Minimum Algorithm Sizes for the Gathering and Related Problems of Autonomous Mobile Robots*

Yuichi Asahiro[a,*], Masafumi Yamashita[b]

[a] *Department of Information Science, Kyushu Sangyo University, 2-3-1 Matsukadai, Higashi-ku, 813-8503, Fukuoka, Japan*
[b] *Kyushu University*

Abstract

This paper investigates a swarm of autonomous mobile robots in the Euclidean plane, under the semi-synchronous ($\mathcal{SSYNC}$) scheduler. Each robot has a target function to determine a destination point from the robots' positions. Conventionally, all robots in the swarm take the same target function. We allow the robots to take different target functions, and investigate the effects of the number of distinct target functions on the problem-solving ability, regarding target function as a resource to solve a problem like time. Specifically, we are interested in how many distinct target functions are necessary and sufficient to solve a problem Π. The number of distinct target functions necessary and sufficient to solve Π is called the *minimum algorithm size* (MAS) for Π. The MAS is defined to be ∞, if Π is unsolvable even for the robots with unique target functions.

We show that the problems form an infinite hierarchy with respect to their MASs; for each integer $c > 0$ and ∞, the set of problems whose MAS is c is not empty, which implies that target function is a resource irreplaceable, e.g., with time. We propose MAS as a natural measure to measure the complexity of a problem.

We establish the MASs for solving the gathering and related problems from any initial configuration, i.e., in a self-stabilizing manner. For example, 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

*An extended abstract of this paper appeared in the Proc. 25th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2023) [9]. This work is licensed under CC BY-NC-ND 4.0.

*Corresponding author

1. Introduction

Swarms of anonymous oblivious mobile robots have been attracting many researchers for three decades, e.g., [1, 3, 15, 17, 18, 19, 20, 22, 26, 31, 32, 33, 35, 36, 43, 45, 46, 52, 53]. An anonymous oblivious mobile robot, which is represented by a point that moves in the Euclidean space, looks identical and indistinguishable, lacks identifiers (i.e., anonymous) and communication devices, and operates in Look-Compute-Move cycles: When a robot starts a cycle, it identifies the multiset of the robots' positions, computes the destination point using a function, called *target function*[1] based only on the multiset identified, and then moves towards the destination point.

All the papers listed above assume that all anonymous robots in a swarm take the same target function. It makes sense, since anonymous robots do not have identifiers: Roughly speaking, robots taking different target functions can behave, as if they had different identifiers, regarding the target functions as their identifiers (which may not be unique). On the other hand, robots with different identifiers can behave, as if they had different target functions, even when they have the same target function. They investigate a variety of problems from simple ones like the convergence and the gathering problems (e.g., [1, 45]) to hard ones like the formation problem of a sequence of patterns and the gathering problem in the presence of Byzantine failures (e.g., [20, 31]), and show that swarms of anonymous oblivious robots are powerful enough to solve certain sufficiently hard problems. At the same time, however, we have also recognized limitation of their problem-solving ability, e.g., the gathering problem for two robots is not solvable [52].

A natural and promising approach to increase the problem-solving ability of a swarm is to allow robots to take different target functions, or equivalently, to give robots different identifiers, since almost all artificial distributed systems enjoy having unique identifiers, e.g., serial numbers, to solve hard problems. We take this approach, and investigate the effects of the number of distinct target functions on the problem-solving ability. That is, we regard target function as a resource like time, and investigate the number of distinct target functions necessary and sufficient to solve a problem.

Let $\mathcal{R}$ and Φ be a swarm of n robots $r_1, r_2, \ldots, r_n$, and a set of target functions such that $|\Phi| \leq n$, respectively. A (target function) assignment $\mathcal{A} : \mathcal{R} \to \Phi$ 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*[2] of $\mathcal{R}$ to solve a problem Π, if $\mathcal{R}$ solves Π, no matter which assignment $\mathcal{A}$ that $\mathcal{R}$ takes. (Thus, we cannot

[1] A formal definition of target function will be given in Section 2.1.

[2] Here, we abuse a term "algorithm" in two ways: First, all robots are conventionally assumed to take the same target function, so that an algorithm is usually a target function that solves a problem. Second and more importantly, an algorithm must have a finite description. However, a target function (and hence a set of target functions) may not, as defined in Section 2.1. To compensate this abuse, when we will show the existence of an algorithm, we insist on giving a finite procedure to compute it. To show its non-existence, on the other hand, we will show that a function (not only an algorithm) does not exist.

assume a particular assignment when designing target functions.) The size $|\Phi|$ of an algorithm Φ is the number of target functions composing Φ. The *minimum algorithm size* (MAS) for Π is the algorithm size necessary and sufficient to solve Π. The MAS for Π is defined to be ∞, if Π is unsolvable even for the robots with unique target functions.

Under the semi-synchronous ($\mathcal{SSYNC}$) scheduler defined in Section 2.1, we establish the MASs of **self-stabilizing** algorithms for solving the gathering and related problems from **any** initial configuration. The papers given in the first paragraph propose a variety of algorithms for anonymous oblivious robots for a variety of problems, but most of them are not self-stabilizing; they solve problems only from initial configurations satisfying some conditions. In what follows, an algorithm means a self-stabilizing algorithm, unless otherwise stated.

Motivations

As pointed out, the MAS for a problem of an anonymous swarm is equal to the number of identifiers of a homonymous swarm necessary and sufficient to solve the problem. We claim that target function (and, hence, identifier) are irreplaceable resources, and propose that MAS is a natural complexity measure worth investigating. To this end, we shall prove that there is a problem with MAS being c for each integer $c > 0$ and ∞, i.e., the problems form an infinite hierarchy with respect to their MASs, which implies that target function (and hence identifier) are resources not substitutable, e.g., with time. We will mainly establish the MASs for the gathering and related problems, hoping that we can reach deeper understanding of their similarity and/or difference, e.g., the reason why any algorithm for a problem needs more target functions than others. Indeed, the proofs are instantiations of the understanding.

Related works – Homonymous distributed systems

A distributed system is said to be *homonymous*, if some processing elements (e.g., processors, processes, agents, or robots) may have the same identifier. Two extreme cases are anonymous systems and systems whose processing elements have unique identifiers.

Maintaining the system elements of a distributed system to have unique identifiers is a promising strategy to efficiently solve distributed problems. In most distributed systems, processing elements have unique identifiers such as serial numbers. As a matter of course, distributed system models such as message-passing and shared memory models assume the processing elements with unique identifiers [41].

Angluin [4] started investigation on anonymous computer network in 1980, and a few researchers (e.g., [11, 42, 47]) followed her, to pursuit a purely distributed algorithm which does not rely on a central controller, in the spirit of the minimalist. Their main research topic was symmetry breaking; they searched for a condition characterizing when symmetry breaking is possible in terms of the network topology. (Later, [14] and [50] characterized the solvable cases.) 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.

Another popular problem in the early years was the function computation problem [10, 11, 13, 37, 49]. They characterized functions computable on anonymous networks, and analyzed their computational complexities. Since then, many articles have appeared on anonymous computing.

Yamashita and Kameda [48] investigated the leader election problem on homonymous computer networks in 1989, and showed that the identifiers, even if they are not unique, are frequently crucial information to solve the problem by characterizing when the problem becomes solvable. The leader election problem on homonymous computer networks (under different problem settings) has also been investigated, e.g., in [2, 24, 29, 51].

Other research topics on homonymous computer networks include failure detectors [6] and the Byzantine agreement problem [23]. In [6], the authors introduced a failure detector class suitable for a homonymous system prone to crash faults, where processes are partially synchronous and links are eventually timely, and showed how to implement and use it in such a system, without assuming the number of processes as initial information. In [23], 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 $\ell > (n + 3f)/2$ 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, and it is greater than $(n + 3f)/2$ in the partially synchronous case.

The anonymous mobile robot model was introduced in 1996 [44]. The model was constructed, inspired by the development of distributed robot systems such as multi-robot systems, drone networks, satellite constellations, and so on. Unlike the anonymous computer network model, system elements called robots reside in the Euclidean space, and change their positions, responding to the current positions of other robots. Some research works are cited in the first paragraph of this section.

The anonymous mobile agent model introduced in 2001 is another model of anonymous distributed system [28]. It models software agents which repeatedly migrate from a computer to another through communication links. In the anonymous mobile agent model, anonymous agents move in a given anonymous finite graph. The black-hole problem [28] and the problem of searching for an intruder [12] were investigated in early days.

The anonymous mobile robot and the anonymous mobile agent models share many problems such as the gathering and the pattern formation problems. Solving such problems in a self-stabilizing manner while keeping the spirit of minimalist as much as possible is one of their main issues (see surveys [16, 34, 40]). For the literature of self-stabilization in general, see, e.g., [30]. Sometimes concepts and techniques introduced in the research of anonymous computer network (e.g., the view [45, 47]) are also applied to solve these problems.

There are a few papers that treat homonymous swarms, e.g., [7, 8, 19, 39].

Team assembly of heterogeneous robots, each dedicated to solve a subtask, is discussed in [39] (see also [38]). In [7], robots' identifiers are used to specify and evaluate the quality of the robots' trajectories. The compatibility of target functions is discussed [8, 19]: A set Φ of target functions is compatible with respect to a problem Π, if a swarm of robots always solves Π, as long as each robot takes its target function from Φ. Thus, two swarms whose robots take target functions from Φ can freely merge to form a larger swarm that solves Π.

Since most of natural distributed systems consist of anonymous entities, and do not have a central controller, natural distributed systems have inspired researchers to introduce other anonymous distributed system models. One of the models is the population protocol model, which was introduced in 2006 [5]. It is a model for a collection of agents, which are identically programmed finite state machines. The amoebot model is another model introduced in 2014 [25] (see also a survey [21]). Like the anonymous mobile agent model, anonymous particles of the amoebot model move in a graph, but unlike the anonymous mobile agent model, they move in an anonymous infinite graph.

Finally, the concept of MAS can be introduced to each of the anonymous systems after allowing processes, robots, agents or amoebots to have different algorithms.

Contributions

This paper investigates the MAS for various self-stabilizing gathering and related problems, which are asked to solve problems from **any** initial configuration. Throughout the paper, the scheduler is assumed to be **semi-synchronous** ($\mathcal{SSYNC}$), i.e., on a robot r, a Look-Compute-Move cycle starts at an integral time instant t, and ends before (not including) $t+1$. More carefully, r observes the robots' positions at time t, and has reached its destination before the cycle ends.

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*, sometimes called the *split 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 thus 1GAT. The *pattern formation problem* (PF) for a pattern G is the problem of forming a configuration P similar[3] 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 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,

[3]Throughout this paper, 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 using a reflection).

Table 1: For each self-stabilizing 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. cSCT is the c-scattering problem, cGAT is the c-gathering problem, PF is the pattern formation problem, fFcS is the fault tolerant c-scattering problem in the presence of at most f faulty robots, fFG is the fault tolerant gathering problem in the presence of at most f faulty robots, and fFGP is the fault tolerant problem of gathering all robots (including a faulty one) at f (or less) points in the presence of at most f faulty robots. fFcS (for some values of f and c) and fFGP are unsolvable, thus their MASs are ∞.

problem Π	MAS	algorithm
cSCT ($1 \leq c \leq n$)	c	cSCTA (Thm. 1)
cGAT ($2 \leq c \leq n$)	1	2GATA (Cor. 2)
GAT (= 1GAT)	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. 9)
fFGP ($1 \leq f \leq n-1$)	∞	– (Thms. 10,13)

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 we will use in this paper in Section 2, we first establish the MAS of the c-scattering problem in Section 3. Then the MASs of the c-gathering and the pattern formation problems are respectively investigated in Sections 4 and 5. Sections 6 and 7 consider the MASs of the fault tolerant scattering and the gathering problems, respectively. Finally, we conclude the paper by giving several open problems in Section 8.

2. Preliminaries

2.1. 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 a local x-y coordinate system Z_i: Z_i is right-handed, and the origin $(0,0)$ always shows the position of r_i, i.e., it is self-centric. Robot r_i has the strong multiplicity detection capability, and can count the number of robots resides at a point.

Let $\mathcal{P}$ be the set of all (non-empty) **multisets** P of points in R^2 such that $(0,0) \in P$. A *target function* ϕ is a function from $\mathcal{P}$ to R^2. 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.

Compute: r_i computes $\boldsymbol{x}_i = \phi_i(P)$. (In case ϕ_i is not computable, we simply assume that $\phi_i(P)$ is given by an oracle.)

Move: r_i moves to $\boldsymbol{x}_i$. (We assume that r_i always reaches $\boldsymbol{x}_i$ before this Move phase ends.)

We assume a discrete time $0, 1, \ldots$. At each time $t \geq 0$, the scheduler nondeterministically activates some (possibly all) robots. Then activated robots execute a cycle which starts at t and ends before (not including) $t+1$, i.e., the scheduler is semi-synchronous ($\mathcal{SSYNC}$).

Let Z_0 be the global x-y 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,[4] 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. 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.

The $\mathcal{SSYNC}$ scheduler is said to be *fully synchronous* ($\mathcal{FSYNC}$), if every robot r_i is activated every time instant $t = 0, 1, 2, \ldots$. The scheduler which is not $\mathcal{SSYNC}$ is said to be *asynchronous* ($\mathcal{ASYNC}$). Throughout the paper, we assume that the scheduler is $\mathcal{SSYNC}$.

In what follows, when we discuss a problem for a fixed size n of the robot swarm, the domain of a target function is the set $\mathcal{P}_n$ of all (non-empty) multisets P of n points such that $(0,0) \in P$. However, if it is obvious from the context, we omit n from $\mathcal{P}_n$ to write $\mathcal{P}$. For two sets $X, Y (\subseteq R^2)$ of points, $dist(X,Y) = \min_{\boldsymbol{x} \in X, \boldsymbol{y} \in Y} dist(\boldsymbol{x}, \boldsymbol{y})$, and $dist(\boldsymbol{x}, Y) = dist(\{\boldsymbol{x}\}, Y)$. For a configuration P, let $-P = \{-\boldsymbol{p} : \boldsymbol{p} \in P\}$.

