Local Algorithms for Autonomous Robot Systems

Reuven Cohen[1] and David Peleg[2] *

[1] Dept. of Electrical and Computer Eng., Boston University, Boston, MA, USA
cohenr@bu.edu
[2] Dept. of Computer Science, Weizmann Institute, Rehovot, Israel
david.peleg@weizmann.ac.il

Abstract. This paper studies local algorithms for autonomous robot systems, namely, algorithms that use only information of the positions of a bounded number of their nearest neighbors. The paper focuses on the spreading problem. It defines measures for the quality of spreading, presents a local algorithm for the one-dimensional spreading problem, prove its convergence to the equally spaced configuration and discusses its convergence rate in the synchronous and semi-synchronous settings. It then presents a local algorithm achieving the exact equally spaced configuration in finite time in the synchronous setting, and proves it is time optimal for local algorithms. Finally, the paper also proposes an algorithm for the two-dimensional case and presents simulation results of its effectiveness.

1 Introduction

1.1 Background and motivation

Swarms of low cost robots provide an attractive alternative when facing various large-scale tasks in hazardous or hostile environments. Such systems can be made cheaper, more flexible and potentially resilient to malfunction. Indeed, interest in autonomous mobile robot systems arose in a large variety of contexts (see [3, 4, 11, 13–16, 21] and the survey [5]).

Along with developments related to the physical engineering aspects of such robot systems, there have been recent research attempts geared at developing suitable algorithmics, particularly for handling the distributed coordination of multiple robots [2, 6, 17, 19, 20]. A number of computational models were proposed in the literature for multiple robot systems. In this paper we consider the model of [2, 19]. (An alternative, weaker, model is found in [6, 9, 18].) In this model, the robots are assumed to be identical and indistinguishable, lack means of communication, and operate in discrete cycles. Each robot may wake up at each of these cycles, and make a move according to the locations of the other robots in the environment. The moves are assumed to be instantaneous.

* Supported in part by a grant from the Israel Science Foundation.

The model also makes the assumption that the robots are *oblivious*, i.e., have no memory and can only act according to a calculation based only on the their last observation of the world. Oblivious algorithms have the advantage of being self-stabilizing (i.e., insensitive to transient errors and to the addition or removal of robots) and are also usually simple to design and implement.

Algorithms were developed in the literature for a variety of control problems for robot swarms [2, 6, 9, 11, 17–20]. Most of those algorithms, however, require the robots to perform "global" calculations in each cycle, relying on the entire current configuration. For example, algorithms for the gathering problem, which requires the robots to gather at one point, typically instruct each robot to calculate the goal point to which it should move based on the exact locations of all the other robots in the current configuration. This calculation can be fairly simple and require only linear time (e.g., computing the average location of the robots), but in some cases involves a more complex computation, such as finding the smallest enclosing circle or the convex hull of the robots. Moreover, for very large swarms, and when each robot operates in fast short cycles, even a linear time computation per cycle may be too costly. A similar difficulty may arise when the task at hand involves coordinated movement in a certain direction while avoiding collisions, or evenly spreading the robot swarm in a given area.

The current paper addresses the issue of local and simple algorithms for control and coordination in robot swarms. It may be instructive to turn to the metaphor of insect swarms, and consider the way such problems are managed. It would appear that in a large swarm of bees, for instance, an individual bee would not calculate its next position based on the exact positions of all other bees. Rather, it is likely to decide its course of movement based on the positions and trajectories of a few nearby bees forming its immediate neighborhood. This local information is often sufficient to allow the individual insect to plan its movement so as to follow its swarm, avoid collisions and so on. Such policy would lead to a much "lighter" calculation, which can be carried out more frequently and effectively.

This analogy thus motivates the idea of exploring the behavior and performance of local, or "light-weight" algorithms for controlling a swarm of robots. Such algorithms have several obvious advantages over their more traditional "global" counterparts, in simplicity, computational complexity, energy consumption and stability. They also have the advantage of being applicable in a "limited visibility" model, where each robot sees only its close vicinity.

An enabling prerequisite is that each robot be equipped with a mechanism enabling it to efficiently obtain as input information about its locality, i.e., its immediate neighbors. (Clearly, if the robot's input device is designed so that its input consists a global picture of the configuration, and the robot's neighborhood can only be deciphered by going through this entire picture and performing complex calculations, then the robot might as well perform a global algorithm on its input). In particular, a robot equipped with a sonar may be able to first detect nearby objects. If the input to the robot is in the form of a visual image, then it may be necessary to have a scanning algorithm on the image, sweeping

the image from the point of the robot outwards, thus hitting the nearby robots first. The scan can be terminated after identifying the immediate neighbors. Hereafter we will ignore this issue, and assume that a suitable input mechanism exists, efficiently providing the robot in each cycle with a concise and accurate description of its immediate neighborhood.

