orders and Symmetries. Let $<$[6] be a lexicographic order on R^2. For distinct points $\boldsymbol{p} = (p_x, p_y)$ and $\boldsymbol{q} = (q_x, q_y)$, $\boldsymbol{p} < \boldsymbol{q}$, if either (i) $p_x < q_x$ or (ii) $p_x = q_x$ and $p_y < q_y$ holds. Let $\sqsubset$ be a lexicographic order on $(R^2)^n$. For distinct multisets of n points $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ and $Q = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n\}$, where for all $i = 1, 2, \ldots, n-1$, $\boldsymbol{p}_i \leq \boldsymbol{p}_{i+1}$ and $\boldsymbol{q}_i \leq \boldsymbol{q}_{i+1}$ hold, $P \sqsubset Q$, if there is an $i (1 \leq i \leq n-1)$ such that $\boldsymbol{p}_j = \boldsymbol{q}_j$ for all $j = 1, 2, \ldots, i-1$,[7] and $\boldsymbol{p}_i < \boldsymbol{q}_i$. The set of distinct points of P is denoted by $\overline{P} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m\}$, where $|P| = n$ and $|\overline{P}| = m$. We denote the multiplicity of $\boldsymbol{q}$ in P by $\mu_P(\boldsymbol{q})$, i.e., $\mu_P(\boldsymbol{q}) = |\{i : \boldsymbol{p}_i = \boldsymbol{q} \in P\}|$. We identify P with the pair $(\overline{P}, \mu_P)$, where μ_P is a labeling function to associate label $\mu_P(\boldsymbol{q})$ with each element $\boldsymbol{q} \in \overline{P}$.

Let G_P be the rotation group $G_{\overline{P}}$ of $\overline{P}$ about $\boldsymbol{o}_P$ preserving μ_P, where $\boldsymbol{o}_P$ is the center of the smallest enclosing circle of P. The order $|G_P|$ of G_P is denoted by k_P. We assume that $k_P = 0$, if $|\overline{P}| = 1$, i.e., if $\overline{P} = \{\boldsymbol{o}_P\}$. The symmetricity $\sigma(P)$ of P is $GCD(k_P, \mu_P(\boldsymbol{o}_P))$ [24]. See Fig. 1(1) for an example.

We use both measures k_P and $\sigma(P)$. Suppose that P is a configuration in Z_0. When activated, a robot r_i identifies the robots' positions $Q^{(i)} = \gamma_i^{-1}(P)$ in

[6] We use the same notation $<$ to denote the lexicographic order on R^2 and the order on R to save the number of notations.
[7] We assume $\boldsymbol{p}_0 = \boldsymbol{q}_0$.

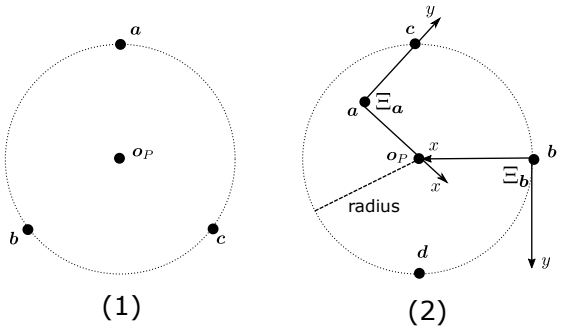

Fig. 1. (1) A configuration P, where $\overline{P} = \{\boldsymbol{o}_P, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$. If $\mu_P(\boldsymbol{a}) = \mu_P(\boldsymbol{b}) = \mu_P(\boldsymbol{c}) = i$ for an integer $i > 0$, then $k_P = 3$. If $\mu_P(\boldsymbol{o}_P) = 3j$ for an integer $j \geq 0$, then $\sigma(P) = 3$; otherwise, $\sigma(P) = 1$. (2) A configuration P, where $\overline{P} = \{\boldsymbol{o}_P, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}\}$. In Z_0, $\boldsymbol{o}_P = (0,0)$, $\boldsymbol{a} = (-1/2, 1/2)$, $\boldsymbol{b} = (1,0)$, $\boldsymbol{c} = (0,1)$, $\boldsymbol{d} = (0,-1)$, and the radius of C is 1. Solid arrows represent directions of x- and y-coordinates of $\Xi_{\boldsymbol{a}}$ and $\Xi_{\boldsymbol{b}}$, and have the unit length (the radius 1 of C). In $\Xi_{\boldsymbol{b}}$, $\boldsymbol{o}_P, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, and $\boldsymbol{d}$ are $(1,0)$,$(3/2,-1/2)$,$(0,0)$,$(1,-1)$, and $(1,1)$, respectively, and thus $\gamma_{\boldsymbol{b}}^{-1}(P) = V_P(\boldsymbol{b}) = \{(1,0),(3/2,-1/2),(0,0),(1,-1),(1,1)\}$.

Z_i in Look phase. Since P and $Q^{(i)}$ are similar, $k_P = k_{Q^{(i)}}$ and $\sigma(P) = \sigma(Q^{(i)})$ hold, i.e., all robots can consistently compute k_P and $\sigma(P)$.