[4] An $\mathcal{SSYNC}$ schedule is an activation schedule produced by the $\mathcal{SSYNC}$ scheduler.

2.2. Orders and symmetries

We use three orders $<$, $\sqsubset$, and $\succ$ (besides the conventional order $<$ on R). Let $<^5$ 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 and only 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 $\mathcal{P}$ defined as follows: 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 and only if there is an i ($1 \leq i \leq n-1$) such that (i) $\boldsymbol{p}_j = \boldsymbol{q}_j$ for all $j = 1, 2, \ldots, i-1$,[6] and (ii) $\boldsymbol{p}_i < \boldsymbol{q}_i$. Note that $\boldsymbol{p} \in \mathcal{P}_m$ and $\boldsymbol{q} \in \mathcal{P}_n$ are incomparable in $\sqsubset$, when $m \neq n$.

Let $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\} \in \mathcal{P}_n$. 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 a 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 of P is $\sigma(P) = GCD(k_P, \mu_P(\boldsymbol{o}_P))$, i.e., the greatest common divisor of k_P and $\mu_P(\boldsymbol{o}_P)$ [45]. Here, $GCD(\ell, 0) = GCD(0, \ell) = \ell$.

Example 1. *Let $P_1 = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$, $P_2 = \{\boldsymbol{a}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{b}, \boldsymbol{c}\}$, and $P_3 = \{\boldsymbol{a}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{c}\}$, where a triangle $\boldsymbol{abc}$ is equilateral. Then $k_{P_1} = \sigma(P_1) = 3$, $k_{P_2} = \sigma(P_2) = 1$, and $k_{P_3} = \sigma(P_3) = 3$.*

Let $P_4 = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{o}, \boldsymbol{o}\}$, $P_5 = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{o}, \boldsymbol{o}, \boldsymbol{o}\}$, and $P_6 = \{\boldsymbol{o}, \boldsymbol{o}, \boldsymbol{o}\}$, where $\boldsymbol{o}$ is the center of the smallest enclosing circle of an equilateral triangle $\boldsymbol{abc}$. Then $k_{P_4} = 3$, $\sigma(P_4) = 1$, $k_{P_5} = \sigma(P_5) = 3$, $k_{P_6} = 0$, and $\sigma(P_6) = 3$. See Figure 1 for an illustration.

We use both 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 Z_i in Look phase. Since P and $Q^{(i)}$ are similar, $k_P = k_{Q^{(i)}}$ and $\sigma(P) = \sigma(Q^{(i)})$, 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, 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 may be unable to consistently compare $\boldsymbol{x}$ and $\boldsymbol{y}$ using $>$. And, it is true for $\sqsubset$, as well. We thus introduce a total order $\succ_P$ 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 $\succ_P$ 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 and only if $\boldsymbol{q}' \in \Gamma_P(\boldsymbol{q})$. If $\boldsymbol{o}_P \in \overline{P}$, $\Gamma_P(\boldsymbol{o}_P) =$

[5] We use the same notation $<$ to denote the lexicographic order on R^2 and the order on R to save the number of notations.

[6] We assume $\boldsymbol{p}_0 = \boldsymbol{q}_0$.

Figure 1: A configuration P, where $\overline{P} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{o}\}$. A triangle $\boldsymbol{abc}$ is equilateral, and $\boldsymbol{o}$ is the center of the smallest enclosing circle of P. If $\mu_P(\boldsymbol{a}) = \mu_P(\boldsymbol{b}) = \mu_P(\boldsymbol{c}) = i$ for an integer $i > 0$, $k_P = 3$; otherwise, $k_P = 1$. If ($k_P = 3$ and) $\mu_P(\boldsymbol{o}) = 3j$ for an integer $j \geq 0$, then $\sigma(P) = 3$; otherwise, $\sigma(P) = 1$.

$\{\boldsymbol{o}_P\}$. Let $\Gamma_P = \{\Gamma_P(\boldsymbol{q}) : \boldsymbol{q} \in \overline{P}\}$. Then Γ_P is a partition of $\overline{P}$. Define x-y coordinate system $\Xi_{\boldsymbol{q}}$ for any 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 Figure 2 for an illustration.

Any robot r_i, in Compute phase, can compute $\Xi_{\boldsymbol{q}}$ and $V_{Q^{(i)}}(\boldsymbol{q})$ for each $\boldsymbol{q} \in \overline{Q^{(i)}} \setminus \{\boldsymbol{o}_{Q^{(i)}}\}$, and thus $View_{Q^{(i)}}$, from $Q^{(i)}$. 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 $\Gamma_P(\boldsymbol{q}')$ in Γ_P, $\Gamma_P(\boldsymbol{q}) \succ_P \Gamma_P(\boldsymbol{q}')$, if and only 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. [7]

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 $\succ_P$, provided $k_P = 1$.

[7] 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 $\sqsupset$.

Figure 2: A configuration P, where $P = \overline{P} = \{\boldsymbol{o}_P, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}\}$. In Z_0, $\boldsymbol{o}_P = (0,0)$, $\boldsymbol{a} = (-1/2, 1/2)$, $\boldsymbol{b} = (1,0)$, $\boldsymbol{c} = (0,1)$, and $\boldsymbol{d} = (0,-1)$. Then the smallest enclosing circle C of P has the center $\boldsymbol{o}_P$ and radius 1. Solid arrows represent the x- and y-axes of $\Xi_{\boldsymbol{a}}$ and $\Xi_{\boldsymbol{b}}$ with the unit length (the radius 1 of C). In $\Xi_{\boldsymbol{b}}$, points $\boldsymbol{o}_P$, $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ are $(1,0)$, $(3/2,-1/2)$, $(0,0)$, $(1,-1)$, and $(1,1)$, respectively, and thus $\gamma_{\boldsymbol{b}}^{-1}(P) = V_P(\boldsymbol{b}) = \{(1,0),(3/2,-1/2),(0,0),(1,-1),(1,1)\}$.

3. *C*-scattering problem

We start with showing that there is a problem whose MAS is c for all integer $c > 0$. Thus the problems form an infinite hierarchy with respect to their MASs.[8]

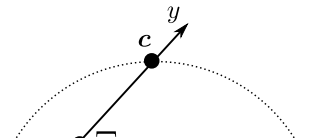
The *scattering problem* (SCT), sometimes called the *split problem*, is the problem to have the robots occupy distinct positions, starting from any configuration [26]. For $1 \leq c \leq n$, let the *c-scattering problem* (cSCT) be the problem to transform any initial configuration to a configuration 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 $2 \leq c \leq n$.

Theorem 1. *For any $1 \leq c \leq n$, the MAS for the c*SCT *is c.*

Proof. (I) We first show that the MAS for the cSCT is at least c. 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 $m \leq i \leq n$.

[8] We will show the existence of a problem whose MAS is ∞. However, this claim holds, regardless of whether or not there is such a problem with MAS being ∞.



3. All robots initially occupy the same location $(0,0)$. That is, $P_0 = \{(0,0), (0,0), \ldots, (0,0)\}$.
4. The scheduler 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.

(II) We next present an algorithm cSCTA $= \{sct_1, sct_2, \ldots, sct_c\}$ of size c for the cSCT, 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 $\overline{P}$.

We show that cSCTA is an algorithm for the cSCT. Consider any execution $\mathcal{E} : P_0, P_1, \ldots$, starting from any initial configuration P_0. We can assume $|\overline{P_0}| < c$ without loss of generality, since $|\overline{P_t}| \geq c$ implies $P_t = P_{t'}$ for all $t' \geq t$.

Proof is by contradiction. To derive a contradiction, we assume that $|\overline{P_t}| < c$ for all $t \geq 0$. Let $m = \max_{t \geq 0} |\overline{P_t}| < c$, and assume without loss of generality $|\overline{P_0}| = m$.

Suppose that $m = 1$, i.e., $\overline{P_0} = \{\boldsymbol{p}\}$ for some $\boldsymbol{p} \in R^2$. By the fairness of the $\mathcal{SSYNC}$ scheduler, let $t \geq 0$ be the time instant at which a robot r_i that takes a target function $sct_a (a \neq 1)$ is activated for the first time. Then $P_t = P_0$, and there is a robot r_j that takes sct_1 at $\boldsymbol{p}$. By the definition of cSCTA, $\boldsymbol{x}_{t+1}(r_j) = \boldsymbol{p}$, and $\boldsymbol{x}_{t+1}(r_i) = \gamma_i((1,0))$. Thus $|\overline{P_{t+1}}| \geq 2$, since $\boldsymbol{p} = \gamma_i((0,0)) \neq \gamma_i((1,0))$. It is a contradiction.

Suppose that $1 < m$ $(< c)$. First observe that $|\overline{P_t}| \geq m$ for all $t \geq 0$. Suppose $\boldsymbol{x}_t(r_i) \neq \boldsymbol{x}_t(r_j)$. Then $dist(\boldsymbol{x}_t(r_i), \boldsymbol{x}_t(r_j)) \geq \delta$. Since they both move at most distance $\delta/4$, $\boldsymbol{x}_{t+1}(r_i) \neq \boldsymbol{x}_{t+1}(r_j)$, regardless of their local x-y coordinate systems, Z_i and Z_j.

Suppose that $|\overline{P_t}| = m$ holds for all $t \geq 0$. At each t, if a robot r at $\boldsymbol{x}_t(r) = \boldsymbol{q}$ is activated, all robots r' such that $\boldsymbol{x}_t(r') = \boldsymbol{q}$ must be activated simultaneously, and move to the same location, i.e., their target functions are the same. However, it is a contradiction. There is a point $\boldsymbol{q} \in \overline{P_0}$ such that $\boldsymbol{x}_0(r_i) = \boldsymbol{x}_0(r_j) = \boldsymbol{q}$ for some $i \neq j$, and r_i and r_j take different target functions, since $m < c$. Thus, there is a time $t' > t$ such that $|\overline{P_{t'}}| > |\overline{P_t}| = m$.

Therefore, $\mathcal{E}$ eventually reaches a configuration P_t such that $|\overline{P_t}| \geq c$. □

Corollary 1. *Let $\mathcal{R}$ be a swarm consisting of n robots. For $c = 1, 2, \ldots, n$, there is a problem Π_c for $\mathcal{R}$ such that the MAS for Π_c is c.*

Proof. Let Π_c be the cSCT. Then the MAS for Π_c is c. □

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 $n \neq 2$ [45], and the GAT from any initial configuration is solvable (by an algorithm of size 1), if and only if n is odd [27]. 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 [45] and Algorithm 1 (for odd n robots from any initial positions) in [27] 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 $\overline{P}$.[9]

[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 $2gat(P) = \boldsymbol{o}_P$.

We call a configuration P *unfavorable* if $m_P = k_P = 2$, which is said to be *bivalent* in [15]. Otherwise, it is *favorable*. We show that algorithm 2GATA transforms any initial configuration either to a goal configuration or to an unfavorable configuration. A multiset P is said to be *linear* if $CH(P)$ is a line segment. We need the following technical lemma.

Lemma 1 ([8]). *Let A be a set (not a multiset) of points satisfying (1) A is not linear, (2) $k_A \geq 2$, and (3) $\boldsymbol{o}_A \notin A$. For a point $\boldsymbol{a} \in A$, let $B = (A \cup \{\boldsymbol{o}_A\}) \setminus \{\boldsymbol{a}\}$, i.e., B is constructed from A by replacing $\boldsymbol{a} \in A$ with $\boldsymbol{o}_A$. Then $k_B = 1$.*

Theorem 2. *Algorithm 2GATA transforms any initial configuration P_0 to a configuration P which is either $m_P = 1$ or unfavorable.*

[9] We will use 2GATA as a main building block to construct many gathering and fault tolerant gathering algorithms in the rest of this paper. 2GATA is different from $\psi_{f-point(n)}$ and Algorithm 1, particularly in that it uses total order $\succ_P$ defined in this paper. Since $\psi_{f-point(n)}$ assumes the robots to occupy initially distinct positions, and Algorithm 1 assumes n to be odd, they cannot solve some problems that 2GATA can, like the cGAT (for all $2 \leq c \leq n$).

Proof. Consider any execution $\mathcal{E} : P_0, P_1, \ldots$ of 2GATA, starting from any initial configuration P_0. Let $m_t = |\overline{P_t}|$, $k_t = k_{P_t}$, $\mu_t = \mu_{P_t}$, $C_t = C_{P_t}$, $\boldsymbol{o}_t = \boldsymbol{o}_{P_t}$, $CH_t = CH(P_t)$, and $\succ_t = \succ_{P_t}$.

We first claim that there is a time $t \geq 0$ such that $m_t = 1$ if $k_0 = 1$. Let $\boldsymbol{q} \in \overline{P_0}$ be the largest point with respect to $\succ_0$. It suffices to show that $\mu_t(\boldsymbol{q}) < \mu_{t+1}(\boldsymbol{q})$ for all $t \geq 0$, provided that $\mu_t(\boldsymbol{q}) < n$. The proof is by induction on t for all $t \geq 1$. The induction hypothesis $IH(t)$ is that (1) $\mu_t(\boldsymbol{q}) > \mu_t(\boldsymbol{q}')$ for any $\boldsymbol{q}'(\neq \boldsymbol{q}) \in \overline{P_t}$ (i.e., $\boldsymbol{q}$ is the largest point with respect to $\succ_t$, and hence P_t is favorable), (2) $\boldsymbol{q} = \boldsymbol{o}_t$ holds if $k_t > 1$, and (3) $\mu_t(\boldsymbol{q}) > \mu_{t-1}(\boldsymbol{q})$.