In this paper we explore the "light-weight" approach to robot swarm control through the concrete example of the *robot spreading* problem. In this problem, N robots are initially distributed in the plane, and the goal is to spread them "evenly" within the perimeters of a given region. To consider methods for spreading one should first define a criterion for the quality of spreading of a given robot distribution. For the one-dimensional case this is easy, as the best configuration is clearly the equal spacing arrangement on the line, and all other configurations can be compared to this one. Quantitative measures for the spreading quality are presented in Section 2. In higher dimensions even the definition of the spreading quality becomes more difficult, as different criteria may be devised. One may consider, to list but a few examples, the minimum distance over all robot pairs, the average distance between each robot and its nearest neighbor, or the time needed to reach the point most distant from any robot. The choice of a definition may also depend on the motivation for the application of the spreading algorithm. The definition we use here is the average over all robots of the minimum distance to the nearest "object",

$$d_{\rm av} = \frac{1}{N} \sum_i \min_{j \neq i} \{d_{i,j}\} \ ,$$

where the objects considered in taking the minimum are all other robots and all points of the perimeter of the region.

We discuss two timing models for the robot operations. In the $\mathcal{FSYNC}$ (synchronous) model, it is assumed that the robots operate in cycles, where all robots operate in each cycle, i.e. take a snapshot of their surroundings, run their algorithm on the snapshot, and move accordingly. In the $\mathcal{SSYNC}$ (semi-synchronous) model, the robots are again assumed to operate in cycles. However, not all robots are active in every cycle. The activation schedule is assumed to be determined by an adversary, but it is guaranteed that every robot is activated an infinite number of times.

When the robots share a common orientation and the model is synchronous the task of spreading via a *global* algorithm is quite easy. A simple solution can be obtained by agreeing on an ordering (such as top-to-bottom and then left-to-right) and deciding on each robot's final location accordingly. This solution can be applied even in asynchronous settings if the movements of each robot are restricted so as to preserve the ordering. When only local information is used, it is difficult to prevent the swarm from converging to a non-optimal configuration, which is locally optimal for almost all robots. This phenomenon is well known in physical systems, e.g., such "defects" are known to determine many of the crystal's properties, as well as in artificial intelligence search, where finding the optimal state is difficult when many local minima exist.

Another advantage of global algorithms over local ones is the convergence rate or finishing time of the algorithm. It is easily seen that an algorithm based on ordering in a synchronous setting, will achieve the final position in a single step. A local algorithm will require at least $O(N)$ steps for the information to reach each of the robots, as proven below in Section 3.

In this paper we present an oblivious local algorithm for spreading in one dimension. We prove its validity for the synchronous and semi-synchronous models and discuss its convergence rate. We also present a non-oblivious local algorithm for spreading in finite (and optimal) time, and discuss some of its shortcomings. We then describe a generalization of the first algorithm to two dimensions and present simulation results showing its behavior. We also discuss several other alternatives and their relative strengths and weaknesses.

A related problem was studied by Dijkstra in [8]. There, n units labeled $\{0, 1, \ldots, n-1\}$ are initially placed around a ring, and apply a rule whereby unit i moves to the middle point between units $i-1$ and $i+1$ (modulo n). It is shown therein that under this rule, the system might in certain cases oscillate and fail to converge, even in a synchronous setting. A discussion of a possible solution to this problem appears in [7].

1.2 The model

The basic model studied in [2, 20] can be summarized as follows. The N robots execute a given algorithm in order to achieve a prespecified task. Time is divided to discrete events, where in each of these events each robot may or may not be active, under the condition that each robot is activated an infinite number of times. Whenever a robot is activated it takes a snapshot of its surroundings, which is used as the input for the algorithm executed by the robot. The output of the algorithm is a destination point, to which the robot moves instantaneously.

Following the common model in this area, the robots are assumed to be rather limited. To begin with, they have no means of directly communicating with each other. Moreover, they are assumed to be *oblivious* (or memoryless), namely, they cannot remember their previous states, their previous actions or the previous positions of the other robots. Hence the algorithm used in the robots' computations cannot rely on information from previous cycles, and its only input is the current configuration. As explained earlier, while this is admittedly an over-restrictive and unrealistic assumption, developing algorithms for the oblivious model has the advantages of self-stabilization (i.e., the ability to recover from any finite number of transient errors) and suitability to dynamical settings, where robots are added and removed during operation.

1.3 Preliminaries

We review some properties of discrete Fourier transforms to be used in our analysis later on.

The discrete sine transform: The discrete sine transform is appropriate for antisymmetric functions or alternatively, functions fixed to zero at both edges, which can be made antisymmetric by an appropriate continuation.

Consider the $N-2$ vectors $\bar{v}_k$ of dimension $N-2$ defined componentwise by

$$(v_k)_i \;=\; \sqrt{\frac{2}{N-1}}\sin ki, \quad i=1,2,\ldots,N-2\,,$$

for k values from the range $K=\{k=\frac{m\pi}{N-1} \mid m=1,\ldots,N-2\}$. We make use of the following known lemmas (proofs can be found in standard books on Fourier series or digital signal processing; see, e.g., [1]).

Lemma 1. *For $k,q\in K$, we have $\bar{v}_k\cdot\bar{v}_q=\delta_{k,q}$, where δ stands for Kroneker's delta, i.e., $\delta_{k,q}=1$ if $k=q$ and 0 otherwise.*

Corollary 1. *The $N-2$ vectors $\bar{v}_k$ form an orthonormal basis to $\mathcal{R}^{N-2}$.*

For a sequence of reals η_i, $i=0,\cdots,N-1$, satisfying $\eta_0=\eta_{N-1}=0$, and for $k\in K$, one can define the discrete Fourier transform of $\bar{\eta}$ as

$$\mu_k \;=\; \sqrt{\frac{2}{N-1}}\sum_{i=0}^{N-1}\eta_i\sin ki. \tag{1}$$

The inverse transform is given by the following lemma.

