

THE DISTRIBUTED COMPUTING COLUMN

BY

STEFAN SCHMID

University of Vienna

Währinger Strasse 29, AT - 1090 Vienna, Austria

schmiste@gmail.com

With this issue of the distributed computing column, we would like to invite you to two tours, one to exciting unexplored borderlands of the gathering problem, and one to the wonderful land of consensus numbers:

- El Mahdi El Mhamdi, Rachid Guerraoui, Alexandre Maurer, and Vladislav Tempez investigate the fundamental question whether gathering is still possible in models where visibility is severely restricted. The authors also initiate to study the question whether gathering behavior can be learned without explicit communication in a partially observable environment.
- Michel Raynal welcomes you to a guided tour on consensus numbers. In addition to more ancient results, he also surveys recent contributions related to the existence of an infinity of objects (of increasing synchronization/agreement power) at each level of the consensus hierarchy.

I hope you enjoy your journeys and on this occasion, I would like to thank the authors very much for their contributions to the EATCS Bulletin.

EXPLORING THE BORDERLANDS OF THE GATHERING PROBLEM

El Mahdi El Mhamdi
EPFL

elmahdi.elmhamdi@epfl.ch

Rachid Guerraoui
EPFL

rachid.guerraoui@epfl.ch

Alexandre Maurer
EPFL

alexandre.maurer@epfl.ch

Vladislav Tempez
LORIA

vladislav.tempez@ens-rennes.fr

Abstract

Problems of pattern formation have been extensively studied in distributed computing. One of these problems is the *gathering* problem: agents must gather at a same position in a distributed manner. When gathering is not possible, a close problem is the *convergence* problem.

In this article, we investigate the two following questions: (1) Can processes gather when each process cannot see more than one other process at the same time? (2) Can a gathering behavior be *learned* by processes?

Regarding the first point, we introduce a new model with an extremely restricted visibility: each process can only see *one* other process (its closest neighbor). Our goal is to see if (and to what extent) the gathering and convergence problems can be solved in this setting. We first show that, surprisingly, the problem can be solved for a small number of processes (at most 5), but not beyond. This is due to indeterminacy in the case where there are several “closest neighbors” for a same process. By removing this indeterminacy with an additional hypothesis (choosing the closest neighbor according to an order on the positions of processes), we then show that the

problem can be solved for any number of processes. We also show that up to one crash failure can be tolerated for the convergence problem.

Regarding the second point, we present the first experimental evidence that a gathering behavior can be *learned* without explicit communication in a partially observable environment. The learned behavior has the same properties as a self-stabilizing distributed algorithm, as processes can gather from any initial state (and thus tolerate any transient failure). Besides, we show that it is possible to scale and then tolerate the brutal loss of up to 90% of agents without significant impact on the behavior.

1 Introduction

An interesting natural phenomenon is the ability of swarms of simple individuals to form complex and very regular patterns: swarms of fishes [78], birds [32], ants [37]. . . They do so in a totally distributed manner, without any centralized or irreplaceable leader. Such behaviors are a great source of inspiration for distributed computing.

Problems of *pattern formation* have been extensively studied by the distributed computing community [72, 74, 11, 2]. In order to prove mathematical results, the model is of course simplified: the individuals (called agents, robots or processes) are usually geometric points in a Euclidean space, operating in “look – compute – move” cycles. A famous example is the circle formation algorithm by Suzuki and Yamashita [74]. Another family of papers considers robots moving on a graph (eg. [34, 39, 54]).

In particular, a pattern formation problem which has been extensively studied is the *gathering* problem [4, 24, 26, 40, 57]: processes must gather at a same point in a finite time. When gathering is impossible, a close problem is the *convergence* problem [28, 7]: processes must get always closer to a same point.

This apparently simple problem can become surprisingly complex, depending on the model and hypotheses. We give a few examples below (the list, of course, is not exhaustive).

- **Asynchronous system.** A first idea is to relax the synchronicity hypothesis. In [66, 23, 25, 29] for instance, the cycles are executed asynchronously – e.g., the “look” operation of a robot can happen during the “move” operation of another robot. [53] studies the feasibility of asynchronous gathering on a ring topology, depending on the level of symmetry of the initial configuration. [41] showed that gathering was possible in the asynchronous model when robots have the same common orientation.
- **Fault tolerance.** Another idea is to make the system fault tolerant. The faults can be transient [6, 35] or permanent [5] – e.g., when a robot stops

moving forever. [5] and [33] show several impossibility results in the case of Byzantine failures – i.e., a robot exhibiting an arbitrary malicious behavior. [15] proves the necessary and sufficient conditions for convergence in a 1D space in the presence of Byzantine robots.