On the contrary, robots cannot consistently compute lexicographic orders $<$ and $\sqsubset$. To see this fact, let $\boldsymbol{x}$ and $\boldsymbol{y}$ be distinct points in $\overline{P}$ in Z_0. Then both $\gamma_i^{-1}(\boldsymbol{x}) < \gamma_i^{-1}(\boldsymbol{y})$ and $\gamma_i^{-1}(\boldsymbol{x}) > \gamma_i^{-1}(\boldsymbol{y})$ can occur, depending on Z_i. Thus robots cannot consistently compare $\boldsymbol{x}$ and $\boldsymbol{y}$ using $>$. And it is the same for $\sqsubset$.

We introduce a total order $\succ$ on $\overline{P}$, in such a way that all robots can agree on the order, provided $k_P = 1$. A key trick behind the definition of $\succ$ is to use, instead of Z_i, an x-y coordinate system Ξ_i which is computable for any robot r_j from $Q^{(j)}$. Let $\Gamma_P(\boldsymbol{q}) \subseteq \overline{P}$ be the orbit of G_P through $\boldsymbol{q} \in \overline{P}$. Then $|\Gamma_P(\boldsymbol{q})| = k_P$ if $\boldsymbol{q} \neq \boldsymbol{o}_P$, and $\mu_P(\boldsymbol{q}') = \mu_P(\boldsymbol{q})$ if $\boldsymbol{q}' \in \Gamma_P(\boldsymbol{q})$. If $\boldsymbol{o}_P \in \overline{P}$, $\Gamma_P(\boldsymbol{o}_P) = \{\boldsymbol{o}_P\}$. Let $\Gamma_P = \{\Gamma_P(\boldsymbol{q}) : \boldsymbol{q} \in \overline{P}\}$. Then Γ_P is a partition of $\overline{P}$. Define an x-y coordinate system $\Xi_{\boldsymbol{q}}$ for each point $\boldsymbol{q} \in \overline{P} \setminus \{\boldsymbol{o}_P\}$. The origin of $\Xi_{\boldsymbol{q}}$ is $\boldsymbol{q}$, the unit distance is the radius of the smallest enclosing circle of P, the x-axis is taken so that it goes through $\boldsymbol{o}_P$, and it is right-handed. Let $\gamma_{\boldsymbol{q}}$ be the coordinate transformation from $\Xi_{\boldsymbol{q}}$ to Z_0. Then the view $V_P(\boldsymbol{q})$ of $\boldsymbol{q}$ is defined to be $\gamma_{\boldsymbol{q}}^{-1}(P)$. Obviously $V_P(\boldsymbol{q}') = V_P(\boldsymbol{q})$ (as multisets), if and only if $\boldsymbol{q}' \in \Gamma_P(\boldsymbol{q})$. Let $View_P = \{V_P(\boldsymbol{q}) : \boldsymbol{q} \in \overline{P} \setminus \{\boldsymbol{o}_P\}\}$. See Fig. 1(2) for an example.

A robot r_i, in Compute phase, can compute from $Q^{(i)}$, for each $\boldsymbol{q} \in \overline{Q^{(i)}} \setminus \{\boldsymbol{o}_{Q^{(i)}}\}$, $\Xi_{\boldsymbol{q}}$, $V_{Q^{(i)}}(\boldsymbol{q})$, and $View_{Q^{(i)}}$. Since P and $Q^{(i)}$ are similar, by the definition of $\Xi_{\boldsymbol{q}}$, $View_P = View_{Q^{(i)}}$, which implies that all robots r_i can consistently compute $View_P$. We define $\succ_P$ on Γ_P using $View_P$. For any distinct orbits $\Gamma_P(\boldsymbol{q})$ and $\Gamma_P(\boldsymbol{q}')$, $\Gamma_P(\boldsymbol{q}) \succ_P \Gamma_P(\boldsymbol{q}')$, if one of the following conditions hold:

1. $\mu_P(\boldsymbol{q}) > \mu_P(\boldsymbol{q}')$.

2. $\mu_P(\boldsymbol{q}) = \mu_P(\boldsymbol{q}')$ and $dist(\boldsymbol{q}, \boldsymbol{o}_P) < dist(\boldsymbol{q}', \boldsymbol{o}_P)$ hold, where $dist(\boldsymbol{x}, \boldsymbol{y})$ is the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{y}$.
3. $\mu_P(\boldsymbol{q}) = \mu_P(\boldsymbol{q}')$, $dist(\boldsymbol{q}, \boldsymbol{o}_P) = dist(\boldsymbol{q}', \boldsymbol{o}_P)$, and $V_P(\boldsymbol{q}) \sqsupset V_P(\boldsymbol{q}')$ hold. [8]

Then $\succ_P$ is a total order on Γ_P. If $k_P = 1$, since $\Gamma_P(\boldsymbol{q}) = \{\boldsymbol{q}\}$ for all $\boldsymbol{q} \in \overline{P}$, we regard $\succ_P$ as a total order on $\overline{P}$ by identifying $\Gamma_P(\boldsymbol{q})$ with $\boldsymbol{q}$. For a configuration P (in Z_0), from $Q^{(i)}$ (in Z_i), each robot r_i can consistently compute $k_P = k_{Q^{(i)}}$, $\Gamma_P = \Gamma_{Q^{(i)}}$ and $View_P = View_{Q^{(i)}}$, and hence $\succ_P = \succ_{Q^{(i)}}$. Thus, all robots can agree on, e.g., the largest point $\boldsymbol{q} \in \overline{P}$ with respect to $\succ_P$.