Lemma 2. $\eta_j \;=\; \sqrt{\frac{2}{N-1}}\sum_{k=\frac{\pi}{N-1}}^{\frac{(N-2)\pi}{N-1}}\mu_k\sin kj.$

The discrete cosine transform: The discrete cosine transform is widely used in digital signal processing. The version presented below is appropriate for symmetric functions, having a symmetry axis at $x=-1/2$.

Define the functions $f(i,k)=A(i,k)\cos\dfrac{k\hat{i}\pi}{m}$, for $i,k\in\{0,1,\ldots,2m-1\}$, where hereafter $\hat{i}=i+\frac{1}{2}$, and

$$A(i,k)=\begin{cases}\frac{1}{\sqrt{m}}\,, & k=0,\\ \frac{1}{\sqrt{2m}}\,, & \text{otherwise.}\end{cases}$$

We bring the following without proof.

Lemma 3. *(1)* $\sum_{i=0}^{2m-1}f(i,k)f(i,q)=\delta_{k,q}$, *(2)* $\sum_{k=0}^{2m-1}f(i,k)f(j,k)=\delta_{i,j}$.

Suppose now that η_i, $i=0,\ldots,2m-1$ is a sequence of numbers with $\eta_i=\eta_{2m-1-i}$, and define the transformed sequence $\mu_k \;=\; \sum_i\eta_i f(i,k)$ for $k=0,\ldots,2m-1$. Lemma 3 implies the following.

Corollary 2. $\eta_i=\sum_k\mu_k f(i,k)$.

Let $\phi(j,k,q) \;=\; \sum_{i=0}^{2m-1}f(i,k)f(i+j,q)$.

Lemma 4. $\phi(i,k,q)=\delta_{k,q}\cos\frac{jq\pi}{m}$.

2 A local spreading algorithm in one dimension

2.1 The algorithm

A swarm of N robots are positioned on a line. The aim is to spread the robots along this line with equal spacing between each pair of adjacent robots, where the size of the occupied segment is determined by the positions of the leftmost and rightmost robots. One may assume, instead, that the leftmost and rightmost positions represent some perimeter marks rather than robots. Each robot uses its own coordinate system. However, since the algorithm, presented below, is linear, any coordinate system will give the same resulting destination. Therefore, we use an external, global coordinate system, on which the robots have no knowledge. We refer to the robots according to their order on the line, and denote the position of each robot i, $0 \leq i \leq N-1$, at time t, in this global coordinate system by $R_i[t]$.

The local algorithm for spreading in constant distances operates as follows.

Algorithm `Spread` (Code for robot i):
If no other robot is seen on the left or on the right then do nothing.
Otherwise, move to the point $\frac{R_{i+1}+R_{i-1}}{2}$.

2.2 The $\mathcal{FSYNC}$ model

We now turn to prove the convergence of Algorithm `Spread`. We choose the global coordinate system such that $R_0[t] = 0$ and $R_{N-1}[t] = 1$ for all t (since the external robots do not move). As the goal is to spread the robots uniformly, upon termination the ith robot should be placed in position $i/(N-1)$. Define $\eta_i[t]$ as the *shift* of the ith robot's location at time t from its final designated position, namely,

$$\eta_i[t] \;=\; R_i[t] - \frac{i}{N-1} \;.$$

As our progress measure we define the quantity

$$\psi[t] \;=\; \sum_{i=0}^{N-1} \eta_i^2[t] \;,$$

where by definition, $\eta_0[t] = \eta_{N-1}[t] = 0$ for all t.

By executing the algorithm, the position of a robot changes to

$$R_i[t+1] \;=\; \frac{R_{i-1}[t] + R_{i+1}[t]}{2} \;,$$

hence for $1 \leq i \leq N-2$, the shifts change with time as

$$\eta_i[t+1] = \frac{\eta_{i+1}[t] + \eta_{i-1}[t]}{2}. \tag{2}$$

We now turn to prove our main lemma.

Lemma 5. *For N robots executing Algorithm* `Spread` *in the $\mathcal{FSYNC}$ model $\psi[t]$ is a decreasing function of t unless the robots are already equally spread (in which case it remains constantly zero).*

Proof: Eq. (2) gives the change in η in every time step. The value of ψ thus changes to

$$\psi[t+1] = \frac{1}{4}\sum_{i=1}^{N-2}(\eta_{i+1}[t] + \eta_{i-1}[t])^2 \ .$$

The decrease in ψ is therefore

$$\psi[t] - \psi[t+1] = \frac{1}{4}(\eta_1[t]^2 + \eta_{N-2}[t]^2) + \frac{1}{4}\sum_{i=1}^{N-2}(\eta_{i+1}[t] - \eta_{i-1}[t])^2 \ ,$$

which is a positive quantity, proving the lemma. ∎

Theorem 1. *For N robots executing Algorithm* `Spread` *in the $\mathcal{FSYNC}$ model:*

1. *Every $O(N)$ cycles, $\psi[t]$ is at least halved.*
2. *The robots converge to a point.*

