Scatter of Weak Robots

Yoann Dieudonné, Franck Petit

Scatter of Weak Robots

Yoann Dieudonné Franck Petit
LaRIA, CNRS, Université de Picardie Jules Verne
Amiens, France

1 Introduction

In this paper, we first formalize the problem to be solved, i.e., the Scatter Problem (SP). We then show that SP cannot be deterministically solved. Next, we propose a randomized algorithm for this problem. The proposed solution is trivially self-stabilizing. We then show how to design a self-stabilizing version of any deterministic solution for the Pattern Formation and the Gathering problems.

In the next section, we describe the model considered in this paper and the formal definition of the problem to be solved, i.e., the *Scatter Problem*. Next, in Section 3, we consider how this problem can be solved. We first show that the Scatter Problem cannot be deterministically solved in the considered model. We then give a probabilistic algorithm for this problem along with its correctness proof. In Section 4, we put the result of Section 3 back in the context of distributed coordination of autonomous mobile robots. In this area, two classes of problem received a particular attention[1]:

1. The *Pattern Formation Problem* (PFP) which includes the *Circle Formation Problem*, e.g. [SY99, FPSW99, FPSW01b, DK02, Kat05, DLP06, DP07];
2. The *Gathering Problem* (GP), e.g., [AOSY99, SY99, FPSW01a, SY99, CP02, CFPS03].

We consider this two major classes of problems into self-stabilization settings. In a self-stabilizing system, regardless of the initial states of the computing units, is guaranteed to converge to the intended behavior in finite time [Dij74, Dol00]. To our best knowledge, all the above solutions assume that in the initial configuration, no two robots are located at the same position. As already noticed [DK02, Her06], this implies that none of them is self-stabilizing. In Section 4, we show that, being self-stabilizing, the proposed algorithm can be used to provide a self-stabilizing version of any deterministic solution for PFP and GP, i.e., assuming any arbitrary initial configuration—*including* configurations where two or more robots can be located at the same position. Finally, we conclude the story in Section 5.

2 Preliminaries

In this section, we define the distributed system, basic definitions and the problem considered in this paper.

[1] Note that some of the following solutions are in a model called CORDA [Pre01a] allowing more asynchrony among the robots than the semi-synchronous model (SSM) used in this paper. However, it is showed in [Pre01b] that any algorithm that correctly solves a problem P in CORDA, correctly solves P in SSM. So, any algorithm described in CORDA also works in SSM.

Distributed Model. We adopt the model introduced [SY96], in the remainder referred as SSM—stands for *Semi-Synchronous Model*. The *distributed system* considered in this paper consists of n mobile *robots* (*entity*, *agent*, or *element*) $r_1, r_2, \cdots, r_n$—the subscripts $1, \ldots, n$ are used for notational purpose only. Each robot r_i, viewed as a point in the Euclidean plane, moves on this two-dimensional space unbounded and devoid of any landmark. When no ambiguity arises, r_i also denotes the point in the plane occupied by that robot. It is assumed that the robots never collide and that two or more robots may simultaneously occupy the same physical location. Any robot can observe, compute and move with infinite decimal precision. The robots are equipped with sensors allowing to detect the instantaneous position of the other robots in the plane. Each robot has its own local coordinate system and unit measure. There is no kind of explicit communication medium. The robots implicitly "communicate" by observing the position of the others robots in the plane, and by executing a part of their program accordingly.

The considered robots are *uniform*, *oblivious*, and *anonymous*. The former indicates that they all follow the same program. Obliviousness states that the robots cannot remember any previous observation nor computation performed in any previous step. Anonymous means that no local parameter (such that an identity) which could be used in the program code to differentiate any of them.

In this paper, we also discuss some capabilities the robots are able to have or not:

Multiplicity Detection: The robots are able to distinguish whether there is more than one robot at a given position;

Localization Knowledge: The robots share a common coordinate system, i.e., a common Cartesian coordinate system with a common origin and common x-y axes with the same orientations.