- **Limited visibility.** One can assume that robots only have a limited visibility range [41, 8]. The usual hypothesis is that the robots can only see other robots within a bounded radius. Another possible limit to visibility are *opaque* robots [13, 3]: if a robot C is between two robots A and B, A cannot see B. [14] considers a setting with both constraints simultaneously (opacity and bounded visibility radius).
- **Limited multiplicity detection.** When several robots are allowed to occupy the same position, the robots may (or may not) know the *multiplicity* of a given position, that is: the number of robots at this position. When total multiplicity detection is available, a gathering strategy is, for each robot, to move to the position with the highest multiplicity. A weaker multiplicity detection hypothesis is that robots can only know if there are “one” or “more than one” robots at a given position (global multiplicity detection) [52, 53]. In [49, 50], this capacity is restricted to the current position (local multiplicity detection). [31] studies gathering on a grid without multiplicity detection.
- **Fat robots.** It is often assumed that robots are geometrical points, without a volume. Some papers consider the model of “fat robots”, where robots actually do have a volume. [3] considers the problem of gathering 4 robots modeled as discs. [30] generalizes this result to n robots. [14] considers the problem of gathering fat robots with a limited visibility.

In this article, we explore two new settings for the gathering problem. Basically, we ask ourselves the two following questions:

1. Can processes gather when each process cannot see more than *one* other process at the same time? (In the following, we call this setting “extremely restricted visibility”.)
2. Can a gathering behavior be *learned* by processes?

Gathering with extremely restricted visibility. Consider the following assumption: each process can only see its closest neighbor (i.e., the closest other process), and ignores the total number of processes. To our knowledge, no paper has yet considered such a minimalist setting. We study to what extent the gathering and

convergence problems can be solved in this setting. We assume a synchronous scheduler and memoryless processes that cannot communicate with messages.

There is an indeterminacy in the case where there are several “closest neighbors” (i.e., two or more processes at the same distance of a given process). We first assume that, in this situation, the closest neighbor is arbitrarily chosen by an external adversary (worst-case scenario).

In this scenario, we show that, surprisingly, the problems can only be solved for a small number of processes. More precisely, if n is the number of processes and d is the number of dimensions of the Euclidean space, then the gathering (resp. convergence) problem can be solved if and only if $d = 1$ or $n \leq 2$ (resp. $d = 1$ or $n \leq 5$). Indeed, for larger values of n , there exists initial configurations from which gathering or convergence is impossible, due to symmetry. The proof is constructive: for the small values of n , we provide an algorithm solving the problems. The proof is non-trivial for $n = 4$ and $n = 5$, as several families of cases need to be considered.

Therefore, to solve the problems for larger values of n , one additional hypothesis must necessarily be added. We remove the aforementioned indeterminacy by making the choice of the closest neighbor (when there is more than one) deterministic instead of arbitrary (according to an order on the positions of processes). Then, we show that the gathering problem is always solved in at most $n - 1$ steps by a simple “Move to the Middle” (MM) algorithm.

We finally consider the case of crash failures, where at most f processes lose the ability to move. We show that the gathering (resp. convergence) problem can only be solved when $f = 0$ (resp. $f \leq 1$). When the convergence problem can be solved, the MM algorithm solves it.

The technical details are presented in Section 2. Beyond this first work, we believe that this minimalist model can be the ground for many other interesting results.

Learning to gather. In previous works, the gathering behavior was obtained by giving an explicit algorithm to each (correct) agent. An alternative approach is *machine learning* [71], that is: automatically extracting a model from a dataset, or from its interactions with the environment. More particularly, *Reinforcement learning* [77, 73] is the specific machine learning paradigm that enables to obtain a desired behavior with the simplest feedback from the environment. It is particularly useful in network related problems [67, 12, 47]. In short, reinforcement learning consists, for the program, in receiving *rewards* and *penalties* from the environment, and learning which behavior leads to rewards and which behavior leads to penalties. To our knowledge (see the state of the art in Section 3), the question whether the agents can learn to gather with only simple rewards and

penalties from the environment (and with no other form of communication than “seeing each other”) remains open.

We present the first experimental evidence that the answer to this question is affirmative: agents can indeed learn a gathering behavior. We show that agents can learn to gather on a one-dimensional ring. The agents are rewarded for being in a group and penalized for being isolated.

A technical difficulty lies in the “combinatorial explosion” of the number of states. To overcome this difficulty, the agents approximate the environment by grouping close positions into clusters: each agent only perceives an *approximation* of the distribution of other agents in each cluster. This enables to keep the learning space constant (i.e., independent of the number of agents and the size of the ring). We show that, surprisingly, the agents manage to gather almost perfectly despite this very rough approximation.

We then consider the problem of increasing the number of agents. A natural belief would be that the agents have to “re-learn” to gather in this case. Interestingly, we show that the learned behavior can directly apply to a much larger number of agents – namely, if agents have learned to gather in groups of 10, we show that they immediately know how to gather in groups of up to 100. Aside from saving learning time, the interest of this approach is that such a group of 100 agents is inherently and deeply *robust* (fault-tolerant), because it can tolerate the loss of up to 90 agents¹. We also compare the learned behavior with a hardcoded algorithm that moves towards the barycenter of the agents. We thus show that, even with a relatively simple learning scheme, we can reach the same performances as this hardcoded behavior.

The technical details are presented in Section 3.