Proof: The equations for the shift changes of the robots are given in Eq. (2) for all $0 < i < N-1$. Denote by μ_k the Fourier (Sine) series as defined in Eq. (1). By Eq. (1) and Lemmas 2 and 1,

$$\begin{aligned}\psi[t] = \sum_i \eta_i^2[t] \ &= \ \frac{2}{N-1}\sum_{i,k,q}\mu_k[t]\mu_q[t]\sin ki \sin qi \\ &= \frac{2}{N-1}\sum_{k,q}\mu_k[t]\mu_q[t]\frac{N-1}{2}\delta_{k,q} \ = \ \sum_k \mu_k^2[t]. \end{aligned} \tag{3}$$

Similarly, applying the transform (1) to the linear equation array (2) and noting that $\eta_0[t] = \eta_{N-1}[t] = 0$ for all times, t, yields

$$\begin{aligned}\mu_k[t+1] &= \sqrt{\frac{1}{2(N-1)}}\sum_{i=1}^{N-2}\sin ki(\eta_{i-1}[t] + \eta_{i+1}[t]) \\ &= \sqrt{\frac{1}{2(N-1)}}\sum_{j=0}^{N-3}\sin k(j+1)\eta_j[t] + \sqrt{\frac{1}{2(N-1)}}\sum_{j=2}^{N-1}\sin k(j-1)\eta_j[t] \\ &= \sqrt{\frac{1}{2(N-1)}}\sum_{j=0}^{N-1}\sin k(j+1)\eta_j[t] + \sqrt{\frac{1}{2(N-1)}}\sum_{j=0}^{N-1}\sin k(j-1)\eta_j[t],\end{aligned}$$

where the two terms added to each sum at the last line are zero as $\sin k \cdot (N-1) = \sin k \cdot 0 = 0$, and $\eta_0[t] = \eta_{N-1}[t] = 0$. Using the fact that $\sin k(i \pm 1) =$

$\sin ki \cos k \pm \sin k \cos ki$, we get

$$\begin{aligned}\mu_k[t+1] &= \sqrt{\frac{1}{2(N-1)}} \sum_{j=0}^{N-1} (\sin k(j+1) + \sin k(j-1))\eta_j[t] \\ &= \sqrt{\frac{1}{2(N-1)}} \sum_{j=0}^{N-1} 2 \sin kj \cos k\eta_j[t] \\ &= \cos k \cdot \mu_k[t].\end{aligned}$$

Hence, by Eq. (3),

$$\psi[t+1] \;=\; \sum_k \mu_k^2[t+1] \;=\; \sum_k (\cos k\mu_k[t])^2 \;=\; \sum_k \cos^2 k \cdot \mu_k^2[t]. \qquad (4)$$

The values of k are $k = \frac{\pi}{N-1}, \ldots, \frac{\pi(N-2)}{N-1}$. The largest factor is $\cos \frac{\pi}{N-1} \approx 1 - \frac{c}{N}$ for constant $c > 0$, hence by (4) and using (3) again, we get that

$$\psi[t+1] \;\leq\; \left(1 - \frac{c}{N}\right)^2 \sum_k \mu_k^2[t] \;\leq\; \left(1 - \frac{c}{N}\right)^2 \psi[t].$$

Thus, ψ is at least halved every $\frac{\ln 2}{2c} N$ steps, proving the claim. ∎

2.3 The $\mathcal{SSYNC}$ model

We now turn to the $\mathcal{SSYNC}$ model. In the $\mathcal{SSYNC}$ model we can no longer assume that all robots move at each time step. Therefore, when looking at the robots moving at some time step we can no longer assume that the robots at the boundary of the moving group are located at their final designated position. Thus, the subsequent analysis employs the cosine series rather than the sine series, since the cosines allow for a constant displacement term (because $\cos 0 \neq 0 = \sin 0$).

Our rational for proving convergence is as follows. We first introduce a non-decreasing quantity and prove its monotonicity. To prove that it decreases by a constant factor, we need to show that the presented quantity is related to the non-constant terms of the cosine series. This is done in Lemmas 7–9. Finally, in Theorem 2, we show that the non-constant terms are decreased by a constant factor on every round, proving convergence.

For $i = 1, \ldots, N-1$, define the robot *gaps* as the quantities

$$\gamma_i[t] \;=\; \eta_i[t] - \eta_{i-1}[t].$$

For each robot *moving* in the tth step, the equation for its position is given in (2). For any time step t, the moving robots can be grouped into chains, each of which is bounded by two stationary robots. Since the chains have no effect on each other, each can be handled separately.

Consider one such chain consisting of $m-1$ moving robots, which we number $1,\ldots,m-1$ regardless of their real numbering. The boundary stationary robots will be numbered 0 and m. The appropriate equations for the gaps between them are

$$\gamma_j[t+1] = \begin{cases} \frac{\gamma_1[t]+\gamma_2[t]}{2}\,, & j=1, \\ \frac{\gamma_{j+1}[t]+\gamma_{j-1}[t]}{2}\,, & j=2,\ldots,m-1, \\ \frac{\gamma_m[t]+\gamma_{m-1}[t]}{2}\,, & j=m. \end{cases}$$