3 C-scattering Problem

Let $\mathcal{P} = \{P \in (R^2)^n : (0,0) \in P,\ n \geq 1\}$. Since a target function returns $\perp$ when $(0,0) \notin P$, we regard $\mathcal{P}$ as the domain of a target function.

The *scattering problem* (SCT) is the problem to have the robots occupy distinct positions, starting from any configuration [15]. For $1 \leq c \leq n$, let the *c-scattering problem* (cSCT) be the problem of transforming any initial configuration to a configuration P such that $|\overline{P}| \geq c$. Thus, the nSCT is the SCT. An algorithm for the cSCT is an algorithm for the $(c-1)$SCT, for $2 \leq c \leq n$.

Consider a set $c\text{SCTA} = \{sct_1, sct_2, \ldots, sct_c\}$ of c target functions, where target function sct_i is defined as follows for any $P \in \mathcal{P}$.

[Target function sct_i]
1. If $|\overline{P}| \geq c$, then $sct_i(P) = (0,0)$ for $i = 1, 2, \ldots, c$.
2. If $|\overline{P}| = 1$, then $sct_1(P) = (0,0)$, and $sct_i(P) = (1,0)$ for $i = 2, 3, \ldots, c$.
3. If $2 \leq |\overline{P}| \leq c-1$, $sct_i(P) = (\delta/(2(i+1)), 0)$ for $i = 1, 2, \ldots, c$, where δ is the smallest distance between two (distinct) points in $\overline{P}$.

Theorem 1. *For any $1 \leq c \leq n$, cSCTA is an algorithm for the cSCT. The MAS for the cSCT is c.*

Proof. We omit the proof that cSCTA is a correct algorithm for the cSCT, and present only a proof that the MAS for the cSCT is at least c.

The proof is by contradiction. Suppose that the MAS for the cSCT is $m < c$ to derive a contradiction. Let $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ be an algorithm for the cSCT. Consider the following situation:

1. All robots r_i $(1 \leq i \leq n)$ share the unit length and the direction of positive x-axis.
2. A target function assignment $\mathcal{A}$ is defined as follows: $\mathcal{A}(r_i) = \phi_i$ for $1 \leq i \leq m-1$, and $\mathcal{A}(r_i) = \phi_m$ for $m \leq i \leq n$.
3. All robots initially occupy the same location $(0,0)$. That is, $P_0 = \{(0,0), (0,0), \ldots, (0,0)\}$.
4. The schedule is $\mathcal{FSYNC}$.

Let $\mathcal{E} : P_0, P_1, \ldots$ be the execution of $\mathcal{R}$ starting from P_0, under the above situation. By an easy induction on t, all robots r_i $(m \leq i \leq n)$ occupy the same location, i.e., for all $t \geq 0$, $\boldsymbol{x}_t(r_m) = \boldsymbol{x}_t(r_{m+1}) = \cdots = \boldsymbol{x}_t(r_n)$. Since $|\overline{P_t}| \leq m < c$ for all $t \geq 0$, a contradiction is derived. □

[8] Since $dist(\boldsymbol{o}_P, \boldsymbol{o}_P) = 0$, $V_P(\boldsymbol{q})$ is not compared with $V_P(\boldsymbol{o}_P)$ with respect to $\sqsupset$.

4 C-gathering Problem

Let $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\} \in \mathcal{P}$, $\overline{P} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{m_P}\}$, $m_P = |\overline{P}|$ be the size of $\overline{P}$, $\mu_P(\boldsymbol{q})$ denote the multiplicity of $\boldsymbol{q}$ in P, $\boldsymbol{o}_P$ be the center of the smallest enclosing circle C_P of P, and $CH(P)$ be the convex hull of P.

The *c-gathering problem* (cGAT) is the problem of transforming any initial configuration to a configuration P such that $|\overline{P}| \leq c$. The 1GAT is thus the *gathering problem* (GAT). An algorithm for the cGAT is an algorithm for the $(c+1)$GAT, for $1 \leq c \leq n-1$.

Under the $\mathcal{SSYNC}$ scheduler, the GAT from distinct initial positions is solvable (by an algorithm of size 1), if and only if $n \geq 3$ [24], and the GAT from any initial configuration is solvable (by an algorithm of size 1), if and only if n is odd [16]. The MAS for the GAT is thus at least 2. Gathering algorithms $\psi_{f-point(n)}$ (for $n \geq 3$ robots from distinct initial positions) in Theorem 3.4 of [24] and Algorithm 1 (for odd n robots from any initial positions) in [16] share the skeleton: Given a configuration P, if there is the (unique) "largest point" $\boldsymbol{q} \in \overline{P}$, then go to $\boldsymbol{q}$; otherwise, go to $\boldsymbol{o}_P$. Consider the following singleton 2GATA $= \{2gat\}$ of a target function $2gat$, which is a direct implementation of this strategy using $\succ_P$ as the measure to determine the largest point in $\overline{P}$.

[Target function 2gat]

1. If $m_P = 1$, or $m_P = 2$ and $k_P = 2$, i.e., $\mu_P(\boldsymbol{q}_1) = \mu_P(\boldsymbol{q}_2)$, then $2gat(P) = (0,0)$.
2. If $m_P \geq 2$ and $k_P = 1$, then $2gat(P) = \boldsymbol{q}$, where $\boldsymbol{q} \in \overline{P}$ is the largest point with respect to $\succ_P$.
3. If $m_P \geq 3$ and $k_P \geq 2$, then $2gat(P) = \boldsymbol{o}_P$.