2 Gathering with extremely restricted visibility

In Section 2.1, we define the model and the problems. In Section 2.2, we characterize the class of algorithms allowed by our model, and define a simple algorithm to prove the positive results. In Section 2.3, we prove the aforementioned lower bounds. In Section 2.4, we remove indeterminacy and show that the gathering problem can be solved for any n . In Section 2.5, we consider the case of crash failures.

¹We do not claim that training a group of 100 agents makes it robust, but that we can easily build a robust group of 100 agents after training a group of 10 agents (which, by the way, is less costly).

2.1 Model and problems

Model. We consider a Euclidean space S of dimension d ($d \geq 1$). The position of each point of S is described by d coordinates (x_1, x_2, \dots, x_d) in a Cartesian system. For two points A and B of coordinates (a_1, \dots, a_d) and (b_1, \dots, b_d) , let $d(A, B) = \sqrt{\sum_{i=1}^d (a_i - b_i)^2}$ be the distance between A and B .

Let P be a set of n processes. $\forall p \in P$, let M_p be the position of p in S . Let Ω be the set of positions occupied by the processes of P . As several processes can share the same position, $1 \leq |\Omega| \leq |P|$. The time is divided in discrete steps $t \in \{0, 1, 2, 3, \dots\}$.

If $|\Omega| = 1$, the processes are *gathered* (they all have the same position). If $|\Omega| \geq 2$, $\forall p \in P$, let $D(p) = \min_{K \in \Omega - \{M_p\}} d(M_p, K)$, and let $N(p)$ be the set of processes q such that $d(M_p, M_q) = D(p)$. At a given time t , the *closest neighbor* of a process p is a process of N_p arbitrarily chosen by an external adversary. We denote it by $C(p)$.

We consider a synchronous execution model. At a given time t , a process p can only see M_p and $M_{C(p)}$ (without global orientation), and use these two points to compute a new position K . Then, the position of p at time $t + 1$ is K .

The processes are oblivious (they have no memory), mute (they cannot communicate) and anonymous (they cannot distinguish each other with identifiers). Note that this model does not assume multiplicity detection (the ability to count the processes at a same position). The processes do not know n . At $t = 0$, the n processes can have any arbitrary positions.

Problems. For a given point $G \in S$ and a given constant ϵ , we say that the processes are (G, ϵ) -gathered if, $\forall M \in \Omega$, $d(G, M) \leq \epsilon$.

An algorithm solves the *convergence* problem if, for any initial configuration, there exists a point $G \in S$ such that, $\forall \epsilon > 0$, there exists a time T such that the processes are (G, ϵ) -gathered $\forall t \geq T$.

An algorithm solves the *gathering* problem if, for any initial configuration, there exists a point G and a time T such that the processes are $(G, 0)$ -gathered $\forall t \geq T$.

2.2 Algorithm

In this section, we describe all possible algorithms that our model allows. Doing so enables us to show lower bounds further – that is, showing that *no algorithm* can solve some problems in our model. This is not to confuse with the *MM* algorithm (a particular case, defined below), which is only used to prove positive results.

Here, an algorithm consists in determining, for any process p , the position of p at the next step, as a function of M_p and $M_{C(p)}$.

First, let us notice that, if the processes are gathered ($|\Omega| = 1$), the processes have no interest in moving anymore. This corresponds to the case where each process cannot see any “closest neighbor”. Thus, we assume that any algorithm is such that, when a process p cannot see any closest neighbor, p does not move.

Now, consider the case where the processes are not gathered ($|\Omega| \geq 2$). Let p be the current process, let $D = D(p)$, and let \vec{x} be the unit vector ($\|\vec{x}\| = 1$) directed from M_p to $M_{C(p)}$. There are 2 possible cases.

Case 1: $d = 1$. The next position of p is $M_p + f_x(D)\vec{x}$, where f_x is an arbitrary function.

Case 2: $d \geq 2$. Let Δ be the axis defined by M_p and $M_{C(p)}$. If $d \geq 2$, as there is no global orientation of processes (M_p can only position itself relatively to $M_{C(p)}$), the next position of p can only be determined by (1) its position on axis Δ and (2) its distance to Δ . The difference here is that, for two given parameters (1) and (2), there are several possible positions (2 positions for $d = 2$, an infinity of positions for $d \geq 3$). Thus, we assume that the next position (among these possible positions) is arbitrarily chosen by an external adversary.

More formally, the next position of p is $M_p + f_x(D)\vec{x} + f_y(D)\vec{y}$, where f_x and f_y are arbitrary functions, and where \vec{y} is a vector orthogonal to \vec{x} which is arbitrarily chosen by an external adversary.

Move to the Middle (MM) algorithm. We finally define one particular algorithm to show some upper bounds. The Move to the Middle (MM) algorithm consists, for each process p and at each step, in moving to the middle of the segment defined by M_p and $M_{C(p)}$.

More formally, if $d = 1$, the MM algorithm is defined by $f_x(D) = D/2$. If $d \geq 2$, the MM algorithm is defined by $f_x(D) = D/2$ and $f_y(D) = 0$.