As for the base case, when $t = 1$, since $k_0 = 1$, robots activated at time 0 move to $\boldsymbol{q} \in \overline{P_0}$. Since $\mu_0(\boldsymbol{q}) \geq \mu_0(\boldsymbol{q}')$ for any $\boldsymbol{q}' \in \overline{P_0}$ by the definition of $\succ_0$, $\mu_1(\boldsymbol{q}) > \mu_0(\boldsymbol{q}) \geq \mu_0(\boldsymbol{q}') \geq \mu_1(\boldsymbol{q}')$. Therefore, $\mu_1(\boldsymbol{q}) > \mu_1(\boldsymbol{q}')$, $\boldsymbol{q} = \boldsymbol{o}_1$ if $k_1 > 1$, and $\mu_1(\boldsymbol{q}) > \mu_0(\boldsymbol{q})$. Thus, $IH(1)$ holds.

As for the induction step, for $t = 1, 2, \ldots$, we show $IH(t+1)$, provided $IH(t)$, which however is almost the same as the base case. When $k_t = 1$, since $\mu_t(\boldsymbol{q}) > \mu_t(\boldsymbol{q}')$ for any $\boldsymbol{q}'(\neq \boldsymbol{q}) \in \overline{P_t}$, and hence $\boldsymbol{q}$ is the largest point with respect to $\succ_t$, robots activated at time t move to $\boldsymbol{q}$, as in the base case, we have $\mu_{t+1}(\boldsymbol{q}) > \mu_t(\boldsymbol{q}) > \mu_t(\boldsymbol{q}') \geq \mu_{t+1}(\boldsymbol{q}')$, which implies that $\mu_{t+1}(\boldsymbol{q}) > \mu_{t+1}(\boldsymbol{q}')$, $\boldsymbol{q} = \boldsymbol{o}_{t+1}$ if $k_{t+1} > 1$, and $\mu_{t+1}(\boldsymbol{q}) > \mu_t(\boldsymbol{q})$, i.e., $IH(t+1)$ holds. Otherwise, if $k_t \geq 2$, robots activated at time t move to $\boldsymbol{o}_t$, which is $\boldsymbol{q}$ by $IH(t)$. Thus, $\mu_{t+1}(\boldsymbol{q}) > \mu_{t+1}(\boldsymbol{q}')$, $\boldsymbol{q} = \boldsymbol{o}_{t+1}$ if $k_{t+1} > 1$, and $\mu_{t+1}(\boldsymbol{q}) > \mu_t(\boldsymbol{q})$, i.e., $IH(t+1)$ holds. Thus, the claim holds: $\mu_t(\boldsymbol{q}) < \mu_{t+1}(\boldsymbol{q})$ for all $t \geq 0$, and hence there is a time $t \geq 0$ such that $m_t = 1$, if $k_0 = 1$.

The case $k_t \geq 2$ for all $t \geq 0$ remains. Without loss of generality, we may assume that P_t is favorable for all $t \geq 0$ (since otherwise we have nothing to show). We show that there is a time $t > 0$ such that $m_0 > m_t$, which implies $m_t = 1$ for some $t \geq 0$.

(I) First consider the case in which P_0 is linear. Let $\overline{P_0} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{m_0}\}$, where $\boldsymbol{q}_i$ $(1 < i < m_0)$ appear in $\overline{\boldsymbol{q}_1\boldsymbol{q}_{m_0}}$ in this order. By the definition of 2GATA, since P_0 is linear, so is P_t for all $t \geq 0$, and hence $k_t = 2$ since $k_t \geq 2$.

(I-A) Suppose that $m_0(\geq 3)$ is odd. Since $k_0 = 2$, $\boldsymbol{o}_0 = \boldsymbol{q}_{\lceil m_0/2 \rceil} \in \overline{P_0}$, and all robots activated at time 0 move to $\boldsymbol{o}_0$. Since $m_1 \leq m_0$, $m_1 = m_0$ (because we have nothing to show if $m_1 < m_0$), $\overline{P_1} = \overline{P_0}$, $\mu_1(\boldsymbol{o}_1) = \mu_1(\boldsymbol{o}_0) > \mu_0(\boldsymbol{o}_0)$, $k_1 = 2$, and P_1 is favorable, by assumption. Repeating this argument, since $\mu_t(\boldsymbol{o}_0) = n$ implies $m_t = 1$, there is a time t such that $m_t < m_0$.

(I-B) Suppose that $m_0(\geq 4)$ is even. Since $k_0 = 2$, all robots activated at time 0 move to $\boldsymbol{o}_0 = (\boldsymbol{q}_1 + \boldsymbol{q}_{m_0})/2 \notin \overline{P_0}$. Thus $m_0 \leq m_1 \leq m_0 + 1$ (since we have nothing to show if $m_1 < m_0$).

If $m_1 = m_0 + 1$, since m_1 is odd, $k_1 = 2$, and P_1 is favorable, by the argument in (I-A), eventually $m_1 > m_t$ holds for some $t > 1$ for the first time. Since all robots activated move to $\boldsymbol{o}_0$, $\overline{P_{t'}} = \overline{P_1}$ for all $1 \leq t' < t$. Since $k_t = 2$ and $m_t = m_0$ (since we have nothing to show if $m_t < m_0$), $\overline{P_t} = \overline{P_1} \setminus \{\boldsymbol{q}_e\} = (\overline{P_0} \cup \{\boldsymbol{o}_0\}) \setminus \{\boldsymbol{q}_e\}$, where e is either 1 or m_0. Furthermore, since $k_0 (= k_1) = k_t = 2$, for all $i = 1, 2, \ldots, m_0 - 1$ except for $m_0/2$, $dist(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}) = d$,

and $dist(\boldsymbol{q}_{m_0/2}, \boldsymbol{o}_0) = dist(\boldsymbol{o}_0, \boldsymbol{q}_{m_0/2+1}) = d$, for some $d > 0$. That is, eventually $\mathcal{E}$ reaches a configuration P_t such that $k_t = 2$, $m_t = m_0$ is even, and all points in $\overline{P_t}(= (\overline{P_0} \cup \{\boldsymbol{o}_0\}) \setminus \{\boldsymbol{q}_e\})$ occur in an equidistant manner in CH_t, where e is either 1 or m_0.

If $m_1 = m_0$, since $k_1 = 2$, by the same argument, $k_1 = 2$, $m_1 = m_0$ is even, and all points in $\overline{P_1}(= (\overline{P_0} \cup \{\boldsymbol{o}_0\}) \setminus \{\boldsymbol{q}_e\})$ occur in an equidistant manner in CH_1, where e is either 1 or m_0.

Thus, without loss of generality, we may assume that $k_0 = 2$, $m_0(\geq 4)$ is even, and all points $\boldsymbol{q}_i \in \overline{P_0}$ occur in an equidistant manner in CH_0.

We repeat the above arguments: Since $k_0 = k_1 = 2$, it must hold $2 \cdot dist(\boldsymbol{q}_1, \boldsymbol{q}_2) = dist(\boldsymbol{q}_{m_0/2}, \boldsymbol{q}_{m_0/2+1})$ in P_0, which contradicts the assumption for P_0.

(II) Next, consider the case in which P_0 is not linear. As in (I), $m_0 \leq m_1 \leq m_0 + 1$ (since we have nothing to show if $m_1 < m_0$).

(II-A) Suppose first that $m_1 = m_0 + 1$. That is, $\boldsymbol{o}_0 \notin \overline{P_0}$, and $\overline{P_1} = \overline{P_0} \cup \{\boldsymbol{o}_0\}$. Since $k_1 \geq 2$ and $\boldsymbol{o}_1 = \boldsymbol{o}_0$, all robots activated at time t move to $\boldsymbol{o}_0$, as long as $\overline{P_t} = \overline{P_1}$ holds.

Let $t > 1$ be the first time such that $m_0 = m_t < m_{t-1} = m_1$ hold (since we have nothing to show if $m_t < m_0$). Since $\overline{P_{t-1}} = \overline{P_1} = \overline{P_0} \cup \{\boldsymbol{o}_0\}$, $\boldsymbol{o}_{t-1} = \boldsymbol{o}_0$, and $k_{t-1} \geq 2$, all robots activated at $t-1$ move to $\boldsymbol{o}_0$. Suppose that $\overline{P_t} = \overline{P_1} \setminus \{\boldsymbol{q}\}$, for some $\boldsymbol{q} \in \overline{P_0}$ (since $\boldsymbol{o}_0$ remains in $\overline{P_t}$). Since $k_0 \geq 2$, P_0 is not linear, $\boldsymbol{o}_0 \notin \overline{P_0}$, and $\overline{P_t} = (\overline{P_0} \setminus \{\boldsymbol{q}\}) \cup \{\boldsymbol{o}_0\}$, we have $k_t = 1$ by Lemma 1, which contradicts the assumption that $k_t \geq 2$ for all t.

(II-B) Suppose next that $m_1 = m_0$. There are two cases to be considered. If $\boldsymbol{o}_0 \in \overline{P_0}$ and $\overline{P_1} = \overline{P_0}$, all robots activated at t move to $\boldsymbol{o}_0$, as long as $\overline{P_t} = \overline{P_0}$, since $k_0 \geq 2$ and P_t is favorable. Thus, eventually $\mathcal{E}$ reaches a configuration P_t such that $m_t < m_0$.

Otherwise, if $\boldsymbol{o}_0 \notin \overline{P_0}$, and $\overline{P_1} = (\overline{P_0} \cup \{\boldsymbol{o}_0\}) \setminus \{\boldsymbol{q}\}$ for some $\boldsymbol{q} \in \overline{P_0}$, where $\boldsymbol{q} \neq \boldsymbol{o}_0$. Since $k_0 \geq 2$, P_0 is not linear, $\boldsymbol{o}_0 \notin \overline{P_0}$, we have $k_t = 1$ by Lemma 1, which contradicts the assumption that $k_t \geq 2$ for all t. Thus, if $k_t \geq 2$ and P_t is favorable for all t, there is a time $t \geq 0$ such that $m_0 > m_t$. □

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

Proof. By Theorem 2, 2GATA is an algorithm for the 2GAT, which implies that it is an algorithm for the cGAT for all $2 \leq c \leq n$. □

If we consider that a problem with a smaller MAS is easier than a one with a larger MAS, the cGAT is not harder than the cSCT for all $2 \leq c \leq n$.

Corollary 3. *Algorithm* 2GATA *solves the* GAT*, if and only if the initial configuration is favorable.*

Proof. By the definition of $2gat$, 2GATA does not solve the GAT, if the initial configuration is unfavorable, since no robots can move at an unfavorable configuration. We show that 2GATA solves the GAT, if the initial configuration is favorable.

If a favorable configuration never yields an unfavorable configuration, any execution starting from a favorable configuration eventually reaches a goal configuration of the GAT by Theorem 2 (since it does not reach unfavorable configuration). We show this fact by contradiction.

Suppose that a configuration P_t at a time t is favorable, but P_{t+1} is unfavorable, to derive a contradiction. By the proof of Theorem 2, if $k_t = 1$, so is k_{t+1}, which implies that P_{t+1} is favorable. Thus, we can assume without loss of generality that $k_t \geq 2$ and $m_t \geq 3$.

Let $\overline{P_t} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{m_t}\}$ and $\overline{P_{t+1}} = \{\boldsymbol{p}_1, \boldsymbol{p}_2\}$. Without loss of generality, $\boldsymbol{p}_1 = \boldsymbol{o}_t$ and $\boldsymbol{p}_2 = \boldsymbol{q}_j$ for some $1 \leq j \leq m_t$, since all robots activated at t move to $\boldsymbol{o}_t$. Note that $\boldsymbol{o}_t$ may not be in $\overline{P_t}$. Observe that

$$\mu_{t+1}(\boldsymbol{p}_1) = n - \mu_{t+1}(\boldsymbol{p}_2) = n - \mu_{t+1}(\boldsymbol{q}_j) \geq n - \mu_t(\boldsymbol{q}_j) > n/2,$$

since $\mu_t(\boldsymbol{q}_j) < n/2$ (because $m_t \geq 3$ and $k_t \geq 2$), which is a contradiction to the fact that P_{t+1} is unfavorable. □

Corollary 3 has been obtained by some researchers: The GAT is solvable, if and only if n is odd [27]. Or more precisely, it is solvable, if and only if the initial configuration is favorable (i.e., not bivalent) [15]. Note that the algorithm of [15] makes use of the Weber point (instead of the center of the smallest enclosing circle), 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. GATA is an extension of 2GATA. Indeed, $gat_1 = 2gat$, and $gat_2 = 2gat$, as long as a configuration is favorable. If a configuration is unfavorable, gat_2 is designed so that it can yield a favorable configuration. If gat_2 works as we expect, which fact we shall show as Theorem 3, GATA is a GAT algorithm by Corollary 3.

[Target function gat_1]

1. $gat_1(P) = 2gat(P)$.

[Target function gat_2]

1. If P is favorable, then $gat_2(P) = 2gat(P)$.
2. If P is unfavorable, then $m_P = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ be a point in $\overline{P}$ such that $\boldsymbol{q} \neq (0,0)$. 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 $gat_2(P) = 2\boldsymbol{q}$.

In what follows, a robot taking ϕ as its target function is called a ϕ robot.