Theorem 2. *Suppose that all robots take $2gat$ as their target functions. Then they transform any initial configuration P_0 to a configuration P satisfying that (1) $m_P = 1$, or (2) $m_P = 2$ and $k_P = 2$.*

Corollary 1. *The MAS for the* cGAT *is* 1*, for all* $2 \leq c \leq n$.

Corollary 2. 2GATA *solves the* GAT*, if and only if the initial configuration* P_0 *satisfies either* $m_{P_0} \neq 2$ *or* $k_{P_0} \neq 2$.

Corollary 2 has been obtained by some researchers: The GAT is solvable (for the robots with the same target function), if and only if n is odd [16]. Or more precisely, it is solvable, if and only if the initial configuration is not bivalent [9]. Note that the algorithm of [9] makes use of the Weber point and tolerates at most $n-1$ crashes.

Consider a set GATA $= \{gat_1, gat_2\}$ of target functions gat_1 and gat_2 defined as follows:

[Target function gat_1]

1. If $m = 1$, then $gat_1(P) = (0,0)$.
2. If $m = 2$ and $k_P = 1$, or $m \geq 3$, $gat_1(P) = 2gat(P)$.
3. If $m = 2$ and $k_P = 2$, then $gat_1(P) = (0,0)$.

[Target function gat_2]

1. If $m = 1$, then $gat_2(P) = (0, 0)$.
2. If $m = 2$ and $k_P = 1$, or $m \geq 3$, then $gat_2(P) = 2gat(P)$.
3. Suppose that $m = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ ($\neq (0, 0)$) be the point in $\overline{P}$. Since $(0, 0) \in \overline{P}$, $\boldsymbol{q}$ is uniquely determined. If $\boldsymbol{q} > (0, 0)$, then $gat_2(P) = \boldsymbol{q}$. Else if $\boldsymbol{q} < (0, 0)$, then $gat_2(P) = 2\boldsymbol{q}$.

Theorem 3. GATA *is an algorithm for the* GAT. *The MAS for the* GAT *is, hence,* 2.

For a configuration P, let $-P = \{-\boldsymbol{p} : \boldsymbol{p} \in P\}$. A target function ϕ is said to be *symmetric* (with respect to the origin) if $\phi(P) = -\phi(-P)$ for all $P \in \mathcal{P}$. An algorithm Φ is said to be *symmetric* if every target function $\phi \in \Phi$ is symmetric. Target function $2gat$ is symmetric, but GATA is not a symmetric algorithm. Indeed, the next lemma holds.

Lemma 1. *There is no symmetric algorithm of size* 2 *for the* GAT.

There is however a symmetric gathering algorithm SGTA $= \{sgat_1, sgat_2, sgat_3\}$, where target functions $sgat_1, sgat_2$, and $sgat_3$ are defined as follows:

[Target function $sgat_1$]

1. If $m = 1$, then $sgat_1(P) = (0, 0)$.
2. If $m \geq 3$, or $m = 2$ and $k_P \neq 2$, then $sgat_1(P) = 2gat(P)$.
3. Suppose that $m = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ ($\neq (0, 0)$) be the point in $\overline{P}$. Since $(0, 0) \in \overline{P}$, $\boldsymbol{q}$ is uniquely determined. Then $sgat_2(P) = -\boldsymbol{q}$.

[Target function $sgat_2$]

1. If $m = 1$, then $sgat_2(P) = (0, 0)$.
2. If $m \geq 3$, or $m = 2$ and $k_P \neq 2$, then $sgat_2(P) = 2gat(P)$.
3. Suppose that $m = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ ($\neq (0, 0)$) be the point in $\overline{P}$. Since $(0, 0) \in \overline{P}$, $\boldsymbol{q}$ is uniquely determined. Then $sgat_2(P) = -2\boldsymbol{q}$.

[Target function $sgat_3$]

1. If $m = 1$, then $sgat_3(P) = (0, 0)$.
2. If $m \geq 3$, or $m = 2$ and $k_P \neq 2$, then $sgat_3(P) = 2gat(P)$.
3. Suppose that $m = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ ($\neq (0, 0)$) be the point in $\overline{P}$. Since $(0, 0) \in \overline{P}$, $\boldsymbol{q}$ is uniquely determined. Then $sgat_3(P) = -3\boldsymbol{q}$.

Theorem 4. SGTA *solves the* GAT. *The MAS of symmetric algorithm for the* GAT *is* 3.

5 Pattern Formation Problem

Given a goal pattern $G \in (R^2)^n$ in Z_0, the *pattern formation problem* (PF) for G is the problem of transforming any initial configuration I into a configuration similar to G. The GAT is the PF for a goal pattern $G = \{(0, 0), (0, 0), \ldots, (0, 0)\} \in (R^2)^n$, and the SCT is reducible to the PF for a right n-gon.

Theorem 5 ([29]). *The* PF *for a goal pattern G is solvable (by an algorithm of size 1) from an initial configuration I such that $|I| = |\overline{I}|$, if and only if $\sigma(G)$ is divisible by $\sigma(I)$. The only exception is the* GAT *for two robots.*

Thus a pattern G is not formable from a configuration I by an algorithm of size 1, if $\sigma(G)$ is not divisible by $\sigma(I)$. In the following, we investigate an algorithm that solves the PF from any initial configuration I, for any G.