To simplify this, we define virtual robots for every integer j outside the range $[0,N-1]$, and extend the definition of γ to every j outside $[0,N-1]$ by requiring that $\gamma_j=\gamma_{j+2m}$, and $\gamma_j=\gamma_{-j+1}$ for every j. Using this definition, $\gamma_0=\gamma_1$ and $\gamma_{m+1}=\gamma_{-m+1}=\gamma_m$. Therefore the equations take a simpler form

$$\gamma_j[t+1] = \frac{\gamma_{j+1}[t]+\gamma_{j-1}[t]}{2}, \quad \text{for every } j\,. \tag{5}$$

Define the Fourier transform of γ_j to be ϵ_k for $k=0,\ldots,2m-1$ satisfying

$$\gamma_j[t] \;=\; \sum_{n=0}^{2m-1} \epsilon_k[t] f(j,k)\;; \qquad \epsilon_k[t] \;=\; \sum_{n=0}^{2m-1} \gamma_j[t] f(j,k)$$

We now define the quantity $\varphi[t] = \sum_{j=1}^{N-1} \gamma_j^2[t]$.

Lemma 6. *In the $\mathcal{SSYNC}$ model $\varphi[t]$ is a non-increasing function of t.*

Proof: Apply the transform to Eq. (5) to obtain

$$\begin{aligned} \epsilon_k[t+1] &= \sum_{j=0}^{m-1} \gamma_j[t+1] f(j,k) \\ &= \sum_{j=0}^{2m-1} \frac{f(j,k)}{2} \left(\sum_{p=0}^{2m-1} f(j+1,p)\epsilon_p[t] + \sum_{p=0}^{2m-1} f(j-1,p)\epsilon_p[t] \right) \\ &= \sum_{p=0}^{2m-1} \cos\frac{p\pi}{m} \delta_{p,k} \epsilon_p[t] = \cos\frac{k\pi}{m}\epsilon_k[t]\,, \end{aligned} \tag{6}$$

where the last line uses Lemma 4. Now, from the orthogonality of the functions $f(j,k)$ (Lemma 3) it follows that

$$\sum_{j=0}^{m-1} \gamma_j^2[t] = \frac{1}{2} \sum_{j=0}^{2m-1} \gamma_j^2[t] = \frac{1}{2} \sum_{k=0}^{2m-1} \epsilon_k^2[t].$$

By Eq. (6),

$$\sum_{k=0}^{2m-1} \epsilon_k^2[t+1] = \sum_{k=0}^{2m-1} \epsilon_k^2[t] \cos^2\frac{k\pi}{m} \le \sum_{k=0}^{2m-1} \epsilon_k^2[t].$$

Since the sum of terms for the moving robots decreased, and the other terms were not affected, φ could not increase. ∎

For a chain of robots $0, \ldots, m$, define $\gamma_{\max}[t] = \max_{j \in \{1,\ldots,m-1\}} \gamma_j[t]$ and $\gamma_{\min}[t] = \min_{j \in \{1,\ldots,m-1\}} \gamma_j[t]$

Lemma 7. *For a chain of robots moving at time t, $[\gamma_{\min}[t+1], \gamma_{\max}[t+1]] \subseteq [\gamma_{\min}[t], \gamma_{\max}[t]]$.*

Proof: By Eq. (5) each new value is the average of two old ones. ∎

For the entire group of robots define Γ_i to be the ith smallest value of the γ_j's. $\Gamma_{\max}[t] = \Gamma_{N-1}[t] = \max_{j \in \{1,\ldots,N-1\}} \gamma_j[t]$ and $\Gamma_{\min}[t] = \Gamma_1[t] = \min_{j \in \{1,\ldots,N-1\}} \gamma_j[t]$.

Lemma 8. *There exists $0 < i < m$ such that $(\Gamma_i[t] - \Gamma_{i-1}[t])^2 \geq \frac{\varphi[t]}{N^3}$.*

Proof: By the definition, $\sum_{j=0}^{N-1} \gamma_j[t] = 0$, and therefore, $\Gamma_{\max}[t] \geq 0 \geq \Gamma_{\min}[t]$. Thus, for every i $\gamma_i[t] \leq \Gamma_{\max}[t] + \Gamma_{\min}[t]$ and $\varphi[t] \leq (N-1)(\Gamma_{\max}[t] + \Gamma_{\min}[t])$. Now, $\Gamma_{N-1}[t] - \Gamma_1[t] = \sum_{j=2}^{N-1} \Gamma_j[t] - \Gamma_{j-1}[t]$, so there must exist some i such that

$$\Gamma_i[t] - \Gamma_{i-1}[t] \;\geq\; \frac{\Gamma_{\max}[t] - \Gamma_{\min}[t]}{N-2} \;\geq\; \frac{\Gamma_{\max}[t] + \Gamma_{\min}[t]}{N-2} \,.$$

Therefore, $(\Gamma_i[t] - \Gamma_{i-1}[t])^2 \geq \frac{\varphi[t]}{N^3}$. ∎

Lemma 9. *For a chain of gaps, $0, \ldots, m-1$ with $\gamma_{\max}[t] - \gamma_{\min}[t] \geq c$, $\epsilon_0^2[t] + \frac{c^2}{2} \leq \sum_{k=0}^{2m-1} \epsilon_k^2[t]$.*