Figures 3 and 4 illustrate how gat_2 robots transform a given unfavorable configuration to a favorable one. Since gat_1 robots do not move when a configuration is unfavorable, Figure 3 illustrates gat_2 robots only: Four gat_2 robots $r_1, \ldots, r_4$ are represented by black and white triangles, where r_1 and r_4 are at $(0,0)$ in Z_4, r_2 and r_3 are at $\boldsymbol{q}$ in Z_4, and points $(0,0)$ and $\boldsymbol{q}$ are represented by large circles. The behaviours of the gat_2 robots do not change, even if the same number of gat_1 robots additionally reside at each of the points, like an unfavorable configuration in Figure 4(1), where gat_1 robots are represented by

Figure 3: An unfavorable configuration P with four gat_2 robots r_1, r_2, r_3, and r_4. The same number of gat_1 robots may reside at each of $(0,0)$ and $\boldsymbol{q}$, but they are omitted. The local x-y coordinate system for a robot represented by a black or a white triangle is depicted in the right box. For r_1, $\boldsymbol{q}$ (in Z_4) is larger than $(0,0)$ with respect to $\succ_P$, and hence it moves to $\boldsymbol{q}$ when activated. For r_2, r_3, and r_4, $(0,0)$ is larger than $\boldsymbol{q}$ with respect to $\succ_P$, when activated, r_2 and r_3 move to $(0,0)$, while r_4 to $2\boldsymbol{q}$ (not to $\boldsymbol{q}$).

white squares. We omit gat_1 robots from Figure 3. Robots represented by black and while triangles differ in their local x-y coordinate systems, as shown in the right box. Finally, a dashed arrow represents the move of a robot, where the target point (if it moves) is represented by a dashed circle.

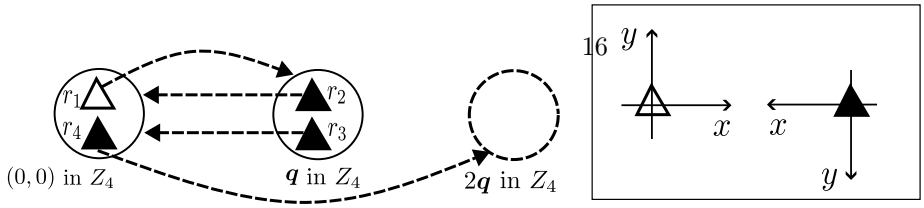
The key trick in GATA to break symmetry between the robots at $(0,0)$ and $\boldsymbol{q}$ works based on the asymmetric behaviors of gat_2 robots r_2 (or r_3) and r_4 represented by black triangles: r_3 can move to $(0,0)$, but it cannot return to $\boldsymbol{q}$. We will formally show the correctness of GATA in Theorem 3 using this asymmetry. Since the proof is by contradiction, to aid readers' understanding, we roughly illustrate, in Figure 4, how the unfarovable configuration in Figure 3 is transformed into a favorable one.

Given an unfavorable configuration in Figure 4(1) (which is the one in Figure 3 when there is a gat_1 robot at each points), an unfavorable configuration yields only if r_1 and r_2 are activated, and an unfavorable configuration in Figure 4(3) yields. Although these two unfavorable configurations in Figure 4(1) and (3) look the same, because of the asymmetry of the behaviors of r_1 and r_2, a favorable configuration in Figure 4(4) must yield, if r_1 and r_2 are activated. (If the scheduler activates another set of robots, a favorable configuration yields, too.)

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

Proof. We show that GATA $= \{gat_1, gat_2\}$ is a gathering algorithm. Consider any execution $\mathcal{E} : P_0, P_1, \ldots$ starting from any initial configuration P_0. By Corollary 3, we can assume that P_0 is unfavorable by the definition of GATA. Without loss of generality, we may assume that P_0 satisfies $\overline{P_0} = \{(0,0),(1,0)\}$ in Z_0, $\mu_0((0,0)) = \mu_0((1,0)) = h$, i.e., $n = 2h$ is even. By the definition of GATA again, it is easy to observe that P_t is linear. We show that there is a time t such that P_t is favorable. Proof is by contradiction.

A gat_1 robot does not move, if a configuration is unfavorable. Since there is at least one gat_1 robot, without loss of generality, we assume that there is a gat_1 robot at $(0,0)$ at $t = 0$. Since P_t is unfavorable for all $t > 0$, all gat_1 robots



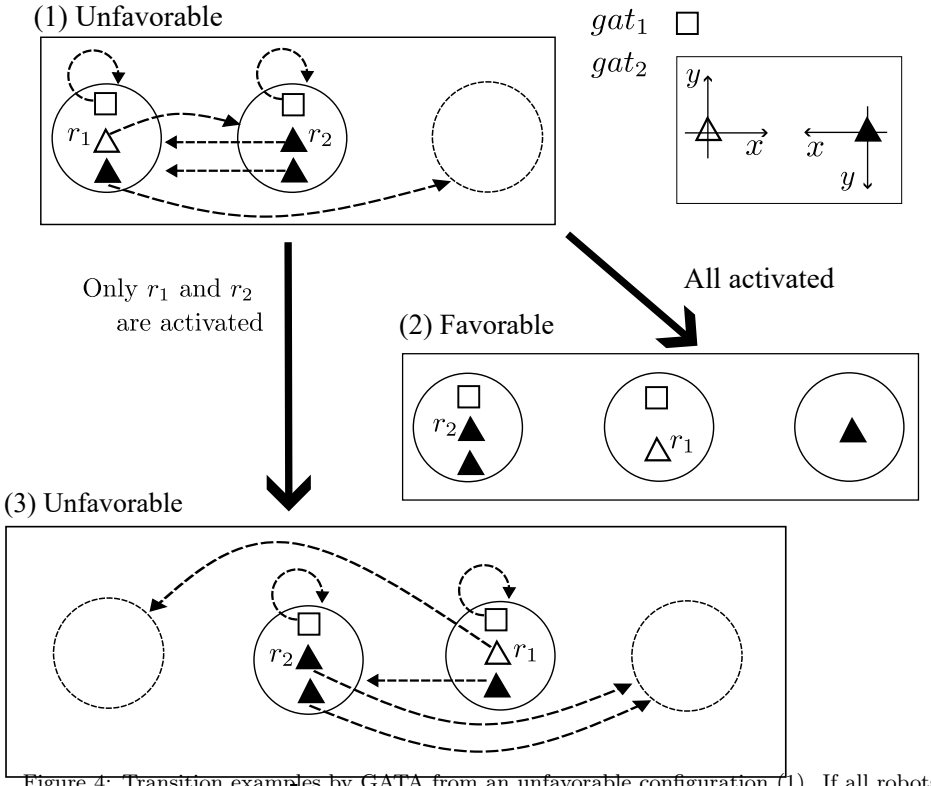


Figure 4: Transition examples by GATA from an unfavorable configuration (1). If all robots are activated, then a favorable configuration (2) yields. An unfavorable configuration (3) yields, only if r_1 and r_2 are activated. No matter which gat_2 robots are activated at (3), a favorable configuration yields. When e.g., r_1 and r_2 are activated, a favorable configuration in (4) yields.

do not move forever. Let $A_t(\boldsymbol{q})$ be the set of gat_2 robots at $\boldsymbol{q}$ activated at time t, and let $a_t(\boldsymbol{q}) = |A_t(\boldsymbol{q})|$.

Consider any robot $r_i \in A_0((0,0))$. If $(0,0) > \boldsymbol{q}$ $(\neq (0,0))$ in Z_i, then $(0,0), (2,0) \in \overline{P_1}$ by the definition of gat_2. Since P_1 is unfavorable, $\overline{P_1} = \{(0,0), (2,0)\}$, which implies that all robots at $(1,0)$ move to $(0,0)$ by the definition of gat_2. It is a contradiction, since $k_1 = 1$ because $\mu_1((0,0)) > h$.

Therefore, every robot $r_i \in A_0((0,0))$ satisfies $(0,0) < \boldsymbol{q}$ in Z_i, and move

to $(1,0)$, which implies $(0,0),(1,0) \in \overline{P_1}$. Since P_1 is unfavorable, the following three conditions hold: $\overline{P_1} = \{(0,0),(1,0)\}$, $a_0((1,0)) = a_0((0,0))$, and all robots $r_j \in A_0((1,0))$ move to $(0,0)$, each satisfying $(0,0) < \boldsymbol{q}$ in Z_j. Then, $P_0 = P_1$ holds.

We make a key observation here. At $t = 1$, each robot $r_i \in A_0((0,0))$ (resp. $r_j \in A_0((1,0))$) currently at $(1,0)$ (resp. $(0,0)$) satisfies $(0,0) > \boldsymbol{q}$ in Z_i (resp. in Z_j) Therefore, any robot r_i in $A_0((0,0))$ (resp. r_j in $A_0((1,0))$) is not included in $A_t((1,0))$ (resp. $A_t((0,0))$) as a member, as long as $P_0 = P_t$ holds. Thus there is a time $t > 0$ such that $a_t((0,0)) = 0$ holds. We consider the smallest such t. That is, $a_{t'}((0,0)) > 0$ for $0 \leq t' \leq t-1$, and $P_0 = P_t$ hold.

If there is a robot $r_j \in A_t((1,0))$ satisfying $(0,0) < \boldsymbol{q}$ in Z_j, then $a_t((0,0)) > 0$, which is a contradiction. Thus every robot $r_j \in A_t((1,0))$ satisfies $(0,0) > \boldsymbol{q}$ in Z_j, and moves to $(-1,0)$, which implies that $(-1,0),(0,0) \in \overline{P_{t+1}}$. Since P_{t+1} is unfavorable, $\overline{P_{t+1}} = \{(-1,0),(0,0)\}$, and all robots r_j at $(1,0)$ at time t are activated at t, and move to $(-1,0)$, each satisfying $(0,0) > \boldsymbol{q}$ in Z_j. Since P_{t+1} is unfavorable, $a_t((1,0)) = h$ and hence $t = 0$.

Without loss of generality, we may assume that $\overline{P_0} = \{(-1,0),(0,0)\}$, $\mu_0((-1,0)) = \mu_0((0,0)) = h$, all robots r_j at $(-1,0)$ are gat_2 robots, each of which satisfies $(0,0) < \boldsymbol{q}$ in Z_j.

By the same argument above, every robot $r_i \in A_t((0,0))$ satisfies $(0,0) < \boldsymbol{q}$ in Z_i, and moves to $(-1,0)$; r_i then satisfies $(0,0) > \boldsymbol{q}$ in Z_i at time $t+1$ (since r_i is now at $(-1,0)$). The same number of robots r_j are included in $A_t((-1,0))$, which should not include a robot r_j satisfying $(0,0) > \boldsymbol{q}$ in Z_j, and they all move to $(0,0)$; r_j then satisfies $(0,0) > \boldsymbol{q}$ in Z_j at time $t+1$ (since r_i is now at $(0,0)$).

Since there is a gat_1 robot at $(0,0)$ at $t = 0$, when the hth robot r_j at $(-1,0)$ (which satisfies $(0,0) < \boldsymbol{q}$ in Z_j) is activated at some time t, there is no robot at $(0,0)$ which can move to $(-1,0)$. It is a contradiction, since P_{t+1} is favorable. □

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

Proof. Proof is by contradiction. Suppose that $\Phi = \{\phi_1, \phi_2\}$ is a symmetric gathering algorithm.

Consider the following case of $n = 4$. Let $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$, where r_1 and r_2, and r_3 and r_4, are respectively clones. That is, r_1 and r_2 (resp. r_3 and r_4) share the same local coordinate system Z (resp. Z'), and r_1 and r_2 (resp. r_3 and r_4) take ϕ_1 (resp. ϕ_2) as their target function. In the initial configuration P_0, $\overline{P_0} = \{\boldsymbol{p}_0, \boldsymbol{q}_0\}$, $\boldsymbol{x}_0(r_1) = \boldsymbol{x}_0(r_2) = \boldsymbol{p}_0$, and $\boldsymbol{x}_0(r_3) = \boldsymbol{x}_0(r_4) = \boldsymbol{q}_0$ hold. Furthermore, the scheduler activates r_1 (resp. r_3), if and only if it activates r_2 (resp. r_4).

Let $\mathcal{E} : P_0, P_1, \ldots$ be an execution satisfying the above conditions. At any time t, $\boldsymbol{x}_t(r_1) = \boldsymbol{x}_t(r_2) = \boldsymbol{p}_t$, and $\boldsymbol{x}_t(r_3) = \boldsymbol{x}_t(r_4) = \boldsymbol{q}_t$, for some $\boldsymbol{p}_t$ and $\boldsymbol{q}_t$.

We consider the following scheduler: For any time t, unless they all meet at time $t+1$, the scheduler activates all robots. Otherwise, it nondeterministically selects either r_1 and r_2, or r_3 and r_4, and activates them. Since $\mathcal{E}$ transforms P_0 to a goal configuration, in which they all gather, there is a time instant t such that either $\boldsymbol{p}_{t+1} = \boldsymbol{p}_t$ and $\boldsymbol{q}_{t+1} = \boldsymbol{p}_t$, or $\boldsymbol{p}_{t+1} = \boldsymbol{q}_t$ and $\boldsymbol{q}_{t+1} = \boldsymbol{q}_t$.

Now, consider an initial configuration Q_0 such that $\boldsymbol{x}_0(r_1) = \boldsymbol{x}_0(r_3) = \boldsymbol{p}_t$ and $\boldsymbol{x}_0(r_2) = \boldsymbol{x}_0(r_4) = \boldsymbol{q}_t$. Then under the $\mathcal{FSYNC}$ scheduler, $Q_0 = Q_1 = \ldots$ hold, since Φ is symmetric. Thus Φ is not a gathering algorithm. □

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. SGATA is an extension of GATA (and 2GATA). Like GATA, $sgat_i = 2gat$ if a configuration is favorable, for $i = 1, 2, 3$. If a configuration is unfavorable, $sgat_i$ $(i = 1, 2, 3)$ are designed so that a favorable configuration must yield. Since target functions must be symmetric, SGATA needs a collaboration of three target functions (unlike two for GATA).

[Target function $sgat_1$]

1. If P is favorable, $sgat_1(P) = 2gat(P)$.
2. If P is unfavorable, then $m_P = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ be a point in $\overline{P}$ such that $\boldsymbol{q} \neq (0, 0)$. Since $(0, 0) \in \overline{P}$, $\boldsymbol{q}$ is uniquely determined. Then, $sgat_1(P) = -\boldsymbol{q}$.