2.3 Lower bounds

In this section, we show the two following results.

- The gathering problem can be solved if and only if $d = 1$ or $n \leq 2$. When it can be solved, the MM algorithm solves it (Theorem [1](#)).
- The convergence problem can be solved if and only if $d = 1$ or $n \leq 5$. When it can be solved, the MM algorithm solves it (Theorem [2](#)).

2.3.1 Gathering problem

Let us prove Theorem [1](#).

Lemma 1. *If $d = 1$, the MM algorithm solves the gathering problem.*

Proof. Let us show that, if $|\Omega| \geq 2$, then $|\Omega|$ decreases at the next step.

As $d = 1$, let $x(K)$ be the coordinate of point K . Let (K_1, K_2, \dots, K_m) be the points of Ω ranked such that $x(K_1) < x(K_2) < \dots < x(K_m)$. $\forall i \in \{1, \dots, m\}$, let $x_i = x(K_i)$. Then, according to the MM algorithm, the possible positions at the next step are: $(x_1 + x_2)/2, (x_2 + x_3)/2, \dots, (x_{m-1} + x_m)/2$ (at most $m - 1$ positions). Thus, $|\Omega|$ decreases at the next step. Therefore, after at most $n - 1$ steps, we have $|\Omega| = 1$, and the gathering problem is solved. \square

Lemma 2. *If $d \geq 2$ and $n \geq 3$, the gathering problem is impossible to solve.*

Proof. First, consider the case $d = 2$. Consider an initial configuration where Ω contains three distinct points K_1, K_2 and K_3 such that $d(K_1, K_2) = d(K_2, K_3) = d(K_3, K_1) = D$.

Let G be the gravity center of the triangle $K_1K_2K_3$. Let $s(1) = 2, s(2) = 3$ and $s(3) = 1$. $\forall i \in \{1, 2, 3\}$, let A_i and B_i be the two half-planes delimited by the axis $(K_iK_{s(i)})$, such that G belongs to B_i . Let \vec{v}_i be the unit vector orthogonal to $(K_iK_{s(i)})$ such that the point $K_i + \vec{v}_i$ belongs to A_i . Let $\vec{y}_i = \vec{v}_i$ if $f_y(D) \geq 0$, and $\vec{y}_i = -\vec{v}_i$ otherwise.

Let p be a process, and let i be such that $M_p = K_i$. The external adversary can choose a closest neighbor $C(p)$ and a vector \vec{y} such that $M_{C(p)} = K_{s(i)}$ and $\vec{y} = \vec{y}_i$.

Thus, at the next step, it is always possible that Ω contains three *distinct* points also forming an equilateral triangle. The choice of vectors \vec{y} prevents the particular case where all processes are gathered in point G . We can repeat this reasoning endlessly. Thus, the gathering problem cannot be solved if $d = 2$.

Now, consider the case $d > 2$. The external adversary can choose the \vec{y} vectors such that the points of Ω always remain in the same plane, and their behavior is the same as for $d = 2$. Thus, the gathering problem cannot be solved if $d > 2$. \square

Theorem 1. *The gathering problem can be solved if and only if $d = 1$ or $n \leq 2$. When it can be solved, the MM algorithm solves it.*

Proof. If $d = 1$, according to Lemma [1](#), the MM algorithm solves the gathering problem. If $n = 1$, the gathering problem is already solved by definition. If $n = 2$, the MM algorithm solves the gathering problem in at most one step. Otherwise, if $d \geq 2$ and $n \geq 3$, according to Lemma [2](#), the gathering problem cannot be solved. \square

2.3.2 Convergence problem

Let us prove Theorem [2](#).

We first introduce some definitions. For a given set of points $X \subseteq S$, let $D_{\max}(X) = \max_{\{A,B\} \subseteq X} d(A,B)$. Let $\Omega(t)$ be the set Ω at time t . Let $d_{\max}(t) = \max_{\{A,B\} \subseteq \Omega(t)} d(A,B)$ and $d_{\min}(t) = \min_{\{A,B\} \subseteq \Omega(t)} d(A,B)$. Let $m(A,B)$ be the middle of segment $[AB]$. Let $\alpha(K) = \sqrt{1 - 1/(4K^2)}$.

Let $R(t) = \arg \min_{G \in S} \max_{M \in \Omega(t)} d(G, M)$ (the radius of the smallest enclosing ball of all processes' positions). Let $X_i(t)$ be the smallest i^{th} coordinate of a point of $\Omega(t)$. We say that a proposition $P(t)$ is true *infinitely often* if, for any time t , there exists a time $t' \geq t$ such that $P(t')$ is true.

Lemma 3. *If there exists a time t such that $|\Omega(t)| \leq 3$, the MM algorithm solves the convergence problem.*

Proof. If $|\Omega(t)| = 1$, the processes are and remain gathered. If $|\Omega(t)| = 2$, then $|\Omega(t+1)| = 1$.