Time is represented as an infinite sequence of time instant $t_0, t_1, \ldots, t_j, \ldots$ Let $P(t_j)$ be the set of the positions in the plane occupied by the n robots at time t_j ($j \geq 0$). For every t_j, $P(t_j)$ is called the *configuration* of the distributed system in t_j. $P(t_j)$ expressed in the local coordinate system of any robot r_i is called a *view*, denoted $v_i(t_j)$. At each time instant t_j ($j \geq 0$), each robot r_i is either *active* or *inactive*. The former means that, during the computation step (t_j, t_{j+1}), using a given algorithm, r_i computes in its local coordinate system a position $p_i(t_{j+1})$ depending only on the system configuration at t_j, and moves towards $p_i(t_{j+1})$—$p_i(t_{j+1})$ can be equal to $p_i(t_j)$, making the location of r_i unchanged. In the latter case, r_i does not perform any local computation and remains at the same position. In every single activation, the distance traveling by any robot r is bounded by σ_r. So, if the destination point computed by r is farther than σ_r, then r moves toward a point of at most σ_r. This distance may be different between two robots.

The concurrent activation of robots is modeled by the interleaving model in which the robot activations are driven by a *fair scheduler*. At each instant t_j ($j \geq 0$), the scheduler arbitrarily activates a (non empty) set of robots. Fairness means that every robot is infinitely often activated by the scheduler.

Specification. The *Scatter Problem* (SP) is to design a protocol for n mobile autonomous robots so that the following properties are true in every execution:

Convergence: Regardless of the initial position of the robots on the plane, no two robots are eventually located at the same position.

Closure: Starting from a configuration where no two robots are located at the same position, no two robots are located at the same position thereafter.

3 Algorithm

The scope of this section is twofold. We first show that,there exists no deterministic algorithm solving SP. The result holds even if the robots are not oblivious, share a common coordinate system, or are able to detect multiplicity. Next, we propose a randomized algorithm which converges toward a distribution where the robots have distinct positions.

3.1 Impossibility of a Deterministic Algorithm

Lemma 1 *There exists no deterministic algorithm that solves the Scatter Problem in SSM, even if the robots have the localization knowledge or are able to detect the multiplicity.*

Proof. Assume, by contradiction, that a deterministic algorithm A exists solving SP in SSM with robots having the localization knowledge and being able to detect the multiplicity. Assume that, initially (t_0), all the robots are located at the same position. So, it makes no matter whether the robots have the localization knowledge, are able to detect the multiplicity, or not, all the robots have the same view of the world. Assume that at t_0, all the robots are active and execute A. Since A is a deterministic algorithm and all the robots have the same view, then all the robots choose the same behavior. So, at time t_1, all of them share the same position on the place. Again, they all have the same view of the world. By induction, we can deduce that there exists at least one execution of A where the robots always share the same position. This contradicts the specification of SP. Hence, such an algorithm A does not exist. □

Note that Lemma 1 also holds whether the robots are oblivious or not. Indeed, assume non-oblivious robots, i.e., any robot moves according to the current and previous configurations. So, each robot r_i is equipped with a (possibly infinite) history register $\mathcal{H}_i$. At time t_0, for each robot r_i, the value in $\mathcal{H}_i$ depends on whether the registers are assumed to be initialized or not.

Assume first that, at t_0, $\mathcal{H}_i$ is initialized for every robot. Since the robots are assumed to be uniform and anonymous, the values stored in the history registers cannot be different. So, for every pair of robots $(r_i, r_{i'})$, $\mathcal{H}_i = \mathcal{H}_{i'}$ at t_0. Then, all the robots have the same view of the world. This case leads to the proof of Lemma 1.

Now, assume that, for every robot r_i, $\mathcal{H}_i$ is not assumed to be initialized at time t_0. Note that this case captures the concept of self-stabilization. In such a system, at t_0, one possible initialization of the history registers can be as follows: $(r_i, r_{i'})$, $\mathcal{H}_i = \mathcal{H}_{i'}$ for every every pair $(r_i, r_{i'})$. This case is similar to the previous case.

3.2 Randomized Algorithm

We use the following concept, *Voronoi diagram*, in the design of Algorithm 1.

Definition 2 (Voronoi diagram) *[Aur91, DK02] The Voronoi diagram of a set of points $P = \{p_1, p_2, \cdots, p_n\}$ is a subdivision of the plane into n cells, one for each point in P. The cells have the property that a point q belongs to the Voronoi cell of point p_i iff for any other point $p_j \in P$, $dist(q, p_i) < dist(q, p_j)$ where $dist(p, q)$ is the Euclidean distance between p and q. In particular, the strict inequality means that points located on the boundary of the Voronoi diagram do not belong to any Voronoi cell.*

We now give an informal description of Procedure SP, shown in Algorithm 1. Each robot uses Function *Random()*, which returns a value randomly and uniformally chosen over $\{0, 1\}$. When any

robot r_i becomes active at time t_j, it first computes the Voronoi Diagram of $P_i(t_j)$, i.e., the set of points occupied by the robots, $P(t_j)$, computed in its own coordinate system. Then, r_i moves toward a point inside its Voronoi cell $Cell_i$ whether $Random()$ returns 0.