[Target function $sgat_2$]

1. If P is favorable, $sgat_2(P) = 2gat(P)$.
2. If P is unfavorable, then $m_P = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ be a point in $\overline{P}$ such that $\boldsymbol{q} \neq (0, 0)$. Since $(0, 0) \in \overline{P}$, $\boldsymbol{q}$ is uniquely determined. Then $sgat_2(P) = -2\boldsymbol{q}$.

[Target function $sgat_3$]

1. If P is favorable, $sgat_3(P) = 2gat(P)$.
2. If P is unfavorable, then $m_P = 2$ and $k_P = 2$. Let $\boldsymbol{q}$ be a point in $\overline{P}$ such that $\boldsymbol{q} \neq (0, 0)$. Since $(0, 0) \in \overline{P}$, $\boldsymbol{q}$ is uniquely determined. Then $sgat_3(P) = -3\boldsymbol{q}$.

Figure 5(1) illustrates how each robot moves at an unfavorable configuration. As in the top right box, $sgat_1$, $sgat_2$, and $sgat_3$ robots are represented by square, triangle, and star, respectively. By the moves of robots in Figure 5(1), if a robot at $(1, 0)$ is activated, then a favorable configuration yields, regardless of which other robots are activated. Thus, a favorable configuration eventually yields by the fairness of the scheduler. Figure 5(2) illustrates a favorable configuration that yields, when one $sgat_2$ and one $sgat_3$ robots are activated. Theorem 4 formally shows this fact.

Theorem 4. SGTA *solves the* GAT *for* $n \geq 3$. *The MAS of symmetric algorithm for the* GAT *is hence* 3.

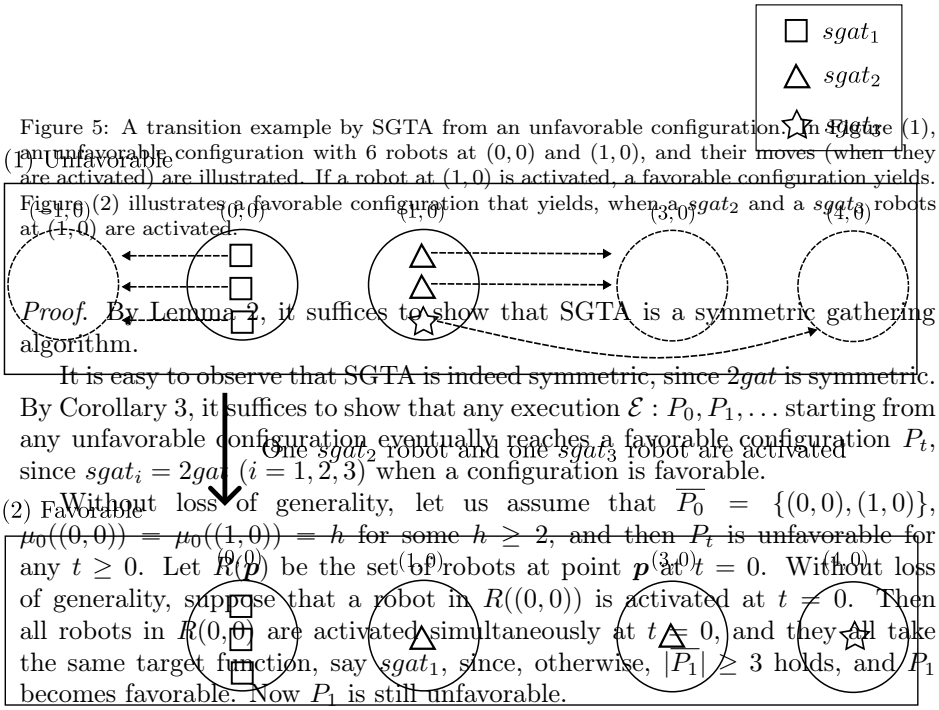


Figure 5: A transition example by SGTA from an unfavorable configuration. In figure (1), an unfavorable configuration with 6 robots at $(0,0)$ and $(1,0)$, and their moves (when they are activated) are illustrated. If a robot at $(1,0)$ is activated, a favorable configuration yields. Figure (2) illustrates a favorable configuration that yields, when a $sgat_2$ and a $sgat_3$ robots at $(1,0)$ are activated.

Proof. By Lemma 2, it suffices to show that SGTA is a symmetric gathering algorithm.

It is easy to observe that SGTA is indeed symmetric, since $2gat$ is symmetric. By Corollary 3, it suffices to show that any execution $\mathcal{E} : P_0, P_1, \ldots$ starting from any unfavorable configuration eventually reaches a favorable configuration P_t, since $sgat_i = 2gat$ $(i = 1, 2, 3)$ when a configuration is favorable.

Without loss of generality, let us assume that $\overline{P_0} = \{(0,0), (1,0)\}$, $\mu_0((0,0)) = \mu_0((1,0)) = h$ for some $h \geq 2$, and then P_t is unfavorable for any $t \geq 0$. Let $R(\boldsymbol{p})$ be the set of robots at point $\boldsymbol{p}$ at $t = 0$. Without loss of generality, suppose that a robot in $R((0,0))$ is activated at $t = 0$. Then all robots in $R(0,0)$ are activated simultaneously at $t = 0$, and they all take the same target function, say $sgat_1$, since, otherwise, $|\overline{P_1}| \geq 3$ holds, and P_1 becomes favorable. Now P_1 is still unfavorable.

By the fairness of the scheduler, there is a time t such that a robot in $R((1,0))$ is activated. Then all robots in $R((1,0))$ are activated simultaneously at t, again, because otherwise, $|\overline{P_{t+1}}| \geq 3$ holds. Observe that there are robots taking $sgat_2$ and $sgat_3$ in $R((1,0))$, and hence P_{t+1} is favorable since $|\overline{P_{t+1}}| \geq 3$.

□

5. Pattern formation problem

Let $\mathcal{P}^*$ be the set of multisets of points in R^2. Given a goal pattern $G \in \mathcal{P}^*$ consisting of n points in Z_0, the *pattern formation problem* (PF) for G is the problem of transforming any initial configuration I of n robots into a configuration similar to G. In what follows, without loss of generality, we assume $|G| = |I| = n$, when we consider the PF for a goal pattern G from an initial configuration I, since G is not formable from I, if $|G| \neq |I|$. The GAT is the PF for a goal pattern $G = \{(0,0),(0,0),\ldots,(0,0)\} \in \mathcal{P}^*$, and the SCT is reducible to the PF for a right n-gon.

Theorem 5 ([52]). *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 3. *The MAS for the* PF *is at least n.*

Proof. Let G be a right n-gon. If there was a pattern formation algorithm for G with size less $m < n$, the SCT could be solved by an algorithm of size $m < n$, which contradicts Theorem 1. □

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., [52]). 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. We describe an algorithm PFA $= \{pf_1, pf_2, \ldots, pf_n\}$ (for a swarm of n robots) which solves the PF for a given goal pattern G, based on 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) (see Figure 6, for illustrations):

(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 (Figure 6(1)):

  (1a) $P_1 = \{\boldsymbol{p}_1\}$ for some $\boldsymbol{p}_1 \in P$.

Figure 6: Two types of good configurations. Figure (1) illustrates a good configuration satisfying condition (1), and Figure (2) a good configuration satisfying condition (2). In Figure (2), in parentheses, we associate symbols $\boldsymbol{q}_1$, Q_2, and Q_3 with $\boldsymbol{p}_1$, P_2, and P_3, as defined in the proof of Lemma 4.

(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 $\boldsymbol{p}_3 = (31,0)$.[10] 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 (Figure 6(2)):

(2a) $P_1 = \{\boldsymbol{p}_1\}$.

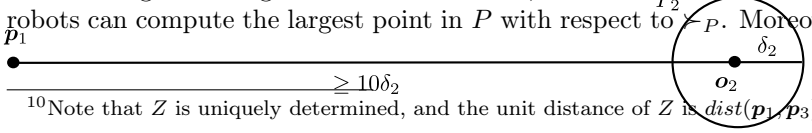
(2b) P_2 is a set (not a multiset).

(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 $H \subseteq G$.

If P is good, it is easy to observe that $\sigma(P) = 1$: Suppose that P satisfies condition (1). By definition, $|P_2| = |\overline{P_2}| \geq 3$. For any two points $\boldsymbol{q}, \boldsymbol{q'} \in P_2$, $dist(\boldsymbol{q}, \boldsymbol{q'}) \leq 2\delta_2$, and $dist(\boldsymbol{p}_1, \boldsymbol{q}) \geq 9\delta_2$. Thus, $\sigma(P) = 1$, since $k_P = 1$ and $\boldsymbol{o}_P \notin P$, where $\boldsymbol{o}_P$ is the center of the smallest enclosing circle of P. Suppose that P satisfies condition (2). Since $P_i \neq \emptyset$ for $i = 1,2,3$, $\boldsymbol{o}_2 = (10,0)$, $\boldsymbol{o_3} = (30,0)$, and C_i is a unit disk for $i = 1,2,3$, $k_P = 1$ and $\boldsymbol{o}_P \notin P$, which implies that $\sigma(P) = 1$.

Since a good configuration P has $k_P = 1$, $\succ_P$ is a total order on P, Hence robots can compute the largest point in P with respect to $\succ_P$. Moreover, when

(1)



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

(2)

P satisfies condition (2), and $|P_2| = |\overline{P_2}| \geq 2$, $\succ_{P'}$ is still a total order on P', where $P' = P \setminus P_3$. Hence, robots can also compute the largest point in P_2 with respect to $\succ_{P'}$. We make use of these facts to construct a PF algorithm PFA.

Lemma 4. *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.*

Proof. Consider any configuration P that satisfies condition (1) for a partition $\{P_1, P_2\}$. Suppose that P also satisfies condition (2). Then there is a partition $\{Q_1, Q_2, Q_3\}$ of P such that Q_1, Q_2, and Q_3 satisfy conditions (2a), (2b), and (2c), respectively.

We claim that $P_1 = Q_1$ and hence $P_2 = Q_2 \cup Q_3$ (see Figure 6(2)). Let $P_1 = \{\boldsymbol{p}_1\}$ and $Q_1 = \{\boldsymbol{q}_1\}$ Suppose $\boldsymbol{p}_1 \neq \boldsymbol{q}_1$ to derive a contradiction. Then, $\boldsymbol{q}_1 \in P_2$, which implies that every point in P_2 is within distance $2\delta_2$ from $\boldsymbol{q}_1$. Since $dist(\boldsymbol{p}_1, \boldsymbol{q}_1) \geq 9\delta_2$, $Q_3 = \{\boldsymbol{p}_1\}$, and hence $Q_2 = P \setminus \{\boldsymbol{p}_1, \boldsymbol{q}_1\}$.

Let $\boldsymbol{o}$ be the center of the smallest enclosing circle of Q_2. Obviously, $dist(\boldsymbol{q}_1, \boldsymbol{o}) \leq 2\delta_2$. In Z, $\boldsymbol{q}_1 = (0, 0)$, $\boldsymbol{p}_1 = (31, 0)$, and $\boldsymbol{o} = (10, 0)$. Since $dist(\boldsymbol{p}_1, \boldsymbol{q}_1) \geq 9\delta_2$, $dist(\boldsymbol{q}_1, \boldsymbol{o}) > 2\delta_2$, which is a contradiction.

Thus, P_2 can be partitioned into two multisets Q_2 and Q_3, and partition $\{P_1, Q_2, Q_3\}$ satisfies condition (2). However, it is impossible by the following reason. Consider the center $\boldsymbol{o}$ of the smallest enclosing circle of Q_2. By condition (1b), $dist(\boldsymbol{p}_1, \boldsymbol{o}) \geq 9\delta_2$. Since $P_2 = Q_2 \cup Q_3$, $dist(\boldsymbol{p}_3, \boldsymbol{o}) \leq 2\delta_2$. However, it is a contradiction, since $\boldsymbol{o}$ cannot be placed at $(10, 0)$ in Z.

By a similar argument, if P satisfies condition (2), then it does not satisfy condition (1).

It is easy to show that $\boldsymbol{p}_1$ is unique by a similar argument. □

We start with defining nSCTA$^* = \{sct_i^* : i = 1, 2, \ldots, n\}$, which is a slight modification of nSCTA, and is designed to yield a good configuration from any configuration. If a configuration P is not good and $P \neq \overline{P}$, we define $sct_i^* = sct_i$ for $i = 1, 2, \ldots, n$, so that nSCTA* can solve the SCT. Then, eventually a configuration P such that $P = \overline{P}$ yields. If a configuration is not good and $P = \overline{P}$, we move the robot with sct_1^* to point $\boldsymbol{p}_1$, so that a good configuration satisfying condition (1) can yield.

[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 $sct_1^*(P) = sct_1(P)$.
   - **(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 $\boldsymbol{o} = (0, 0)$, $\boldsymbol{p} = (10\delta, 0)$.

Figure 7: The target point $\boldsymbol{p}$ of robot r_1 with sct_1^*, when configuration P is not good and $P = \overline{P}$. The two figures (1) and (2) illustrate two cases in (b). In each cases, $dist(\boldsymbol{o}, \boldsymbol{p}) = 10\delta$ holds, and a good configuration satisfying condition (1) yields, when r_1 is activated. Here, r_1 represents the position of robot r_1 at P, and Z_1 is given in the right box.

(c) If $P = \overline{P}$ and $dist((0,0), \boldsymbol{o}) \geq 10\delta$, then $sct_1^*(P) = (0,0)$.