If $|\Omega(t)| = 3$, consider the following proposition P : there exists $t' > t$ such that $|\Omega(t')| \leq 2$. If P is true, the gathering (and thus, convergence) problem is solved. Now, consider the case where P is false.

Let $\Omega(t) = \{A, B, C\}$. As $|\Omega(t+1)| = 3$, $\Omega(t+1) = \{m(A, B), m(B, C), m(C, A)\}$. The center of gravity G of the triangle formed by the three points of Ω always remains the same, and $d_{\max}(t)$ is divided by two at each step. Thus, $\forall \epsilon > 0$, there exists a time T such that the processes are (G, ϵ) -gathered $\forall t \geq T$. \square

Lemma 4. *Let $K \geq 1$. If $R(t) \leq Kd_{\min}(t)$, then $R(t+1) \leq \alpha(K)R(t)$.*

Proof. If the processes move according to the MM algorithm, then $\Omega(t+1) \subseteq \bigcup_{\{A,B\} \subseteq \Omega(t)} \{m(A, B)\}$. Let G be such that, $\forall M \in \Omega(t)$, $d(G, M) \leq R(t)$. Let A and B be two points of S such that $d(G, A) = d(G, B) = R(t)$ and $d(A, B) = d_{\min}(t)$ (two such points A and B exist, as $d_{\min}(t) \leq 2R(t)$). Let $C = m(A, B)$. Then, $\forall M \in \Omega(t+1)$, $d(G, M) \leq d(G, C)$. Thus, $R(t+1) \leq d(G, C)$.

Let $x = d(G, C)$, $y = d_{\min}(t)/2$ and $z = R(t)$. Then, $z^2 = x^2 + y^2$ and $x/z = \sqrt{1 - (y/z)^2}$. As $R(t) \leq Kd_{\min}(t)$, $y/z \geq 1/(2K)$ and $x/z \leq \sqrt{1 - 1/(4K^2)} = \alpha(K)$. Thus, $R(t+1) \leq d(G, C) \leq \alpha(K)R(t)$. \square

Lemma 5. *Let A, B, C, D and E be five points (some of them may be identical). Let $x = d(A, D)/100$. Assume $d(A, B) \leq x$, $d(A, C) \leq x$, $d(A, E) \leq 100x$ and $d(D, E) \geq 40x$. Let $S = \{A, B, C, D, E\}$ and $S' = \bigcup_{\{A,B\} \subseteq S} \{m(A, B)\}$. Then, $D_{\max}(S') \leq 0.99D_{\max}(S)$.*

Proof. As $d(A, D) = 100x$, $D_{\max}(S) \geq 100x$.

Let $M_1 = m(A, D)$, $M_2 = m(A, E)$ and $M_3 = m(D, E)$. We have $d(A, M_1) = 50x$ and $d(A, M_2) \leq 50x$. The maximal value of $y = d(A, M_3)$ is reached when

$d(A, D) = d(A, E) = 100x$ and $d(D, E) = 40x$. In this case, with the Pythagorean theorem, we have $(100x)^2 = y^2 + (20x)^2$, and thus $y \leq 98x$.

Thus, $\max_{i \in \{1,2,3\}} d(A, M_i) \leq 98x$. Now, suppose that $D_{\max}(S') > 99x$. Let $M_4 = m(A, B)$ and $M_5 = m(A, C)$. This would imply that there exists $i \in \{1, 2, 3\}$ such that either $d(M_i, M_4) > 99x$ or $d(M_i, M_5) > 99x$, and thus, that either $d(A, B) > x$ or $d(A, C) > x$, which is not the case. Thus, $D_{\max}(S') \leq 99x \leq 0.99D_{\max}(S)$. \square

Lemma 6. *Let t be a given time. If $n = 5$ and $|\Omega(t)| = 5$, then one of the following propositions is true:*

- (1) $|\Omega(t+1)| \leq 4$
- (2) $R(t+1) \leq \alpha(1000)R(t)$
- (3) $d_{\max}(t+1) \leq 0.99d_{\max}(t)$

Proof. Suppose that (1) and (2) are false. According to Lemma 4, (2) being false implies that $R(t) > 1000d_{\min}(t)$. Let A_0 and B_0 be two points of $\Omega(t)$ such that $d(A_0, B_0) = d_{\min}(t)$. As $|\Omega(t+1)| = 5$, it implies that the processes at A_0 and B_0 did not both move to $m(A_0, B_0)$. Therefore, there is a point C of $\Omega(t)$ such that $d(A_0, C) = d_{\min}(t)$ or $d(B_0, C) = d_{\min}(t)$. If $d(A_0, C) = d_{\min}(t)$, let $A = A_0$ and $B = B_0$. Otherwise, let $A = B_0$ and $B = A_0$.

As $R(t) > 1000d_{\min}(t)$, there exists a point D_0 of $\Omega(t)$ such that $d(A, D_0) \geq 100d_{\min}(t)$. Let E_0 be the fifth point of $\Omega(t)$. If $d(A, D_0) \geq d(A_0, E_0)$, let $D = D_0$ and $E = E_0$. Otherwise, let $D = E_0$ and $E = D_0$.