Compute the Voronoi Diagram;
$Cell_i$:= the Voronoi cell where r_i is located;
$Current\_Pos$:= position where r_i is located;
if $Random()$=0
then Move toward an arbitrary position in $Cell_i$, which is different from $Current\_Pos$;
else Do not move;

Algorithm 1: Procedure SP, for any robot r_i.

Lemma 3 (Closure) *For any time t_j and for every pair of robots $(r_i, r_{i'})$ having distinct positions at t_j ($p_i(t_j) \neq p_{i'}(t_j)$), then by executing Procedure SP, r_i and $r_{i'}$ remains at distinct positions thereafter ($\forall j' > j,\ p_i(t_{j'}) \neq p_{i'}(t_{j'})$).*

Proof. Clearly, if at time t_j, r_i and $r_{i'}$ have distinct positions, then r_i and $r_{i'}$ are in two different Voronoi cells, V_i and V_j, respectively. From Definition 2, $V_i \cap V_j = \emptyset$. Furthermore, each robot can move only in its Voronoi cell. So, we deduce that r_i and $r_{i'}$ have distinct positions at time t_{j+1}. The lemma follows by induction on j', $j' > j$. □

In the following, we employ the notation $Pr[A] = v$ to mean that v is the probability that the event A occurs. Two events A and B are said to be *mutually exclusive* if and only if $A \cap B = \emptyset$. In this case, $Pr[A \cup B] = Pr[A] + Pr[B]$. The probability that an event A occurs given the known occurrence of an event B is the conditional probability of A given B, denoted by $Pr[A|B]$. We have $Pr[A \cap B] = Pr[A|B]Pr[B]$.

Lemma 4 (Convergence) *For any time t_j and for every pair of robots $(r_i, r_{i'})$ such that $p_i(t_j) = p_{i'}(t_j)$. By executing Procedure SP, we have*

$$\lim_{k \to \infty} Pr[p_i(t_{j+k}) \neq p_{i'}(t_{j+k})] = 1$$

Proof. Consider at time t_j, two robots r_i and $r_{i'}$ such that $p_i(t_j) = p_{i'}(t_j)$. Let X_{t_j} (respectively, Y_{t_j}) be the random variable denoting the number of robots among r_i and $r_{i'}$ which are activated (respectively, move). $Pr[X_{t_j} = z]$ (resp. $Pr[Y_{t_j} = z']$) indicates the probability that $z \in [0..2]$ (resp. $z' \in [0..2]$) robots among r_i and $r_{i'}$ are active (resp.move) at time t_j. Note that robot r_i (resp $r_{i'}$) can move only if r_i (resp $r_{i'}$) is active.

Both r_i and $r_{i'}$ are in a single position at time t_{j+1} only if one of the following three events arises in the computation step (t_j, t_{j+1}):

- **Event1**: "Both r_i and $r_{i'}$ are inactive." In this case:

$$Pr[Event1] = Pr[X_{t_j} = 0] \leq 1 \tag{1}$$

- **Event2**: "There is exactly one active robot which does not move and one inactive robot." Then, we have:

$$Pr[Event2] = Pr[X_{t_j} = 1 \cap Y_{t_j} = 0]$$

So,

$$Pr[Event2] = Pr[Y_{t_j} = 0 | X_{t_j} = 1] Pr[X_{t_j} = 1]$$

$$Pr[Event2] \leq \frac{1}{2} Pr[X_{t_j} = 1]$$

Thus,

$$Pr[Event2] \leq \frac{1}{2} \tag{2}$$

- **Event3:** "There are exactly two active robots and both of them move toward the same location." The probability that both robots are activated and move (not necessary at the same location) is given by:

$$Pr[X_{t_j} = 2 \cap Y_{t_j} = 2]$$

But,

$$Pr[X_{t_j} = 2 \cap Y_{t_j} = 2] = Pr[Y_{t_j} = 2 | X_{t_j} = 2] Pr[X_{t_j} = 2]$$