Lemma 2. *The MAS for the* PF *is at least n.*

Proof. If there were a pattern formation algorithm for a right n-gon with size $m < n$, it could solve the SCT, which contradicts to Theorem 1. □

Theorem 6. *The MAS for the* PF *is n.*

Proof. By Lemma 2, the MAS for the PF is at least n.

To show that the MAS for the PF is at most n, we propose a PF algorithm PFA of size n, and then give a sketch of its correctness proof.

A scattering algorithm nSCTA transforms any initial configuration I into a configuration P satisfying $P = \overline{P}$. We can modify nSCTA so that the resulting algorithm nSCTA* can transform any initial configuration I into a configuration P satisfying ($P = \overline{P}$ and) $\sigma(P) = 1$. On the other hand, there is a pattern formation algorithm (of size 1) for G which transforms any initial configuration P satisfying ($P = \overline{P}$ and) $\sigma(P) = 1$ into a configuration similar to a goal pattern G (see e.g., [29]). The pattern formation problem is thus solvable by executing nSCTA* as the first phase, and then such a pattern formation algorithm as the second phase, if we can modify these algorithms so that the robots can consistently recognize which phase they are working. Algorithm PFA takes this approach. Since the cases of $n \leq 3$ are trivial, we assume $n \geq 4$.

We say a configuration P is *good*, if P satisfies either one of the following conditions (1) and (2):

(1) $P = \overline{P}$, i.e., P is a set, and can be partitioned into two subsets P_1 and P_2 satisfying all of the following conditions:
- **(1a)** $P_1 = \{\boldsymbol{p}_1\}$ for some $\boldsymbol{p}_1 \in P$.
- **(1b)** $dist(\boldsymbol{p}_1, \boldsymbol{o}_2) \geq 10\delta_2$, where $\boldsymbol{o}_2$ and δ_2 are respectively the center and the radius of the smallest enclosing circle C_2 of $P \setminus \{\boldsymbol{p}_1\}$.

(2) The smallest enclosing circle C of P contains exactly two points $\boldsymbol{p}_1, \boldsymbol{p}_3 \in P$, i.e., $\overline{\boldsymbol{p}_1\boldsymbol{p}_3}$ forms a diameter of C. Consider a (right-handed) x-y coordinate system Z satisfying $\boldsymbol{p}_1 = (0,0)$ and $\boldsymbol{p}_3 = (31,0)$.[9] For $i = 1,2,3$, let C_i be the unit circle with center $\boldsymbol{o}_i$ (and radius 1 in Z), where $\boldsymbol{o}_1 = (0,0)$, $\boldsymbol{o}_2 = (10,0)$, and $\boldsymbol{o}_3 = (30,0)$. Let $P_i \subseteq P$ be the multiset of points included in C_i for $i = 1,2,3$. Then P is partitioned into three submultisets P_1, P_2, and P_3, i.e., $P \setminus (P_1 \cup P_2 \cup P_3) = \emptyset$, and P_1, P_2, and P_3 satisfy the following conditions:
- **(2a)** $P_1 = \{\boldsymbol{p}_1\}$.

[9] Note that Z is uniquely determined, and the unit distance of Z is $dist(\boldsymbol{p}_1, \boldsymbol{p}_3)/31$.

(2b) P_2 is a set (not a multiset).

(2c) P_3 is a multiset that includes $\boldsymbol{p}_3$ as a member. It has a supermultiset P^* which is similar to G, and is contained in C_3, i.e., P_3 is similar to a submultiset $H \subseteq G$.

Let P be a good configuration. Then P satisfies exactly one of conditions (1) and (2), and $\boldsymbol{p}_1$ is uniquely determined in each case. We first define nSCTA$^* = \{sct_i^* : i = 1, 2, \ldots, n\}$, which is a slight modification of nSCTA.

[Target function sct_i^*]
(I) If P is good: $sct_i^*(P) = (0, 0)$ for $i = 1, 2, \ldots, n$.
(II) If P is not good:

1. For $i = 2, 3, \ldots, n$:
   If $P \neq \overline{P}$, then $sct_i^*(P) = sct_i(P)$. Else if $P = \overline{P}$, then $sct_i^*(P) = (0, 0)$.
2. For $i = 1$:
   (a) If $P \neq \overline{P}$, then $sct_1^*(P) = sct_i(P)$.
   (b) If $P = \overline{P}$ and $dist((0, 0), \boldsymbol{o}) < 10\delta$, then $sct_1^*(P) = \boldsymbol{p}$. Here $\boldsymbol{o}$ and δ are, respectively, the center and the radius of the smallest enclosing circle of $P \setminus \{(0, 0)\}$. If $\boldsymbol{o} \neq (0, 0)$, $\boldsymbol{p}$ is the point such that $(0, 0) \in \overline{\boldsymbol{op}}$ and $dist(\boldsymbol{p}, \boldsymbol{o}) = 10\delta$. If $\boldsymbol{o} = (0, 0)$, $\boldsymbol{p} = (10\delta, 0)$.
   (c) If $P = \overline{P}$ and $dist((0, 0), \boldsymbol{o}) \geq 10\delta$, then $sct_1^*(P) = (0, 0)$.

Then nSCTA* transforms any initial configuration P_0 to a good configuration P. We next explain how to construct a configuration similar to G from a good configuration P.