Finally, let $x = d(A, D)/100$. Thus, we have $d(A, B) \leq x$, $d(A, C) \leq x$ and $d(A, E) \leq 100x$. If $d(D, E) < 40x$, then the processes at positions D and E both move to $m(D, E)$, and $|\Omega(t+1)| = 4$: contradiction. Thus, $d(D, E) \geq 40x$. Let $S = \Omega(t)$, and let $S' = \bigcup_{\{A,B\} \subseteq S} \{m(A, B)\}$. Then, according to Lemma 5, $D_{\max}(S') \leq 0.99D_{\max}(S)$.

As the processes move according to the MM algorithm, $\Omega(t+1) \subseteq S'$, and $d_{\max}(t+1) \leq D_{\max}(S') \leq 0.99D_{\max}(S) = 0.99d_{\max}(t)$. Thus, (3) is true.

Therefore, either (1) or (2) are true, or (3) is true. \square

Lemma 7. *Let t be a given time. If $|\Omega(t)| = 4$, then one of the following propositions is true:*

- (1) $|\Omega(t+1)| \leq 3$
- (2) $R(t+1) \leq \alpha(1000)R(t)$
- (3) $d_{\max}(t+1) \leq 0.99d_{\max}(t)$

Proof. Suppose that (1) and (2) are false. According to Lemma 4, (2) being false implies that $R(t) > 1000d_{\min}(t)$. Let A and B be two points of $\Omega(t)$ such that $d(A, B) = d_{\min}(t)$.

As $R(t) > 1000d_{\min}(t)$, there exists a point D_0 of $\Omega(t)$ such that $d(A, D_0) \geq 100d_{\min}(t)$. Let E_0 be the fourth point of $\Omega(t)$. If $d(A, D_0) \geq d(A_0, E_0)$, let $D = D_0$ and $E = E_0$. Otherwise, let $D = E_0$ and $E = D_0$.

Let $C = A$ and $x = d(A, D)/100$. Thus, we have $d(A, B) \leq x$, $d(A, C) \leq x$ and $d(A, E) \leq 100x$. If $d(D, E) < 40x$, then the processes at D and E (resp. A and B) both move to $m(D, E)$ (resp. $m(A, B)$), and $|\Omega(t+1)| = 2$: contradiction. Thus, $d(D, E) \geq 40x$.

Let $S = \Omega(t)$, and let $S' = \bigcup_{\{A, B\} \subseteq S} \{m(A, B)\}$. Then, according to Lemma 5, $D_{\max}(S') \leq 0.99D_{\max}(S)$.

As the processes move according to the MM algorithm, $|\Omega(t+1)| \subseteq S'$, and $d_{\max}(t+1) \leq D_{\max}(S') \leq 0.99D_{\max}(S) = 0.99d_{\max}(t)$. Thus, (3) is true.

Therefore, either (1) or (2) are true, or (3) is true. \square

Lemma 8. *At any time t , $R(t+1) \leq R(t)$.*

Proof. Suppose the opposite: $R(t+1) > R(t)$. Let G be a point such that, $\forall M \in \Omega(t)$, $d(G, M) \leq R(t)$. If, $\forall M \in \Omega(t+1)$, $d(G, M) \leq R(t)$, then we do not have $R(t+1) > R(t)$. Thus, there exists a point A of $\Omega(t+1)$ such that $d(G, A) > R(t)$. Let B be the previous position of processes at position A . As the processes at position B moved to A , according to the MM algorithm, there exists a point C of $\Omega(t)$ such that $A = m(B, C)$. As $d(G, B) \leq R(t)$ and $d(G, A) > R(t)$, we have $d(G, C) > R(t)$. Thus, there exists a point C of $\Omega(t)$ such that $d(G, C) > R(t)$: contradiction. \square

Lemma 9. *At any time t , $d_{\max}(t+1) \leq d_{\max}(t)$.*

Proof. Suppose the opposite: $d_{\max}(t+1) > d_{\max}(t)$. Let A and B be two points of $\Omega(t+1)$ such that $d(A, B) = d_{\max}(t+1)$. According to the MM algorithm, there exists four points A_1, A_2, B_1 and B_2 of $\Omega(t)$ such that $A = m(A_1, A_2)$ and $B = m(B_1, B_2)$.

Let L be the line containing A and B . Let A'_1 (resp. A'_2, B'_1 and B'_2) be the projection of A_1 (resp. A_2, B_1 and B_2) on L . Then, there exists $i \in \{1, 2\}$ and $j \in \{1, 2\}$ such that $d(A'_i, B'_j) \geq d(A, B)$. Thus, $d(A_i, B_j) \geq d(A, B) = d_{\max}(t)$: contradiction. \square

Lemma 10. *Let $n \leq 5$. Let $P_1(t)$ (resp. $P_2(t)$) be the following proposition: $R(t+1) \leq \alpha(1000)R(t)$ (resp. $d_{\max}(t+1) \leq 0.99d_{\max}(t)$). Let $P(t) = P_1(t) \vee P_2(t)$. If, for any time t , $|\Omega(t)| \geq 4$, then $P(t)$ is true infinitely often.*

Proof. Let P^* be the following proposition: “ $|\Omega(t)| = 4$ ” is true infinitely often.