That is,

$$Pr[X_{t_j} = 2 \cap Y_{t_j} = 2] = \frac{1}{4} Pr[X_{t_j} = 2]$$

Thus,

$$Pr[X_{t_j} = 2 \cap Y_{t_j} = 2] \leq \frac{1}{4}$$

Since the probability that all the robots are activated and move (not necessary at the same location) is lower than or equal to $\frac{1}{4}$, the probability of Event3 (i.e both move toward the same location) is also lower than or equal to $\frac{1}{4}$, i.e.

$$Pr[Event3] \leq \frac{1}{4} \tag{3}$$

Let Ω be a sequence of time instants starting from t_j. Denote by k the number of time instants in Ω. The value a (resp. na) indicates the number of instant in Ω where at least one robot is active (resp. both r_i and $r_{i'}$ are inactive) among r_i and $r_{i'}$. Obviously, $a + na = k$. From the equations (2) and (3) and the fact that Event2 and Event3 are mutually exclusive, we have:

$$Pr[Event2 \cup Event3] = Pr[Event2] + Pr[Event3]$$

So,

$$Pr[Event2 \cup Event3] \leq \frac{1}{2} + \frac{1}{4} = \frac{3}{4} \tag{4}$$

From the equations (1) and (4), the probability that r_i and $r_{i'}$ are located at the same position after k time instant is

$$Pr[p_i(t_{j+k}) = p_{i'}(t_{j+k})] \leq (\frac{3}{4})^a Pr[Event1]^{na} \leq (\frac{3}{4})^a$$

By fairness, both r_i and $r_{i'}$ are infinitely often activated. Therefore, $\lim_{k \to \infty} a = \infty$, and then

$$\lim_{k \to \infty} Pr[p_i(t_{j+k}) = p_{i'}(t_{j+k})] = 0$$

The lemma follows from the fact that $Pr[p_i(t_{j+k}) \neq p_{i'}(t_{j+k})] = 1 - Pr[p_i(t_{j+k}) = p_{i'}(t_{j+k})]$. □

From Lemma 3 and 4 follows:

Theorem 5 *Procedure SP solves the Scatter Problem in SSM with a probability equal to 1.*

Note that as a result of Theorem 5 and by the specification of the Scatter Problem, Procedure SP provides a self-stabilizing solution in SSM.

4 Related Problems and Self-Stabilization

The acute reader should have noticed that by executing Procedure SP infinitely often, the robots never stop moving inside their Voronoi cells, even if no two robots are located at the same position. This comes from the fact that Procedure SP does not require robots having the multiplicity detection capability. Henceforth in this section, let us assume that the robots are equipped of such an ability. This assumption trivially allows the robots to stop if there exists no position with more than one robot. So, with the multiplicity detection, Procedure SP provides a valid initial configuration for every solution for PFP and GP. In the next two subsections, we show how Procedure SP can be used to provide self-stabilizing algorithms for PFP and GP.

4.1 Pattern Formation Problem

This problem consists in the design of protocols allowing the robots to form a specific class of patterns.

Let Procedure $A_{PF}(C)$ be a deterministic algorithm in SSM allowing the robots to form a class of pattern C. Algorithm 2 shows Procedure $SSA_{PF}(C)$, which can form all the patterns in C starting from any arbitrary configuration.

---

if there exists at least **one** position with a strict multiplicity
then SP;
else A_{PF};

Algorithm 2: Procedure $SSA_{PF}(C)$ for any robot r_i.

Theorem 6 *Procedure $SSA_{PF}(C)$ is a self-stabilizing protocol for the Pattern Formation Problem in SSM with a probability equal to 1.*

4.2 Gathering Problem

This problem consists to make $n \geq 2$ robots gathering in a point (not predetermined in advance) in finite time. In [Pre01b], it has been proved that GP is deterministically unsolvable in SSM and CORDA. In fact, one feature that the robots must have in order to solve GP is the multiplicity detection [SY99, CP02, CFPS03]. Nevertheless, even with the ability to detect the multiplicity, GP remains unsolvable, in a deterministic way, for $n = 2$ in SSM [SY99]. For all the other cases ($n \geq 3$), GP is solvable. So, when $n \geq 3$, the common strategy for solving GP is to combine two subproblems which are easier to solve. In this way, GP is separated into two distinct steps:

1. Starting from an arbitrary configuration wherein all the positions are distinct, the robots must move in such a way to create exactly one position with at least two robots on it;
2. Then, starting from there, all the robots move toward that unique position with a strict multiplicity.