(I) Suppose that P satisfies condition (1) for a partition $\{P_1, P_2\}$, where $P_1 = \{\boldsymbol{p}_1\}$. If there is a point $\boldsymbol{q}$ such that $P_2 \cup \{\boldsymbol{q}\}$ is similar to G, then we move the robot at $\boldsymbol{p}_1$ to $\boldsymbol{q}$ to complete the formation.

Otherwise, let $\boldsymbol{p}_3$ be the point satisfying $\boldsymbol{o}_2 \in \overline{\boldsymbol{p}_1\boldsymbol{p}_3}$ and $dist(\boldsymbol{o}_2, \boldsymbol{p}_3) = 21\delta_2$, where $\boldsymbol{o}_2$ and δ_2 are, respectively, the center and the radius of the smallest enclosing circle C_2 of P_2. We choose a point $\boldsymbol{p}$ in P_2, and move the robot at $\boldsymbol{p}$ to $\boldsymbol{p}_3$. Note that the robot at $\boldsymbol{p}$ is uniquely determined, since $P_2 = \overline{P_2}$.

Then P is transformed into a configuration P' which is good, and satisfies condition (2) for partition $\{P_1, P_2 \setminus \{\boldsymbol{p}_2\}, \{\boldsymbol{p}_3\}\}$.

(II) Suppose that P satisfies condition (2) for a partition $\{P_1, P_2, P_3\}$, where $P_1 = \{\boldsymbol{p}_1\}$. Like the above case, we choose a point $\boldsymbol{p}$ in P_2, and move the robot at $\boldsymbol{p}$ to a point $\boldsymbol{q}$. Here $\boldsymbol{q}$ must satisfy that there is a superset P^* of $P_3 \cup \{\boldsymbol{q}\}$ which is contained in C_3, and is similar to G.

By repeating this transformation, a configuration P such that $|P_3| = n - 1$ and P_3 is similar to a submultiset of G is eventually obtained, when P_2 becomes empty. Then $\boldsymbol{p}_1$ can move to $\boldsymbol{q}$ to complete the formation.

To carry out this process, we need to specify (i) $\boldsymbol{p} \in P_2$ in such a way that all robots can consistently recognize it, and (ii) $\boldsymbol{p}_3$ in (I) and $\boldsymbol{q}$ in (II).

We define a point $\boldsymbol{p} \in P_2$. When $|P_2| = 1$, $\boldsymbol{p}$ is the unique element of P_2. When $|P_2| \geq 2$, let $P_{12} = P_1 \cup P_2$. Then $k_{P_{12}} = 1$ by the definition of $\boldsymbol{p}_1$. Since $k_{P_{12}} = 1$, $\succ_{P_{12}}$ is a total order on P_{12} (and hence on P_2), which all robots in P (in particular, in P_2) can compute. Let $\boldsymbol{p} \in P_2$ be the largest point in P_2 with

respect to $\succ_{P_{12}}$. Since P_2 is a set, the robot r at $\boldsymbol{p}$ is uniquely determined, and r (or its target function) knows that it is the robot to move to $\boldsymbol{p}_3$ or $\boldsymbol{q}$.

We define the target points $\boldsymbol{p}_3$ and $\boldsymbol{q}$. It is worth emphasizing that r can choose the target point by itself, and the point is not necessary to share by all robots. Point $\boldsymbol{p}_3$ is uniquely determined. To determine $\boldsymbol{q}$, note that P_3 has a supermultiset P^* which is similar to G, and is contained in C_3. Thus r arbitrarily chooses such a multiset P^*, and takes any point in $P^* \setminus P_3$ as $\boldsymbol{q}$. (There may be many candidates for P^*. Robot r can choose any one, e.g., the smallest one in terms of $\sqsupset$ in its x-y local coordinate system.)

Using points $\boldsymbol{p}$, $\boldsymbol{p}_3$, and $\boldsymbol{q}$ defined above, we finally describe $\mathrm{PFA} = \{pf_1, pf_2, \ldots, pf_n\}$ for a goal pattern G, where target functions $pf_i (i = 1, 2, \ldots, n)$ are defined as follows:

[Target function pf_i]

1. When P is not good: $pf_i(P) = sct_i^*(P)$.
2. When P is a good configuration satisfying condition (1):
   (2a) Suppose that there is a $\boldsymbol{q}$ such that $P_2 \cup \{\boldsymbol{q}\}$ is similar to G. Then $pf_i(P) = \boldsymbol{q}$ if $(0,0) \in P_1$; otherwise, $pf_i(P) = (0,0)$.
   (2b) Suppose that there is no point $\boldsymbol{q}$ such that $P_2 \cup \{\boldsymbol{q}\}$ is similar to G. Then $pf_i(P) = \boldsymbol{p}_3$ if $(0,0)$ is the largest point in P_2 with respect to $\succ_{P_1 \cup P_2}$; otherwise, $pf_i(P) = (0,0)$.
3. When P is a good configuration satisfying condition (2): $pf_i(P) = \boldsymbol{q}$ if $(0,0)$ is the largest point in P_2 with respect to $\succ_{P_1 \cup P_2}$; otherwise, $pf_i(P) = (0,0)$.

The correctness of PFA is clear from its construction. □

6 Fault Tolerant Scattering Problems

A fault means a crash fault in this paper. The *f-fault tolerant c-scattering problem* (fFcS) is the problem of transforming any initial configuration to a configuration P such that $|\overline{P}| \geq c$, as long as at most f robots have crashed.