Let r_1 be the robot taking sct_1^*. When $P = \overline{P}$ and P is not good, all robots except r_1 can move, and a good configuration satisfying condition (1) yields, if r_1 is activated. To this end, sct_1^* defines $\boldsymbol{p}$, as $\boldsymbol{p}_1$, sufficiently far away from $\boldsymbol{o}$. Specifically, $\boldsymbol{p}$ is defined in (b) and (c) of sct_1^*, in such a way that $dist(\boldsymbol{o}, \boldsymbol{p}) \geq 10\delta$. Two cases in ($b$) are illustrated in Figure 7(1) and (2). Here, r_1 represents the position of robot r_1 at P.

Lemma 5. *nSCTA* is an algorithm to transform any initial configuration P_0 to a good configuration P.*

Proof. Consider any execution $\mathcal{E} : P_0, P_1, \ldots$ of nSCTA* from any initial configuration P_0, which is not good. By the proof (II) of Theorem 1, at some time t, $P_t = \overline{P_t}$ holds. If P_t is good, we have nothing to show.

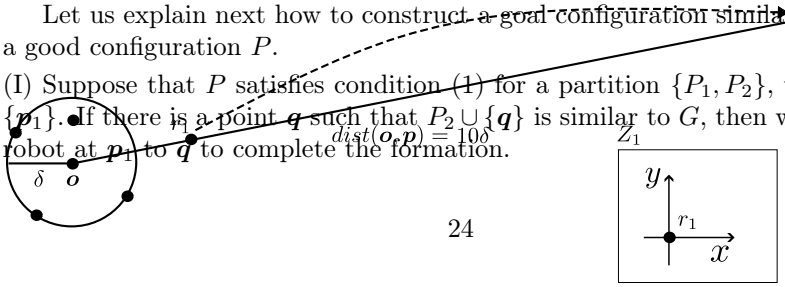
Suppose that P_t is not good. Let r_1 be the robot taking sct_1^* as the target function. The only robot that can move is r_1, and it computes $sct_1^*(P)$, where $P = \gamma_1^{-1}(P_t)$. By the definition of sct_1^*, $(P \setminus \{(0,0)\}) \cup \{\boldsymbol{p}\}$ is good (see Figure 7).

By definition, $\gamma_1((0,0)) = \boldsymbol{x}_t(r_1)$, P and P_t are similar, and no other robots move at time t. Thus $P_{t+1} = (P_t \setminus \{\boldsymbol{x}_t(r_1)\}) \cup \{\gamma_i(\boldsymbol{p})\}$, which is similar to $(P \setminus \{(0,0)\}) \cup \{\boldsymbol{p}\}$, although $\gamma_i(\boldsymbol{p})$ depends on Z_1 when $\boldsymbol{o}_2 = (0,0)$ in Z_1. Thus P_{t+1} is good. □

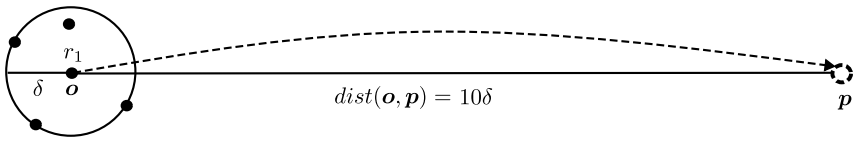
Let us explain next how to construct a goal configuration similar to G from a good configuration P.

(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.

(1)



(2)



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 $\{P_1, P_2 \setminus \{\boldsymbol{p}\}, \{\boldsymbol{p}_3\}\}$.

(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.

Suppose that a configuration P' yields from P. Then, P' has a partition $\{P_1, P_2 \setminus \{\boldsymbol{p}\}, P_3 \cup \{\boldsymbol{q}\}\}$, if $P_2 \setminus \{\boldsymbol{p}\} \neq \emptyset$. Since this partition clearly satisfies condition (2), P' is good (and satisfies condition (2)).

By repeating this transformation, $P_2 \setminus \{\boldsymbol{p}\}$ eventually becomes empty, and P' satisfies condition (1) with a partition (P'_1, P'_2), where $P'_1 = P_1$ and P'_2 is similar to a submultiset H of G. And, finally, the unique robot in P'_1 moves to a point $\boldsymbol{q}$ such that $P'_2 \cup \{\boldsymbol{q}\}$ is similar to G as in case (I).

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 local x-y coordinate system.)

Using points $\boldsymbol{p}$, $\boldsymbol{p}_3$, and $\boldsymbol{q}$ defined above, we describe a pattern formation algorithm 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 $\succ_{P_{12}}$; 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 $\succ_{P_{12}}$; otherwise, $pf_i(P) = (0,0)$.

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

Proof. By Lemma 3, the MAS for the PF is at least n.

On the other hand, PFA is an algorithm of size n for the PF, by the arguments above and the description of PFA. □

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.

Lemma 6. *For all* $1 \leq f \leq n-1$ *and* $2 \leq c \leq n$, *the* fFcS *is unsolvable, if* $c + f - 1 > n$.

Proof. Suppose that $f(> 0)$ faulty robots have crashed at the same point in P_0. For any configuration P reachable from P_0, $|\overline{P}| \leq n - f + 1$. Thus, if $c > n - f + 1$, then the fFcS is unsolvable. □

Since nF2S is unsolvable by Observation 1, we consider the fF2S for $1 \leq f \leq n-1$ in the next lemma.

Lemma 7.

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

Proof. (I) We first show that the MAS for the fF2S is at least $f + 2$. Proof is by contradiction. Suppose that there is an algorithm $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ of size $m < f + 2$ for the fF2S, to derive a contradiction.

Let $\boldsymbol{O} = \{(0,0), (0,0), \ldots, (0,0)\} \in \mathcal{P}_n$. We show that there is an i such that $\phi_i(\boldsymbol{O}) = (0,0)$. To derive a contradiction, suppose that $\phi_i(\boldsymbol{O}) = \boldsymbol{p}_i \neq (0,0)$ for $i = 1, 2, \ldots, m$. Consider any robot r_j and assume that its target function is ϕ_i. We choose the local x-y coordinate system Z_j of r_j that satisfies $\gamma_j(\boldsymbol{p}_i) = (1,0)$.

Let $P_0 = \boldsymbol{O}$ be an initial configuration. Then, under the $\mathcal{FSYNC}$ scheduler, since $P_1 = \{(1,0), (1,0), \ldots, (1,0)\}$, all robots r_j observe $\boldsymbol{O}$ and move to $(2,0)$

(in Z_0) by the definition of Z_j. Thus, by a simple induction, we have $P_t = \{(t,0),(t,0),\ldots,(t,0)\}$ and $|\overline{P_t}| = 1$ for all $t \geq 0$. It is a contradiction.

Without loss of generality, we assume that $\phi_m(\boldsymbol{O}) = (0,0)$. Consider the following assignment: For all $i = 1,2,\ldots,m-1$, robot r_i takes target function ϕ_i, and for all $j = m, m+1, \ldots, n$, robot r_j takes target function ϕ_m. We also assume that all robots $r_i (i = 1,2,\ldots,m-1)$ have crashed at time 0. Note that $m-1 \leq f$, because $m < f+2$.

Then under the $\mathcal{FSYNC}$ scheduler, $P_t = P_0$ for all $t \geq 0$. It is a contradiction. Thus, the MAS for the fF2S is at least $f+2$.

Since the MAS for the fF2S is less than or equal to n if fF2S is solvable, the $(n-1)$F2S is unsolvable (since the MAS is at least $n+1$).

(II) We next show that $(f+2)$SCTA solves the fF2S, if $f \leq n-2$. Proof is by contradiction. Consider any initial configuration P_0 satisfying $|\overline{P_0}| = 1$. We assume that $|\overline{P_t}| = 1$ for all $t \geq 0$, to derive a contradiction. There is a non-faulty robot r that does not take sct_1, and there is another robot r' that takes sct_1 by definition.

Since the $\mathcal{SSYNC}$ scheduler is fair, there is a time instant at which r is activated for the first time. Let $\overline{P_t} = \{\boldsymbol{p}\}$. Then $\boldsymbol{x}_{t+1}(r') = \boldsymbol{p}$ (regardless of whether it is faulty or not). On the other hand, $\boldsymbol{x}_{t+1}(r) \neq \boldsymbol{p}$, since $sct_i((0,0)) = (1,0)$ for all $i \neq 1$. Thus $|\overline{P_{t+1}}| \geq 2$. It is a contradiction.

Thus the MAS for the fF2S is $f+2$, if $f \leq n-2$. □

Lemma 8. *Suppose that* $1 \leq f \leq n-1$, $3 \leq c \leq n$, *and* $c+f-1 \leq n$. *Algorithm* $(c+f-1)$SCTA *solves the* fFcS.

Proof. First we show that $(c+f-1)$SCTA solves the fFcS, from any initial configuration P_0 satisfying $|\overline{P_0}| \geq 2$.

Let P_t be a configuration at time t. Suppose that $|\overline{P_t}| \geq 2$, and a non-faulty robot r_i is activated at t. By the proof of Theorem 1, if r_i and r_j have different target functions, then $\boldsymbol{x}_{t+1}(r_i) \neq \boldsymbol{x}_{t+1}(r_j)$, independently of their current positions $\boldsymbol{x}_t(r_i)$ and $\boldsymbol{x}_t(r_j)$, and regardless of whether or not r_j is activated. Thus $|\overline{P_t}| \leq |\overline{P_{t+1}}|$. To complete the proof, we claim: If $|\overline{P_t}| < c$, then $|\overline{P_t}| < |\overline{P_{t'}}|$ for some $t' > t$.

For any point $\boldsymbol{p} \in \overline{P_t}$, let $R(\boldsymbol{p})$ be the set of robots which reside at $\boldsymbol{p}$. If $R(\boldsymbol{p})$ includes two robots such that (1) both of them are non-faulty and have different target functions, or (2) exactly one of them is faulty, then the claim holds.

Since $(c+f-1)-f = c-1$, there are at least $c-1$ non-faulty robots whose target functions are distinct. By the argument above, there is no point $\boldsymbol{p} \in \overline{P_t}$ such that $R(\boldsymbol{p})$ includes two non-faulty robots having different target functions. Since $|\overline{P_t}| < c$, $|\overline{P_t}| = c-1$, and there is a non-faulty robot in each $R(\boldsymbol{p})$. It is because, otherwise, two non-faulty robots having different target functions reside at a point. Since $f \geq 1$, there is a point $\boldsymbol{p}$ such that $R(\boldsymbol{p})$ includes both non-faulty and faulty robots, and the claim hold.

Next we show that $(c+f-1)$SCTA solves the fFcS, from any initial configuration P_0. By Lemma 7, $(f+2)$SCTA solves the fF2S, if $f+2 \leq n$. Since

$c \geq 3$, $f+2 \leq c+f-1 \leq n$, which implies that $(c+f-1)$SCTA solves the fF2S by a similar argument to the proof (II) of Lemma 7. Thus, $(c+f-1)$SCTA solves the fFcS from any initial configuration. □

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

1. *If* $c = 2$, *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* $c+f-1 \leq n$.

Proof. The case of $c = 2$ holds by Lemma 7.

As for the case of $3 \leq c \leq n$, by Lemmas 6 and 8, it suffices to show that the fFcS is not solvable by an algorithm of size less than $c+f-1$, provided that $c+f-1 \leq n$ and $c \geq 3$.

Suppose that there is an algorithm of size $m < c+f-1$ for the fFcS, to derive a contradiction. Without loss of generality, we may assume that $m = c+f-2 \geq f+1$.

The proof is almost the same as the proof (I) of Theorem 1. Let $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ be any algorithm for the fFcS. 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 $(0,0)$, i.e., $\overline{P_0} = \{(0,0)\}$.
4. All robots r_i $(1 \leq i \leq f)$ have crashed at time 0.
5. The scheduler 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 faulty robots r_i $(1 \leq i \leq f)$ occupy $(0,0)$, for all $t \geq 0$, and all robots r_i $(f+1 \leq i \leq n)$ with the same target function occupy the same location, for all $t \geq 0$. Thus, $|\overline{P_t}| \leq m-f+1 = (c+f-2)-f+1 = c-1 < c$, which implies that Φ does not solve the fFcS, for $c \geq 3$. □

7. Fault tolerant gathering problems

7.1. At most one robot crashes

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. We omit f from the abbreviations to write FG and FGP when $f = 1$.

Theorem 8. *Provided that at most one robot crashes,* 2GATA *transforms any initial configuration into a configuration P satisfying one of the following conditions:*

1. $|\overline{P}| = 1$,
2. *n is even,* $\overline{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2\}$, *and* $\mu_P(\boldsymbol{p}_1) = \mu_P(\boldsymbol{p}_2) = n/2$, *or*
3. $\overline{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2\}$, $\mu_P(\boldsymbol{p}_1) = n - 1$, $\mu_P(\boldsymbol{p}_2) = 1$, *and the robot at $\boldsymbol{p}_2$ has crashed.*

Proof. We associate a triple $\lambda(P) = (k_P, m_P, -\mu_P)$ with a configuration P of n robots, where $m_P = |\overline{P}|$, and μ_P is $\mu_P(\boldsymbol{o}_P)$ if $k_P \geq 2$; otherwise, it is $\mu_P(\boldsymbol{p})$, where $\boldsymbol{p}$ is the largest point in $\overline{P}$ with respect to $\succ_P$. Let $<$ be the lexicographic order on $\mathbf{Z}^3$, i.e., $(a, b, c) < (a', b', c')$, if and only if (1) $a < a'$, (2) $a = a'$ and $b < b'$, or (3) $a = a'$, $b = b'$, and $c < c'$ holds, where $\mathbf{Z}$ is the set of integers.

For any given initial configuration P_0, let V be the set of configurations reachable from P_0, i.e., those which occur in an execution starting from P_0. We draw a directed graph $DG = (V, \vdash)$, which represents the transition relation defined by 2GATA starting from P_0, i.e., for any two distinct configurations $P, Q \in V$, $P \vdash Q$, if and only if Q is directly reached from P by activating some robots.