If P^* is false, there exists a time t' such that $\forall t \geq t'$, $|\Omega(t)| = 5$. Thus, the result follows, according to Lemma 6. If P^* is true, there exists an infinite set $T = \{t_1, t_2, t_3, \dots\}$ such that $\forall t \in T$, $|\Omega(t)| = 4$. Then, according to Lemma 7, $P(t+1)$ is true $\forall t \in T$. Thus, the result follows. \square

Lemma 11. *Let $n \leq 5$. Suppose that, for any time t , $|\Omega(t)| \geq 4$. Then, for any time t , there exists a time $t' > t$ such that $R(t') \leq \alpha(1000)R(t)$.*

Proof. Suppose the opposite: there exists a time t_0 such that, $\forall t > t_0$, $R(t) > \alpha(1000)R(t_0)$.

Consider the propositions $P_1(t)$ and $P_2(t)$ of Lemma 10. Then, $\forall t \geq t_0$, $P_1(t)$ is false. Thus, according to Lemma 10, it implies that $P_2(t)$ is true infinitely often.

Let $t' > t_0$ be such that, between time t_0 and time t' , $P_2(t)$ is true at least 200 times. According to Lemma 9, for any time t , we have $d_{\max}(t+1) \leq d_{\max}(t)$. Thus, $d_{\max}(t') \leq 0.99^{200}d_{\max}(t_0) \leq d_{\max}(t_0)/4$. For any time t , $d_{\max}(t) \geq R(t)$ and $d_{\max}(t) \leq 2R(t)$. Thus, $R(t') \leq R(t_0)/2 \leq \alpha(1000)R(t_0)$: contradiction. Thus, the result follows. \square

Lemma 12. *Let G be a point such that, $\forall M \in \Omega(t)$, $d(G, M) \leq R(t)$. Then, $\forall M \in \Omega(t+1)$, $d(G, M) \leq R(t)$.*

Proof. Suppose the opposite: there exists a point K of $\Omega(t+1)$ such that $d(G, K) > R(t)$. According to the MM algorithm, there exists two points A and B of $\Omega(t)$ such that $K = m(A, B)$. Then, as $d(G, K) > R(t)$, either $d(G, A) > R(t)$ or $d(G, B) > R(t)$: contradiction. Thus, the result follows. \square

Lemma 13. *$\forall i \in \{1, \dots, d\}$ and for any two instants t and $t' > t$, $|X_i(t') - X_i(t)| \leq 2R(t)$.*

Proof. For any point M , let $x_i(M)$ be the i^{th} coordinate of M . Let G be a point such as described in Lemma 12. According to Lemma 12, $\forall M \in \Omega(t+1)$, $|x_i(M) - x_i(G)| \leq R(t)$. By induction, $\forall t' > t$ and $\forall M \in \Omega(t')$, $|x_i(M) - x_i(G)| \leq R(t)$. In particular, $|X_i(t) - x_i(G)| \leq R(t)$ and $|X_i(t') - x_i(G)| \leq R(t)$. Thus, $|X_i(t') - X_i(t)| \leq 2R(t)$. \square

Lemma 14. *Let $(u_k)_k$ be a sequence, Let $\alpha \in]0, 1[$ and let N be an integer. If $\forall k \geq N$, $|u_{k+1} - u_k| \leq \alpha^k$, then $(u_k)_k$ converges.*

Proof. As $\alpha \in]0, 1[$, $S_\alpha = 1 + \alpha + \alpha^2 + \alpha^3 + \dots$ converges. Let $\epsilon > 0$. Let $K = \log(\epsilon/S_\alpha)/\log \alpha$. Then, $\alpha^K S_\alpha = \epsilon$.

Let $k \geq \max(K, N)$ and let $m > k$. $|u_m - u_k| \leq \sum_{i=k}^{m-1} |u_{i+1} - u_i| \leq \sum_{i=k}^{m-1} \alpha^i \leq \alpha^k S_\alpha \leq \alpha^K S_\alpha = \epsilon$

Thus, $(u_k)_k$ is a Cauchy sequence and it converges. \square

Lemma 15. *Let $\alpha \in]0, 1[$. If, for any time t , there exists a time $t' > t$ such that $R(t') \leq \alpha R(t)$, then the MM algorithm solves the convergence problem.*

Proof. Let t_0 be an arbitrary time. $\forall k \geq 0$, we define $t_{k+1} > t_k$ as the first time such that $R(t_{k+1}) \leq \alpha R(t_k)$. By induction, $\forall k \geq 0$, $R(t_k) \leq \alpha^k R(t_0)$.