Observation 1 *1.* 1SCTA *solves the* fF1S *for all* $1 \leq f \leq n$*, since* $|\overline{P}| \geq 1$ *for any configuration* P*. The MAS for the* fF1S *is thus* 1 *for all* $1 \leq f \leq n$*.*
2. The MAS for the nFcS *is* ∞ *for all* $2 \leq c \leq n$*, since* $|\overline{P_0}| = |\overline{P_t}| = 1$ *holds for all* $t \geq 0$*, if* $|\overline{P_0}| = 1$*, and all robots have crashed at time* 0*.*

Theorem 7. *Suppose that* $1 \leq f \leq n-1$ *and* $2 \leq c \leq n$*.*

1. *The MAS for the* fF2S *is* ∞*, if* $f = n-1$*; otherwise if* $1 \leq f \leq n-2$*, the MAS for the* fF2S *is* $f+2$*. Indeed,* $(f+2)$SCTA *solves the* fF2S*, if* $1 \leq f \leq n-2$*.*
2. *If* $3 \leq c \leq n$*, the MAS for the* fFcS *is* ∞*, if* $c+f-1 > n$*; otherwise if* $c+f-1 \leq n$*, the MAS for the* fFcS *is* $c+f-1$*. Indeed,* $(c+f-1)$SCTA *solves the* fFcS*, if* $c+f-1 \leq n$*.*

7 Fault Tolerant Gathering Problems

The f-fault tolerant c-gathering problem (fFcG) is the problem of gathering **all non-faulty robots** at c (or less) points, as long as at most f robots have crashed. *The f-fault tolerant c-gathering problem to c points* (fFcGP) is the problem of gathering **all robots** (including faulty ones) at c (or less) points, as long as at most f robots have crashed. When $c = 1$, fFcG is abbreviated as fFG, and fFcGP is abbreviated as fFGP when $c = f$. The fFcG is not harder than the fFcGP by definition. In general, the fFcGP is not solvable if $c < f$.

Theorem 8. SGTA *solves the* fFG *for all* $f = 1, 2, \ldots, n-1$. *The MAS for the* fFG *is* 3.

The fFGP is definitely not easier than the fFG by definition. You might consider that the difference of difficulty between them would be subtle. Indeed, it is not the case.

Theorem 9. *The* fFGP *is unsolvable for all* $f = 1, 2, \ldots, n-1$. *That is, the MAS for the* fFGP *is* ∞ *for all* $f = 1, 2, \ldots, n-1$.

Proof (sketch). Suppose that there is an algorithm Φ for the fFGP. We arbitrarily choose a configuration P_0 such that $m_0 > f$, and consider any execution $\mathcal{E} : P_0, P_1, \ldots$ from P_0, provided that no crashes occur, under a schedule $\mathcal{S}$ we specify as follows: For P_t, let A_t be a largest set of robots such that its activation does not yield a goal configuration. If there are two or more such largest sets, A_t is an arbitrary one. Then $\mathcal{S}$ activates all robots in A_t at time t, and the execution reaches P_{t+1}, which is not a goal configuration. (A_t may be an empty set.)

Then $|U| \leq f$ holds. Suppose that at t_0 all robots in U crash, and consider a schedule $\mathcal{S}'$ that activates A_t for all $0 \leq t \leq t_0 - 1$, and $A_t \cup U$ for all $t \geq t_0$. Then the execution $\mathcal{E}'$ starting from P_0 under $\mathcal{S}'$ is exactly the same as $\mathcal{E}$, and does not reach a goal configuration, despite that $\mathcal{S}'$ is fair; Φ is not an algorithm for the fFGP. It is a contradiction. □

It is interesting to see that the fF$(f+1)$GP, which looks to be the "slightest" relaxation of the fFGP (= fFfGP), is solvable by an easy algorithm 2GATA of size 1.

Theorem 10. 2GATA *solves both of the* fF2G *and the* fF$(f+1)$GP, *for all* $f = 1, 2, \ldots, n-1$. *The MASs for the* fF2G *and* fF$(f+1)$GP *are both* 1, *for all* $f = 1, 2, \ldots, n-1$.

8 Conclusions

There is a problem like the self-stabilizing gathering problem which is not solvable by a swarm of anonymous oblivious mobile robots when all robots take the same target function. For a problem Π, we have investigated the minimum algorithm size (MAS) for Π, which is the number of distinct target functions

necessary and sufficient to solve Π from **any** initial configuration. To figure out the effects of the number of distinct target functions on the problem-solving ability, we have established the MASs for the gathering and related problems.

As mentioned in Section 1, we consider target function as a resource like time and message, and regard the MAS of a problem as a measure to measure the complexity of the problem. There is an apparent trade-off between the number of distinct target functions and the time complexity, but this topic has not been investigated in this paper, and is left as an interesting open problem.

In the real world, gathering objects needs energy (since entropy decreases), while scattering them does not (since entropy increases). Thus a natural guess would be that cGAT is harder than cSCT. On the contrary, we have showed that, for $2 \leq c \leq n$, the MAS is c for cSCT, while it is 1, for cGAT. Other main results are summarized in Table 1.

Finally, we conclude the paper by giving a list of some open problems.