By the proof of Theorem 2, if $P \vdash Q$, then $\lambda(P) > \lambda(Q)$, and hence DG is a directed acyclic graph with sinks P_f, where $\lambda(P_f)$ is either $(0, 1, -n)$ or $(2, 2, 0)$.

Suppose that for any configuration P there are (at least) two distinct configurations Q and Q' such that $P \vdash Q$ and $P \vdash Q'$. Let $M(P, P')$ be the set of robots which move during the transition $P \vdash P'$. Since $Q \neq Q'$, $M(P, Q) \neq M(P, Q')$. We claim that even if a robot, say r, has crashed, there remains a transition $P \vdash Q''$ in DG. That is, we show that, for any robot r, there is a Q'' such that $P \vdash Q''$ and $r \notin M(P, Q'')$.

If $M(P, Q) \neq \{r\}$, then there is a robot $r' (\neq r) \in M(P, Q)$. Let Q'' be the configuration reachable from P by activating r' alone. Then $P \vdash Q''$ and $r \notin M(P, Q'')$. Otherwise, if $M(P, Q) = \{r\}$, since $M(P, Q) \neq M(P, Q')$, there is a robot $r' (\neq r) \in M(P, Q')$. By the same reason as above, there is a Q'' such that $P \vdash Q''$ and $r \notin M(P, Q'')$.

Consider an acyclic directed graph DG' constructed from DG by removing all transitions $P \vdash Q$ such that $r \in M(P, Q)$. Since we assume that every configuration P has at least two transitions in DG, there is a transition $P \vdash Q'$ in DG such that $r \notin M(P, Q)$ for all P, i.e., DG' still has sinks P_f.

However, there exists a configuration P having exactly one transition to another configuration. Consider a configuration P such that $\lambda(P) = (1, 2, -n + 1)$. P has exactly one transition; if $\overline{P} = \{\boldsymbol{p}_1, \boldsymbol{p_2}\}$, $\mu_P(\boldsymbol{p}_1) = n - 1$, and $\mu_P(\boldsymbol{p}_2) = 1$, then the unique robot at $\boldsymbol{p}_2$ moves to $\boldsymbol{p}_1$, and the other robot do not move even if they are activated. Recall that a configuration P such that $\lambda(P)$ is either $(0, 1, -n)$ or $(2, 2, 0)$ has no transition in DG. Except for these configurations, i.e., if $\lambda(P) \notin \{(1, 2, -n + 1), (0, 1, -n), (2, 2, 0)\}$, there are two transitions from P in DG, which is obvious from the definition of Steps 2 and 3 of $2gat$. Thus, DG' has sinks P'_f such that $\lambda(P'_f)$ is $(1, 2, -n + 1)$, $(0, 1, -n)$, or $(2, 2, 0)$.

If P such that $\lambda(P) = (1, 2, -n+1)$ has no transition in DG', then P satisfies $\overline{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2\}$, $\mu_P(\boldsymbol{p}_1) = n-1$, $\mu_P(\boldsymbol{p}_2) = 1$, and the robot at $\boldsymbol{p}_2$ has crashed. □

We have a corollary to Theorem 8.

Corollary 4. *If n is odd,* 2GATA *is an algorithm for the* FG*. Thus the MAS for the* FG *is 1, if n is odd.*

2GATA is not a gathering algorithm (and hence does not solve the FG), since it does not change an unfavorable configuration. The case in which n is even remains. Indeed, we have the following theorem.

Lemma 9. *The MAS for the* FG *is at least* 3*.*

Proof. Proof is by contradiction. Suppose that there is a fault tolerant gathering algorithm $\Phi = \{\phi_1, \phi_2\}$ of size 2.

We follow the proof of Lemma 2. Consider the case of $n = 4$. Let $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$. Then there is a configuration $P = \{\boldsymbol{p}, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{q}\}$ such that a ϕ_1 robot r at $\boldsymbol{p}$ does not move for some local x-y coordinate system Z and a ϕ_2 robot r' at $\boldsymbol{q}$ moves to $\boldsymbol{p}$ for some local x-y coordinate system Z'.

We consider the following configuration: Robots r_1 and r_2 with Z occupy $\boldsymbol{p}$. Robot r_3 occupies $\boldsymbol{q}$, whose local x-y coordinate system is a one constructed from Z by rotating π about the origin. Robots r_1, r_2, and r_3 take ϕ_1 as their target functions. Robot r_4 takes ϕ_2, occupies $\boldsymbol{q}$, and has crashed. Since all of them do not move, it is a contradiction. □

However, there is a symmetric algorithm of size 3 for the FG.

Theorem 9. SGTA $= \{sgat_1, sgat_2, sgat_3\}$ *solves the* FG *for $n \geq 3$. The MAS for the FG is* 3*.*

Proof. It suffices to show that SGTA solves the FG, since the MAS for the FG is at least 3 by Lemma 9. The proof is almost the same as that of Theorem 4.

By Theorem 8, it suffices to show that any execution $\mathcal{E} : P_0, P_1, \ldots$ starting from any unfavorable configuration eventually reaches a favorable configuration P_t, in the presence of at most one faulty robot. We follow the proof of Theorem 4. If there is a non-faulty robot which takes $sgat_i$ for $i = 1, 2, 3$, the proof of Theorem 4 correctly works as a proof of this theorem.

All what we need to show is the case in which a target function, say $sgat_1$, is taken by a unique robot r, and r crashes, so that there is no non-faulty robot taking $sgat_1$ after the crash. However, the same argument holds, since the "target point" of r, i.e., the current position, is different from the target points of the other robots. □

Let us next consider the FGP. The FGP is definitely not easier than the FG by definition. However, you might consider that the difference of difficulty between the FG and the FGP would be subtle, since the problem of converging all robots (including faulty ones) at a point in the presence of at most one faulty robot (the counterpart of the FGP in convergence) is solvable by a simple Goto-Gravity-Center algorithm (CoG) of size 1 [8].

Theorem 10. *The* FGP *is unsolvable. That is, the MAS for the* FGP *is* ∞.

Proof. To derive a contradiction, suppose that there is an algorithm Φ for the FGP. Consider any execution $\mathcal{E} : P_0, P_1, \ldots$ with initial configuration P_0 such that $|\overline{P_0}| > 1$ under a central schedule $\mathcal{S}$, which activates exactly one robot at a time. (Note that any central schedule is an $\mathcal{SSYNC}$ schedule.) Provided that no robots crash, $\mathcal{S}$ tries not to lead $\mathcal{E}$ to a goal configuration as long as possible (i.e., $\mathcal{S}$ is an adversary for Φ).

Since Φ is an algorithm for the FGP, no matter which $\mathcal{S}$ is given, there is a time t such that $\overline{P_t} = \{\boldsymbol{q}_1, \boldsymbol{q}_2\}$, $\mu_t(\boldsymbol{q}_1) = n - 1$, $\mu_t(\boldsymbol{q}_2) = 1$, and the destination point of each robot at $\boldsymbol{q}_1$ is $\boldsymbol{q}_1$.

If the robot at $\boldsymbol{q}_2$ crashes at t, then for all $t' > t$, $P_{t'} = P_t$, a contradiction. □

7.2. At most f robots crash

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 10 shows that fFGP is unsolvable when $f = 1$. As a corollary of Theorem 8, we have:

Corollary 5. 2GATA *solves the* 1F2GP. *The MAS for the* 1F2GP *is* 1.

SGTA transforms any goal configuration of the 1F2G into a goal configuration of the FG by Theorem 9, but the transformation of any goal configuration of the 1F2G into a goal configuration of the FGP is unsolvable by the proof of Theorem 10, which shows the difference between FG and FGP. For a configuration P_t, let $m_t = |\overline{P_t}|$, $k_t = k_{P_t}$, $\mu_t = \mu_{P_t}$, $C_t = C_{P_t}$, $\boldsymbol{o}_t = \boldsymbol{o}_{P_t}$, $CH_t = CH(P_t)$, and $\succ_t = \succ_{P_t}$.

Lemma 10. *For any* $1 \leq f \leq n - 1$, 2GATA *solves the* fFG *from any initial configuration* P_0 *satisfying* $k_0 = 1$.

Proof. Consider any initial configuration P_0 satisfying $k_0 = 1$, and let $\mathcal{E} : P_0, P_1, \ldots$ be any execution, provided that at most f robots crash. Let F_t be the set of faulty robots at time t. For any t, $F_t \subseteq F_{t+1}$. Let $F = \lim_{t\to\infty} F_t$. Then $|F| \leq f$.

By the definition of 2GATA, $k_t = 1$ for all t, and every non-faulty robots activated at t moves to $\boldsymbol{p}_0$, where $\boldsymbol{p}_0 \in \overline{P_0}$ is the largest point with respect to $\succ_0$. Note that $\boldsymbol{p}_0 \in \overline{P_t}$ and it is still the largest point in $\overline{P_t}$ with respect to $\succ_t$.

If $r \notin F$ and r is not at $\boldsymbol{p}_0$, then eventually r is activated and move to $\boldsymbol{p}_0$. Thus all non-faulty robots eventually gather at $\boldsymbol{p}_0$. □

Note that even if a configuration P_t such that $k_t = 1$ and $\mu_t(\boldsymbol{p_0}) = n - 1$ is reached, we cannot terminate 2GATA (to solve the fFG), since the robot not at $\boldsymbol{p_0}$ may be non-faulty.

Lemma 11. *For any $1 \leq f \leq n-1$, provided that at most f robots crash,* 2GATA *transforms any configuration P_0 into a configuration P satisfying one of the following conditions :*

1. *P is a goal configuration of the f*FG*, or*
2. *n is even, $\overline{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2\}$, and $\mu_P(\boldsymbol{p_1}) = \mu_P(\boldsymbol{p}_2) = n/2$.*

Proof. Consider any execution $\mathcal{E} : P_0, P_1, \ldots$ starting from any configuration P_0. Proof is by contradiction. By Lemma 10, to derive a contradiction, assume that $k_t \geq 2$ for all $t \geq 0$.

We follow the proof of Theorem 8, and construct a directed acyclic graph $DG = (V, \vdash)$ for P_0, where V is the set of configurations reachable from P_0 by 2GATA. It represents the transition relation between configurations, when no robots are faulty. However, any execution $\mathcal{E}$ in which some robots may crash is also represented by a path in DG from P_0.

Consider any configuration P_t such that $k_t \geq 2$. Let A_t be the set of robots not at $\boldsymbol{o}_t$, and let F_t be the set of faulty robots at t. If $A_t \subseteq F_t$, all non-faulty robots are at $\boldsymbol{o}_t$, and hence P_t is a goal configuration of the fFG.

If $A_t \not\subseteq F_t$, since there is a non-faulty robot r not at $\boldsymbol{o}_t$, there is a configuration Q such that $P_t \vdash Q$ and $\lambda(P_t) > \lambda(Q)$.

Since DG is a directed acyclic graph, eventually the fFG is solved (satisfying $A_t \subseteq F_t$), or P_t reaches a sink P_f of DG (since $(V, >)$ is a well-founded set). □

Theorem 11. 2GATA *solves both the f*F2G *and the f*F$(f+1)$GP*, for all $f = 1, 2, \ldots, n-1$. The MAS for each of the problems f*F2G *and f*F$(f+1)$GP *is 1, for all $f = 1, 2, \ldots, n-1$.*

Proof. Immediate by Lemmas 10 and 11. □

We already know that SGTA solves fFG if $f = 1$ by Theorem 9.

Theorem 12. SGTA *solves the f*FG *for all $f = 2, 3, \ldots, n-1$. The MAS for the f*FG *is* 3*.*

Proof. Since there is no algorithm of size 2 for the 2FG by Lemma 9, it suffices to show that SGTA solves fFG. Consider any execution $\mathcal{E} : P_0, P_1, \ldots$ starting from any configuration P_0. We show that eventually $\mathcal{E}$ reaches a goal configuration of fFG, provided that at most f robots have crashed. By Lemma 11, we may assume that P_0 is unfavorable, i.e., $m_0 = 2$ and $k_0 = 2$ hold. By the definition of SGTA, P_t is linear for all $t \geq 0$.

Suppose first that there is a time t such that P_t is (linear and) favorable, i.e., either $m_t \neq 2$ or $k_t \neq 2$ holds. Like the proof of Lemma 11, consider $DG = (V, \vdash)$ for P_t. Using the same argument as the proof of Lemma 11, eventually the fFG is solved (satisfying $A_t \subseteq F_t$), or P_t reaches a sink P_f of

DG (since $(V, >)$ is a well-founded set). By (I) of the proof of Theorem 2, the sink P_f is a goal configuration of fFG, i.e., $m_{P_f} = 1$.

Without loss of generality, we may assume that n is even, $\overline{P_0} = \{\boldsymbol{p}_1, \boldsymbol{p}_2\}$, and $\mu_0(\boldsymbol{p}_1) = \mu_0(\boldsymbol{p}_2) = n/2$. We assume that $m_{P_t} = 2$ and $k_{P_t} = 2$ for all $t \geq 1$, to derive a contradiction. We follow the proof of Theorem 9. Let R_i be the robots initially at $\boldsymbol{p}_i$ for $i = 1, 2$. Suppose that a non-faulty robot at $\boldsymbol{p}_1$, say r_1, is activated at time 0. Then all robots at $\boldsymbol{p}_1$ are non-faulty and must be activated simultaneously. Furthermore, they all take the same target function, say $sgat_1$, since otherwise $m_1 \geq 3$ holds. As long as the robots R_1 are activated, a configuration P_t satisfies $\overline{P_t} = \{\boldsymbol{q}_1, \boldsymbol{p}_2\}$ for some point $\boldsymbol{q}_1$, and $\mu_0(\boldsymbol{q}_1) = \mu_0(\boldsymbol{q}_2) = n/2$, where the set of robots in R_1 is at $\boldsymbol{q}_1$.