Proof: By definition $\epsilon_0^2[t] = \frac{1}{2m}(\sum \Gamma_i[t])^2 \leq \sum_i \gamma_i^2[t]$. Denote $\gamma_s[t] = \gamma_{\max}[t] = a + b$ and $\gamma_p[t] = \gamma_{\min}[t] = a - b$, with $b \geq c/2$. The sequence with $\gamma_s[t]$ and $\gamma_p[t]$ replaced by a will have the same $\epsilon_0[t]$ since the average remains the same. However

$$\sum_i \gamma_i^2[t] = \sum_{i \neq s,p} \gamma_i^2[t] + (a-b)^2 + (a+b)^2 = \sum_{i \neq s,p} \gamma_i^2[t] + 2a^2 + 2b^2 \geq \epsilon_0^2[t] + 2b^2.$$

Since $b \geq c/2$ the lemma follows. ∎

Theorem 2. *In the $\mathcal{SSYNC}$ model N robots executing Algorithm* **`Spread`** *will converge to a configuration with equal distances.*

Proof: For a time t take $\Delta[t] = \max_j(\Gamma_j[t] - \Gamma_{j-1}[t])$ and $g = \arg\max_j(\Gamma_j[t] - \Gamma_{j-1}[t])$. By Lemma 8, $\Delta^2[t] > \varphi[t]/N^3$. Define the sets of gaps $A = \{i | \gamma_i[t] \geq \Gamma_g[t]\}$ and $B = \{i \mid \gamma_i[t] \leq \Gamma_{g-1}[t]\}$. Suppose that $t' \geq t$ is the first time a robot i, such that one of its neighboring gaps is of set A and the other of set B makes a move. For as long as no robot sitting between a gap in A and a gap in B is activated no gap can leave either set, by Lemma 7 and therefore $\Gamma_g[t^*] - \Gamma_{g-1}[t^*] \geq \Gamma_g[t] - \Gamma_{g-1}[t]$ for $t \leq t^* \leq t'$.

Denote by $C = \{\ldots, i, i+1 \ldots\}$ the chain of gaps surrounding robot i that change at time t'. By Lemma 8 $|\Gamma_i[t'] - \Gamma_{i-1}[t']| \geq \frac{\sqrt{\varphi}[t']}{N^{3/2}}$ and therefore $\max(|\Gamma_i[t']|, |\Gamma_{i-1}[t']|) \geq \frac{\sqrt{\varphi}[t']}{2N^{3/2}}$, leading to $\max(\Gamma_i^2[t'], \Gamma_{i-1}^2[t']) \geq \frac{\varphi[t']}{4N^3}$. Now, $\sum_{i\in C} \gamma_i^2[t'] \geq \max(\Gamma_i^2[t'], \Gamma_{i-1}^2[t']) \geq \frac{\varphi[t']}{4N^3}$. Consider the change to $\sum_{i\in C} \gamma_i^2$ after step t'. Again, we number the gaps in C by $j = 0, \ldots, m-1$ and complete with virtual robots. We have

$$\sum_{j=0}^{2m-1} \gamma_j^2[t'+1] = \sum_{k=0}^{2m-1} \epsilon_k^2[t'+1] = \sum_{k=0}^{2m-1} \cos^2 \frac{k\pi}{m} \epsilon_k^2[t'].$$

For all $k > 0$, $\cos^2 \frac{k\pi}{m} \leq \cos^2 \frac{\pi}{m} = O\left(1 - \frac{1}{m^2}\right) \leq O\left(1 - \frac{1}{N^2}\right)$.

By Lemma 9 $\sum_{k\neq 0} \epsilon_k^2[t'] \geq \frac{\varphi[t']}{N^3}$ and by the above, whenever an appropriate robot makes a move the terms ϵ_k, $k \neq 0$ are decreased by $O(1/N^2)$, therefore, φ is decreased by $O(1/N^5)$ at timestep t'. By Lemma 6 $\varphi[t'] \leq \varphi[t]$. The theorem follows. ∎

3 An exact global algorithm in one-dimension

Assuming each robot knows the number of robots at each of its sides (i.e., robot $1 \leq i \leq N$ knows it is the ith robot in the line), it is possible to achieve the goal state after a finite number of steps using the following algorithm.

Algorithm `Fast_Spread` (Code for robot i):
1. $t \leftarrow 1$.
2. While $t < N-2$, move to the point $\frac{R_{i+1}+R_{i-1}}{2}$, and set $t \leftarrow t+1$.
3. If $t = N-2$ then solve the linear equation array Eq. (7) and move to the point $(x_1[0] + x_N[0])\frac{i-1}{N-1}$.

We assume each robots designates its coordinate center at the point where it starts the algorithm. We first define the equation array:

$$x_i[0] \;=\; 0\,; \qquad x_{i\pm1}[t] \;=\; \sum_{j=0}^{t} \binom{t}{j} \frac{x_{i\pm1+2j-t}[0]}{2^t} \tag{7}$$

We now show that Algorithm `Fast_Spread` guarantees reaching the wanted position after exactly $N-2$ moves.