As for the deterministic algorithms solving PFP, the deterministic algorithm solving GP ($n \geq 3$) requires that the robots are arbitrarily placed in the plane but with no two robots in the same position. Let Procedure A_{GP} be a deterministic algorithm solving GP, for $n \geq 3$, with multiplicity detection in SSM. Algorithm 3 shows Procedure SSA_{GP}, which solves GP with multiplicity detection starting from any arbitrary configuration whether $n \geq 3$. Remark that it is paradoxical that to make GP self-stabilizing, the robots must scatter before gathering.

---

if there exist at least **two** positions with a strict multiplicity
then SP;
else A_{GP};

Algorithm 3: Procedure SSA_{GP} for any robot r_i, $n \geq 3$.

Theorem 7 *Procedure SSA_{GP} is a self-stabilizing protocol for the Gathering Problem in SSM with a probability equal to 1 whether $n \geq 3$.*

Note that, for the case $n = 2$, we can provide a randomized algorithm solving GP. Informally, when any robot becomes active, it chooses to move to the position of the other robot with a probability $\frac{1}{2}$. By using a similar idea as in the proof of Lemma 4, we can prove that both robots eventually occupy the same position with a probability 1. By combining our basic routine for $n = 2$ with Procedure SSA_{GP}, we obtain a procedure which solves the self-stabilizing GP with multiplicity detection starting from any arbitrary configuration. It follows:

Theorem 8 *There exists a self-stabilizing protocol for the Gathering Problem in SSM with a probability equal to 1 for any $n \geq 2$.*

5 Conclusion

We shown that the Scatter Problem cannot be deterministically solved. We proposed a randomized self-stabilizing algorithm for this problem. We used it to design a self-stabilizing version of any deterministic solution for the Pattern Formation and the Gathering problems.

References

[AOSY99] H Ando, Y Oasa, I Suzuki, and M Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transaction on Robotics and Automation*, 15(5):818–828, 1999.

[Aur91] F Aurenhammer. Voronoi diagrams- a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[CFPS03] M Cieliebak, P Flocchini, G Prencipe, and N Santoro. Solving the robots gathering problem. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP 2003)*, pages 1181–1196, 2003.

[CP02] M Cieliebak and G Prencipe. Gathering autonomous mobile robots. In *9th International Colloquium on Structural Information and Communication Complexity (SIROCCO 9)*, pages 57–72, 2002.

[Dij74] EW Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the Association of the Computing Machinery*, 17:643–644, 1974.

[DK02] X Defago and A Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *2nd ACM International Annual Workshop on Principles of Mobile Computing (POMC 2002)*, pages 97–104, 2002.

[DLP06] Y Dieudonné, O Labbani, and F Petit. Circle formation of weak mobile robots. In *Eighth International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'06), LNCS 4280*, pages 262–275, 2006.

[Dol00] S. Dolev. *Self-Stabilization*. The MIT Press, 2000.

[DP07] Y Dieudonné and F Petit. Circle formation of weak robots and Lyndon words. *Information Processing Letters*, 104(4):156–162, 2007.

[FPSW99] 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 *10th Annual International Symposium on Algorithms and Computation (ISAAC 99)*, pages 93–102, 1999.

[FPSW01a] P Flocchini, G Prencipe, N Santoro, and P Widmayer. Gathering of autonomous mobile robots with limited visibility. In *STACS 2001*, pages 247–258, 2001.

[FPSW01b] P Flocchini, G Prencipe, N Santoro, and P Widmayer. Pattern formation by autonomous robots without chirality. In *VIII International Colloquium on Structural Information and Communication Complexity (SIROCCO 2001)*, pages 147–162, 2001.

[Her06] T Herman. Private communication, 2006.

[Kat05] B Katreniak. Biangular circle formation by asynchronous mobile robots. In *12th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2005)*, pages 185–199, 2005.

[Pre01a] G Prencipe. Corda: Distributed coordination of a set of autonomous mobile robots. In *ERSADS 2001*, pages 185–190, 2001.

[Pre01b] G Prencipe. Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots. In *ICTCS 2001*, pages 185–190, 2001.

[SY96] I Suzuki and M Yamashita. Agreement on a common x-y coordinate system by a group of mobile robots. *Intelligent Robots: Sensing, Modeling and Planning*, pages 305–321, 1996.

[SY99] I Suzuki and M Yamashita. Distributed anonymous mobile robots - formation of geometric patterns. *SIAM Journal of Computing*, 28(4):1347–1363, 1999.