1. What is the MAS for the gathering problem under the $\mathcal{ASYNC}$ scheduler?
2. What is the MAS for the pattern formation problem for a fixed G?
3. What is the MAS for the f-fault tolerant convergence problem to f points, for $f \geq 3$?
4. What is the MAS for the Byzantine fault tolerant gathering problem?
5. Characterize the problem whose MAS is 2.
6. Investigate trade-off between the number of distinct target functions and the time complexity.

Acknowledgments. This work is supported in part by JSPS KAKENHI Grant Numbers JP17K00024 and JP22K11915.

References

1. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. In: 15th Annual ACM-SIAM Symposium on Discrete Algorithms, pp.1063–1071(2004)
2. Altisen, K., Datta, A.K., Devismes, S., Durand, A., Larmore, L.L.: Election in unidirectional rigns with homonyms. J. Parallel and Distributed Computing **146**, 79–95(2010)
3. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: A distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Trans. Robot. Autom. **15**, 818–828(1999)
4. Angluin, D.,: Local and global properties in networks of processors. In: 12th ACM Symposium on Theory of Computing, pp.82–93(1980)
5. Arévalo, S., Anta, A.F., Imbs, D., Jiménez, E., Raynal, M.: Failure detectors in homonymous distributed systems (with an application to consensus). J. Parallel and Distributed Computing **83**, 83–95(2015)
6. Asahiro, Y., Yamashita, M.: Compatibility of convergence algorithms for autonomous mobile robots. In: SIROCCO 2023, LNCS, vol.13892, pp.149–164(2023) (See also arXiv: 2301.10949 for the full version).
7. Asahiro, Y., Yamashita, M.: Minimum algorithm sizes for self-stabilizing gathering and related problems of autonomous mobile robots. arXiv: 2304.02212.

8. Attiya, H., Snir, M., Warmuth, M.K.: Computing on the anonymous ring. J. ACM **35**(4), 845–875(1988)
9. Bouzid, Z., Das, S., Tixeuil, S.: Gathering of mobile robots tolerating multiple crash faults. In: IEEE 33rd Int'l Conference on Distributed Computing Systems, pp.337–346(2013)
10. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: gathering. SIAM J. Comput. **41**, 829–879(2012)
11. Cord-Landwehr, A., et al.: A new approach for analyzing convergence algorithms for mobile robots. In: ICALP 2011, LNCS, vol.6756, pp.650–661(2011)
12. Das, S., Flocchini, P., Santoro, N., Yamashita, M.: Forming sequences of geometric patterns with oblivious mobile robots. Distrib. Comput. **28**, 131–145(2015)
13. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Kermarrec, A., Ruppert, E., Tran-The, H.: Byzantine agreement with homonyms, Distrib. Comput. **26**, 321–340(2013)
14. Delporte-Gallet, C., Fauconnier, H., Tran-The, H.: Leader election in rings with homonyms, In: Int'l Conf. Networked Systems, pp.9–24(2014)
15. Dieudonné, Y., Petit, F.: Scatter of weak mobile robots. Parallel Processing Letters **19**(1), 175–184(2009)
16. Dieudonné, Y., Petit, F.: Self-stabilizing gathering with strong multiplicity detection. Theor. Comput. Sci. **428**, 47–57(2012)
17. Dolve, S.: Self-stabilization. MIT Press(2000)
18. Dobrev, S., Pelc, A.: Leader election in rings with nonunique labels. Fundamenta Informaticae **59**(4), 333–347(2004)
19. Flocchini, P. : Gathering. In: Distributed Computing by Mobile Entities. LNCS, vol.11340, pp.63–82(2019)
20. Liu, Z., Yamauchi, Y., Kijima, S., Yamashita, M.: Team assembling problem for asynchronous heterogeneous mobile robots. Theor. Comput. Sci. **721**, 27–41(2018)
21. Lynch, N.: Distributed Algorithms. Morgan Kaufmann(1996)
22. Matias, Y., Afek, Y.: Simple and efficient election algorithms for anonymous networks. In: 3rd Int'l Workshop on Distributed Algorithms, pp.183–194(1989)
23. Prencipe, G.: Pattern formation In: Distributed Computing by Mobile Entities. LNCS, vol.11340, pp.37–62(2019)
24. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots – formation and agreement problems. SIAM J. Comput. **28**, 1347–1363(1999)
25. Yamashita, M., Kameda, T.: Computing on an anonymous network. In: 7th ACM Symposium on Principles of Distributed Computing, pp.117–130(1988)
26. Yamashita, M., Kameda, T.: Electing a leader when processor identity numbers are not distinct (extended abstract). In: Int'l Workshop WDAG'89, pp.303–314(1989)
27. Yamashita, M., Kameda, T.: Computing on anonymous networks I. Characterizing solvable cases. IEEE Trans. Parallel and Distributed Systems **7**(1), 69–89(1996)
28. Yamashita, M., Kameda, T.: Leader election problem on networks in which processor identity numbers are not distinct. IEEE Trans. Parallel and Distributed Systems **10**(9), 878–887(1999)
29. Yamashita, M., Suzuki, I.: Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theor. Comput. Sci. **411**, 2433–2453(2010)
30. Yamauchi, Y., Uehara, T., Kijima, S., Yamashita, M.: Plane formation by synchronous mobile robots in the three-dimensional Euclidean space. J. ACM **64**, 1–43(2017)