Since the scheduler is fair, let t be the first time instant that a non-faulty robot in R_2, say r_2, is activated. Then all robots at $\boldsymbol{p}_2$ are non-faulty and must be activated simultaneously. Furthermore, they all take the same target function. It is a contradiction, since both $sgat_2$ and $sgat_3$ are taken by some robots in R_2. Thus there is a time t such that P_t is linear and favorable, and the proof completes. □

Theorem 13. *The fFGP is unsolvable for $f = 2, \ldots, n-1$.*

Proof. A configuration P is a goal configuration if $m_P = |\overline{P}| \leq f$. Proof is by contradiction. 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.)

Since $\mathcal{E}$ does not reach a goal configuration forever, $\mathcal{S}$ violates the fairness, because Φ is an algorithm for fFGP. Let U be the set of robots which are not activated infinitely many times in $\mathcal{E}$. We assume that $\mathcal{S}$ makes U as small as possible.

We show that $|U| \leq f$. The proof is by contradiction. Suppose that $f < |U|$ to derive a contradiction. Let t_0 be a time such that any robot in U is not activated thereafter. The positions of the robots in U at t_0 is denoted by $Q = \{\boldsymbol{x}_{t_0}(r) \mid r \in U\}$. Then obviously $\overline{Q} \subseteq \overline{P_t}$ for all $t \geq t_0$. If $\boldsymbol{x}_{t_0}(r) = \boldsymbol{x}_{t_0}(r') \in Q$ for some $r, r' \in U$, then a contradiction to the maximality of A_{t_0} is derived; activating $A_{t_0} \cup \{r\}$ does not yield a goal configuration. Thus, exactly one robot in U resides at each $\boldsymbol{q} \in \overline{Q}$, and hence $|\overline{Q}| = |U| > f$.

Let r be any robot in U, and let $\boldsymbol{q}$ be its destination at time t_0. If $\boldsymbol{q} \notin \overline{Q}$, then a contradiction to the maximality of A_{t_0} is derived; activating $A_{t_0} \cup \{r\}$ does not yield a goal configuration, since $|\overline{(Q \setminus \{\boldsymbol{x}_{t_0}(r)\}) \cup \{\boldsymbol{q}\}}| \geq (|Q| - 1) + 1 = |U| > f$. Thus, $\boldsymbol{q} \in \overline{Q}$.

Consider a directed graph $G = (\overline{Q}, E)$, where $(\boldsymbol{p}, \boldsymbol{q}) \in E$, if and only if the destination of a robot $r \in U$ at $\boldsymbol{p}$ at time t_0 is $\boldsymbol{q}$. Recall that there is a bijection

between U and $\overline{Q}$. Since the outdegree of every vertex $\boldsymbol{p} \in \overline{Q}$ is 1, there is a directed loop $L \subseteq \overline{Q}$ (which may be a self-loop) in G. It is a contradiction to the maximality of A_{t_0}; activating $A_{t_0} \cup L$ does not yield a goal configuration. Thus, $|U| \leq f$.

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. □

8. Conclusions

We have proposed the minimum algorithm size (MAS) as an essential measure to measure the complexity of a problem, and showed an infinite hierarchy of the problems with respect to their MASs to justify this proposal; the set of problems with MAS being c is not empty for each integer $c > 0$ and ∞, and hence target function is a resource irreplaceable, e.g., with time. Then, under the $\mathcal{SSYNC}$ scheduler, we have established the MASs for the self-stabilizing gathering and related problems.

Main results are summarized in Table 1. For example, we have clarified the difference between two similar fault tolerant gathering problems, the FG and the FGP; the MAS for the FG is 3, while the one for the FGP is ∞. It might be interesting to observe that, for $2 \leq c \leq n$, the MAS is c for the cSCT and the one for the cGAT is 1. This fact is counter-intuitive, since gathering objects (which decreases entropy) seems to be a harder task than scattering objects (which increases entropy).

An open question asks how the time complexity decreases, as the number of target functions that an algorithm can use increases. It may be interesting if a kind of time acceleration theorem holds.

Another question asks the relation between the MAS of a problem Π and $\sigma = \min_{G \in \mathcal{G}} \sigma(G)$, where $\mathcal{G}$ is the set of Π's goal configurations and $\sigma(G)$ is the symmetricity of G. Observe that identifiers are mainly used to break symmetry, i.e., to decrease the symmetricity.

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

1. Complete Table 1 for the problems under the $\mathcal{FSYNC}$ scheduler.
2. What is the MAS for the gathering problem under the $\mathcal{ASYNC}$ scheduler?
3. For a fixed pattern G, what is the MAS for the pattern formation problem for G?
4. What is the MAS for the f-fault tolerant convergence problem to f points, for $f \geq 3$?
5. What is the MAS for the Byzantine fault tolerant gathering problem?
6. Characterize the problem whose MAS is 2.
7. Characterize the problem whose MAS is ∞.
8. For a homonymous distributed network with topology G, consider the number of identifiers necessary and sufficient to solve the leader election problem.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP22K11915 and JP24K02902.

References

[1] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1063–1071, 2004.

[2] K. Altisen, A. K. Datta, S. Devismes, A. Durand, and L. L. Larmore. Election in unidirectional rings with homonyms. *Journal of Parallel and Distributed Computing*, 146:79–95, 2010.

[3] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. Robotics and Automation*, 15:818–828, 1999.

[4] D. Angluin. Local and global properties in networks of processors. In *Proc. 12th ACM Symposium on Theory of Computing*, pages 82–93, 1980.

[5] D. Angluin, J. Aspens, Z. Diamadi, M. J. Fisher, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

[6] S. Arévalo, A. F. Anta, D. Imbs, E. Jiménez, and M. Raynal. Failure detectors in homonymous distributed systems (with an application to consensus). *Journal of Parallel and Distributed Computing*, 83:83–95, 2015.

[7] Y. Asahiro, I. Suzuki, and M. Yamashita. Monotonic self-stabilization and its application to robust and adaptive pattern formation. *Theoretical Computer Science*, 934:21–46, 2022.

[8] Y. Asahiro and M. Yamashita. Compatibility of convergence algorithms for autonomous mobile robots. arXiv: 2301.10949, 2023. (See also the extended abstract appeared in Proc. 30th Colloquium on Structural Information and Communication Complexity, Lecture Notes in Computer Science, 13892, pp. 149–164, 2023.).

[9] Y. Asahiro and M. Yamashita. Minimum algorithm sizes for self-stabilizing gathering and related problems of autonomous mobile robots (extended abstract). In *Proc. 25th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS2023), Lecture Notes in Computer Science*, volume 14310, pages 312–327, 2023.

[10] H. Attiya and M. Snir. Better computing on the anonymous ring. *J. Algorithms*, 12:204–238, 1991.

[11] H. Attiya, M. Snir, and M. K. Warmuth. Computing on the anonymous ring. *J. ACM*, 35(4):845–875, 1988.

[12] L. Barriére, P. Flocchini, P. Fraignaud, and N. Santoro. Capture of an intruder by mobile agents. In *Proc. 14th Symposium on Parallel Algorithms and Architectures*, pages 200–209, 2002.

[13] P. Boldi and S. Vigna. Computing vector functions on anonymous networks. In *Proc. 4th Colloquium on Structural Information and Communication Complexity*, pages 201–214, 1997.

[14] P. Boldi and S. Vigna. An effective characterization of computability in anonymous networks. In *Proc. International Symposium on Distributed Computing*, pages 33–47, 2001.

[15] Z. Bouzid, S. Das, and S. Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *Proc. IEEE 33rd International Conference on Distributed Computing Systems*, pages 337–346, 2013.

[16] S. Cicerone, G. Di Stefano, and A. Navarra. Asynchronous robots on graphs: gathering. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 184–217, 2019.

[17] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: gathering. *SIAM J. Computing*, 41:829–879, 2012.

[18] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM J. Computing*, 34:1516–1528, 2005.

[19] A. Cord-Landwehr, B. Degener, M. Fischer, M Hüllman, B. Kempkes, A. Klaas, P. Kling, S. Kurras, M. Märtens, F. Meyer auf der Heide, C. Raupach, K. Swierkot, D. Warner, C. Weddemann, and D. Wonisch. A new approach for analyzing convergence algorithms for mobile robots. In *Proc. 38th International Colloquimum on Automata, Languates, and Programming(ICALP2011), Part II, Lecture Notes in Computer Science*, volume 6756, pages 650–661, 2011.

[20] S. Das, P. Flocchini, N. Santoro, and M Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, 28:131–145, 2015.

[21] J. J. Daymude, K. Hinnenthal, A. W. Richa, and C. Scheideler. Computing by programmable particles. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 615–681, 2019.

[22] X. Défago, M. G. Potop-Butucaru, and S. Tixeuil. Fault-tolerant mobile robots. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 234–251, 2019.

[23] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, A. Kermarrec, E. Ruppert, and H. Tran-The. Byzantine agreement with homonyms. *Distributed Computing*, 26:321–340, 2013.

[24] C. Delporte-Gallet, H. Fauconnier, and H. Tran-The. Leader election in rings with homonyms,. In *Proc. International Conference on Networked Systems*, pages 9–24, 2014.

[25] Z. Derakhshandeh, S. Dolev, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Brief announcement: amoebot - a new model for programmable matter. In *Proc. 26th ACM Symposium on Parallel Algorithms and Architectures*, pages 220–222, 2014.

[26] Y. Dieudonné and F. Petit. Scatter of weak mobile robots. *Parallel Processing Letters*, 19(1):175–184, 2009.

[27] Y. Dieudonné and F. Petit. Self-stabilizing gathering with strong multiplicity detection. *Theoretical Computer Science*, 428:47–57, 2012.

[28] S. Dobrev, P. Flonicchi, Prencipe G, and N. Santoro. Mobile search for a black hole in an anonymous ring. In *Proc. 15th International Conference on Distributed Computing*, pages 166–179, 2001.

[29] S. Dobrev and A. Pelc. Leader election in rings with nonunique labels. *Fundamenta Informaticae*, 59(4):333–347, 2004.

[30] S. Dolev. *Self-stabilization.* MIT Press, 2000.

[31] P. Flocchini. Gathering. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 63–82, 2019.

[32] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots, Synthesis Lectures on Distributed Computing Theory 10.* Morgan & Claypool Publishers, 2012.

[33] 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 *Proc. 10th International Symposium on Algorithms and Computation(ISAAC1999), Lecture Notes in Computer Science*, volume 1741, pages 93–102, 1999.

[34] D. Ilcinkas. Oblivious robots on graphs: exploration. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 218–233, 2019.

[35] T. Izumi, S. Soussi, Y. Katayama, N. Inuzuka, X. Défago, K. Wada, and M. Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM J. Computing*, 41:26–46, 2012.

[36] B. Katreniak. Convergence with limited visibility by asynchronous mobile robots. In *Proc. 18th International Colloquium on Structural Information and Communication Complexity (SIROCCO2011), Lecture Notes in Computer Science*, volume 6786, pages 125–137, 2011.

[37] E. Kranakis, D. Krizanc, and J. van den Berg. Computing boolean functions on anonymous networks. In *Proc. 17th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science*, volume 443, pages 254–267, 1990.

[38] M. Lgothetis, G. Karras, L. Alevizos, C. Verginis, P. Roque, K. Roditakis, A. Mkris, S. Garcia, P. Schillinger, A. Di Fava, P. Pelliccione, A. Argyros, K. Kyriakopoulos, and D. V. Dimarogonas. Efficient cooperation of heterogeneous robotic agents: a decentralized framework. *IEEE, Trans. Robotics and Automation Magazine*, 28(2):74–87, 2021.

[39] Z. Liu, Y. Yamauchi, S. Kijima, and M. Yamashita. Team assembling problem for asynchronous heterogeneous mobile robots. *Theoretical Computer Science*, 721:27–41, 2018.

[40] G. A. Di Luna. Mobile agents on dynamic graphs. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 549–586, 2019.

[41] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[42] Y. Matias and Y. Afek. Simple and efficient election algorithms for anonymous networks. In *Proc. 3rd International Workshop on Distributed Algorithms (WDAG'89)*, pages 183–194, 1989.

[43] G. Prencipe. Pattern formation. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 37–62, 2019.

[44] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots – formation and agreement problems. In *Proc. 3rd Colloquium on Structural Information and Communication Complexity*, pages 313–330, 1996.

[45] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots – formation and agreement problems. *SIAM J. Computing*, 28:1347–1363, 1999.

[46] G. Viglietta. Uniform circle formation. In *Distributed Computing by Mobile Entities, Lecture Notes in Computer Science*, volume 11340, pages 83–108, 2019.

[47] M. Yamashita and T. Kameda. Computing on an anonymous network. In *Proc. 7th ACM Symposium on Principles of Distributed Computing*, pages 117–130, 1988.

[48] M. Yamashita and T. Kameda. Electing a leader when processor identity numbers are not distinct (extended abstract). In *Proc. 3rd International Workshop Distributed Algorithms (WDAG'89)*, pages 303–314, 1989.

[49] M. Yamashita and T. Kameda. Computing functions on asynchronous anonymous networks. *Mathematical Systems Theory*, 29:331–356, 1996.

[50] M. Yamashita and T. Kameda. Computing on anonymous networks I. Characterizing solvable cases. *IEEE Trans. Parallel and Distributed Systems*, 7(1):69–89, 1996.

[51] M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Trans. Parallel and Distributed Systems*, 10(9):878–887, 1999.

[52] M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411:2433–2453, 2010.

[53] Y. Yamauchi, T. Uehara, S. Kijima, and M. Yamashita. Plane formation by synchronous mobile robots in the three-dimensional euclidean space. *J. ACM*, 64:1–4, 2017.