Lemma 10. *The equation array (7) for a fixed $1 < i < N$, in conjunction with the data of the robot's neighbors $x_{i\pm1}[t]$ at times $t = 0, 1, \ldots, N-3$ provides a unique solution for $x_1[0], \ldots, x_N[0]$.*

Proof: Look at the set of equations for $x_i[0]$, $x_{i-1}[t]$ for $t = 0, 1, \ldots, i-2$ and $x_{i+1}[t]$ for $t = 0, 1, \ldots, N-i-1$. This set includes exactly N equations in N unknowns. The equations are independent since for each time t_0 there is a nonzero coefficient that was zero for all times $t < t_0$. Therefore, a unique solution exists. ∎

Theorem 3. *In the $\mathcal{FSYNC}$ model N robots executing algorithm* `Fast_Spread` *will reach their exact final position after $N-2$ steps.*

Proof: By Lemma 10 each robot can deduce the position of all other robots after $N-3$ steps. Afterwards it can solve an array of linear equations (7) and deduce its final position, which it will assume in the last step. ∎

Theorem 4. *In the SYN model the algorithm* `Fast_Spread` *achieves the fastest possible convergence to the final position.*

Proof: Since each robot can only see its nearest neighbors, and no communication is allowed, no information of the positions of the external robots can move more than one robot per move. At the beginning, each robot has information on its own position and its nearest neighbors. After the j step it has information on its $j+1$st nearest neighbors. Therefore, at least $N-3$ steps are needed for the 2nd and $N-2$st robots to get information on the position of the most distant robots, and another move to achieve their final position. ∎

Notice, however, that the coefficient of $\eta_j[0]$ in the linear equations obtained by robot k is proportional to $2^{-|j-k|}$. Therefore, the information accuracy decays exponentially quickly with the distance, and thus is hardly usable in any reasonable model of finite accuracy robots.

4 Local spreading in two dimensional space

In two dimensions the task of spreading via a local algorithm becomes more complicated. The set of nearest neighbors is of undetermined size, as even with no articulation points, a robot may have all other robots at an equal distance. Furthermore, the boundaries of the region in which the robots may spread cannot be efficiently marked by robots in a local manner. To simplify the situation we assume that the robots are confined to the region $[0,1]\times[0,1]$, and that the walls of this region are visible by the robots and serve as detractors. Furthermore, we assume the robots share the same orientation, where the axes parallel the walls. The algorithm is based on each robot, i, dividing space into four quadrants, Q_0 to Q_3, according to the orientation (see Fig. 1). In our simulation the quadrant boundaries were taken to lay in an angle of 45° to the axes to simplify the treatment of walls. This, assumption, however, should not be relevant to the results. Objects situated on the dividing lines may be considered, for instance, to belong to the lower numbered quadrant.

To illustrate the behavior of local algorithms for achieving spreading in two dimensions, we introduce an algorithm, named `Spread_2d`, in the spirit of the 1-dimensional Algorithm `Spread` presented above, and present empirical results indicating that this algorithm converges to a good approximation of the equal spacing spreading. Let us remark that an inetresting alternative approach to the problem follows from reversing the gathering technique presented in [10], for a slightly different continuous model.

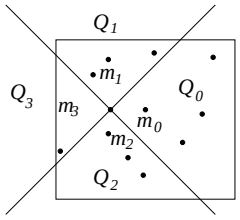


Fig. 1. The four quadrants of a robot's view. Here $m_q = m_2$.

Algorithm `Spread_2d` (Code for robot i):
For $j = 0, \ldots, 3$ **do:**
(a) $m_j \leftarrow$coordinate of nearest robot or perimeter point in quadrant Q_j.
(b) $d_j \leftarrow dist(i, m_j)$.
$q \leftarrow \arg\min_j \{d_j\}$; $d_{min} = \min_j \{d_j\}$; $d_{opp} = d_{3-q}$;
Move away from current location by $\frac{d_{min} - d_{opp}}{2d_{min}} m_q$.

Notice that in Algorithm `Spread_2d` an object may be either a robot or (the nearest point of) the perimeter.

To study the behavior of Algorithm `Spread_2d` simulations were conducted under various circumstances. We define the parameter

$$d_{av} = \frac{1}{N} \sum_i \min_{j \neq i} \{d_{i,j}\} \quad ,$$

where the minimum is taken over all other robots and all points on the perimeter of the region. The behavior of this parameter, used as an indication of the spreading efficiency, was studied as a function of time. In an equally spaced square grid formation of the robots, the value of the parameter is $d_{av}^{opt} = \frac{1}{\sqrt{N}+1}$. This value can be used as an indication of the optimal spreading.

Fig. 2 presents the behavior of d_{av} under Algorithm `Spread_2d` as a function of time. The optimal value for this number of robots is $d_{av}^{opt} = 0.1$ and, as can be seen, the system saturates at a value close to the optimum. Fig. 3 presents the locations of the robots at the initial and final states of the algorithm application.

One may consider other generalizations of the one-dimensional algorithm to two dimensions. One such possible algorithm is based on moving to the average position of the four nearest robots (one in each quadrant). Our experiments show that this algorithm saturates quickly to a configuration which may sometimes be very far from the optimal. Another possible algorithm may rely on each robot calculating its Voronoi cell and moving to its center (according to some measure). This algorithm has the problem of failing the locality criterion, since in the worst case some of the robots may need to consider $\Omega(N)$ other robots in order to calculate their Voronoi cell. The analysis of this algorithm is also difficult, since the robots movement may change the entire structure of the Voronoi diagram such that the Voronoi neighbors of a robot may change from cycle to cycle.

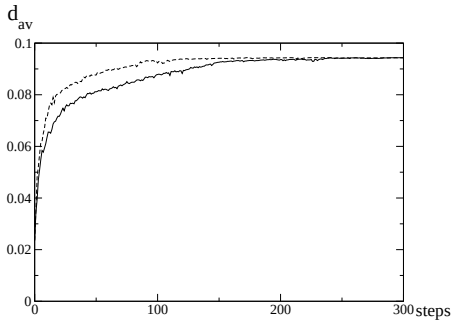
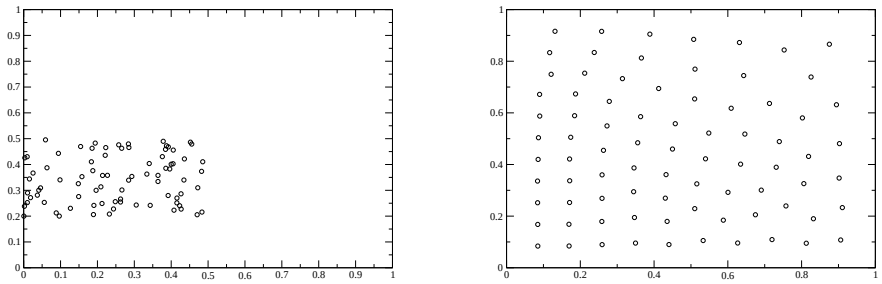


Fig. 2. Results of the simulation of algorithm `Spread_2d` with $N = 81$ robots. The dashed line is for the $\mathcal{FSYNC}$ timing model. The solid line is for the $\mathcal{SSYNC}$ timing model with each robot having probability 1/2 for moving at each step.



(a) (b)

Fig. 3. (a) The initial location of the robots, selected randomly in the region $[0, 0.5] \times [0.2, 0.5]$. (b) The final locations of the robots, taken after 1000 steps.

An approximation for this method may be obtained by using θ-graphs [12], which guarantees that each robot has exactly $2\pi/\theta$ neighbors, thus guaranteeing locality. The algorithm `Spread_2d` may be considered as such an approximation with $\theta = \pi/2$.

It should also be noted that Algorithm `Spread_2d` presented above, as well as any other algorithm, will fail to break symmetry when starting from some highly symmetrical configurations such as a line or a circle, which are very far from the optimal configuration. This problem can be overcome by adding randomness to the algorithm.

References

1. Ali N. Akansu and Richard A. Haddad. *Multiresolution Signal Decomposition.*

Academic Press, San Diego, CA, USA, 1992.
2. H. Ando, I. Suzuki, and M. Yamashita. Formation and agreement problems for synchronous mobile robots with limited visibility. In *Proc. IEEE Symp. of Intelligent Control*, pages 453–460, August 1995.
3. T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Trans. on Robotics and Automation*, 14, December 1998.
4. G. Beni and S. Hackwood. Coherent swarm motion under distributed control. In *Proc. DARS'92*, pages 39–52, 1992.
5. Y.U. Cao, A.S. Fukunaga, and A.B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, March 1997.
6. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem. In *Proc. 30th Int. Colloq. on Automata, Languages and Programming*, pages 1181–1196, 2003.
7. X. Defago and A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In *Proc. 2nd ACM Workshop on Principles of Mobile Computing*, pages 97–104. ACM Press, 2002.
8. Edsger W. Dijkstra. *Selected Writings on Computing: A Personal Perspective.* Springer, New York, 1982. pages 34-35.
9. 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 Int. Symp. on Algorithms and Computation*, 93–102, 1999.
10. N. Gordon, I.A. Wagner, and A.M. Bruckstein. Gathering multiple robotic a(ge)nts with limited sensing capabilities. In *Proc. 4th Int. Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 142–153, September 2004.
11. D. Jung, G. Cheng, and A. Zelinsky. Experiments in realising cooperation between autonomous mobile robots. In *Proc. Int. Symp. on Experimental Robotics*, 1997.
12. J.M. Keil and C.A Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete computational Geometry*, 7:13–28, 1992.
13. M.J. Mataric. *Interaction and Intelligent Behavior.* PhD thesis, MIT, 1994.
14. L.E. Parker. Designing control laws for cooperative agent teams. In *Proc. IEEE Conf. on Robotics and Automation*, pages 582–587, 1993.
15. L.E. Parker. On the design of behavior-based multi-robot teams. *J. of Advanced Robotics*, 10, 1996.
16. L.E. Parker, C. Touzet, and F. Fernandez. Techniques for learning in multi-robot teams. In T. Balch and L.E. Parker, editors, *Robot Teams: From Diversity to Polymorphism.* A. K. Peters, 2001.
17. G. Prencipe. CORDA: Distributed coordination of a set of atonomous mobile robots. In *Proc. 4th European Research Seminar on Advances in Distributed Systems*, pages 185–190, May 2001.
18. G. Prencipe. *Distributed Coordination of a Set of Atonomous Mobile Robots.* PhD thesis, Universita Degli Studi Di Pisa, 2002.
19. K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *J. of Robotic Systems*, 13(3):127–139, 1996.
20. I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. on Computing*, 28:1347–1363, 1999.
21. I.A. Wagner and A.M. Bruckstein. From ants to a(ge)nts. *Annals of Mathematics and Artificial Intelligence*, 31, special issue on ant-robotics:1–5, 1996.