Let $i \in \{1, \dots, d\}$. According to Lemma [13], $\forall k \geq 0$, we have $|X_i(t_{k+1}) - X_i(t_k)| \leq 2R(t_k) \leq 2\alpha^k R(t_0)$. $\forall k \geq 0$, let $u_k = X_i(t_k)/(2R(t_0))$. Then, $\forall k \geq 0$, $|u_{k+1} - u_k| \leq \alpha^k$.

According to Lemma [14], the sequence $(u_k)_k$ converges and so does $(X_i(t_k))_k$. Let L_i be the limit of $(X_i(t_k))_k$, and let G be the point of coordinates (L_1, L_2, \dots, L_d) .

$R(t_k)$ decreases exponentially with k . Then, $\forall \epsilon > 0$, there exists an integer k such that $R(t_k) < \epsilon/2$. According to Lemma [8], $\forall t > t_k$, $R(t) \geq R(t_k)$. Therefore, the processes are (G, ϵ) -gathered $\forall t \geq t_k$, and the convergence problem is solved. \square

Lemma 16. *If $d = 1$ or $n \leq 5$, the MM algorithm solves the convergence problem.*

Proof. If $d = 1$, according to Lemma [1], the MM algorithm solves the gathering problem, and thus the convergence problem. Now, suppose that $n \leq 5$.

Suppose that, for any time t , $|\Omega(t)| \geq 4$. Then, according to Lemma [11] and Lemma [15], the MM algorithm solves the convergence problem. Otherwise, i.e., if $|\Omega(t)| \leq 3$, then according to Lemma [3], the MM algorithm solves the convergence problem. \square

Lemma 17. *If $d \geq 2$ and $n \geq 6$, the convergence problem is impossible to solve.*

Proof. Assume the opposite: there exists an algorithm that always solves the convergence problem for $d \geq 2$ and $n \geq 6$.

First, assume that Ω contains 3 points, as described in the proof of Lemma [2]. Consider the infinite execution described in the proof of Lemma [2]. Let G be the barycenter of these 3 points.

Let P be the following proposition: there exists a constant D such that the distance between G and any of the 3 points of Ω is at most D .

If P is false, then by definition, the convergence problem cannot be solved. We now consider the case where P is true.

If P is true, then consider the following case: Ω contains 6 points K_1, K_2, K_3, K_4, K_5 and K_6 . K_1, K_2 and K_3 are arranged such as described in the proof of Lemma [2], and so are K_4, K_5 and K_6 . Let G (resp G') be the barycenter of the triangle formed by K_1, K_2 and K_3 (resp. K_4, K_5 and K_6). Assume that $d(G, G') = 10D$.

Now, assume that the points of the two triangles respectively follow the infinite execution described in the proof of Lemma [2]. Then, the distance between any two of the 6 points is always at least $8D$, and the convergence problem cannot be solved. \square

Theorem 2. *The convergence problem can be solved if and only if $d = 1$ or $n \leq 5$. When it can be solved, the MM algorithm solves it.*

Proof. The result follows from Lemma [16] and Lemma [17]. \square

2.4 Breaking symmetry

We showed that the problems were impossible to solve for $n \geq 6$. This is due to particular configurations where a process p has several “closest neighbors” (i.e., $|N_p| > 1$). Until now, we assumed that the actual closest neighbor $C(p)$ of p was chosen in N_p by an external adversary.

We now assume that, whenever $|N_p| > 1$, $C(p)$ is chosen deterministically, according to an *order* on the positions of processes. Namely, we assume that there exists an order “ $<$ ” such that any set of distinct points can be ordered from “smallest” to “largest” ($A_1 < A_2 < A_3 < \dots < A_k$).

Let $L(p)$ be the largest element of N_p , that is: $\forall q \in N_p - \{L(p)\}, M_q < M_{L(p)}$. We now assume that, for any process p , $C(p) = L(p)$. With this new hypothesis, we show the following result: $\forall n \geq 2$, the MM algorithm solves the gathering problem in $n - 1$ steps, and no algorithm can solve the gathering problem in less than $n - 1$ steps (Theorem [3](#)).

Proof

Lemma 18. $\forall n \geq 2$, no algorithm can solve the gathering problem in less than $n - 1$ steps.

Proof. Suppose the opposite: there exists an algorithm X solving the gathering problem in less than $n - 1$ steps.

First, consider a case with two processes, initially at two distinct positions. Then, eventually, the two processes are gathered. Let t be the first time where the two processes are gathered. Let A and B be their position at time $t - 1$, and let $D = d(A, B)$. By symmetry, the two processes should move to $m(A, B)$ at time t . Thus, with algorithm X , whenever a process p is such that $d(M_p, M_{C(p)}) = D$, p moves to $m(M_p, M_{C(p)})$ at the next step.

Let $K(x)$ be the point of coordinates $(x, 0, 0, \dots, 0)$. Now consider n processes, a set $\Omega(0) = \bigcup_{i \in \{0, \dots, n-1\}} \{K(iD)\}$, and an order such that, $\forall x < y, K(x) < K(y)$.²

Let us prove the following property P_k by induction, $\forall k \in \{0, \dots, n - 1\}$:

$$\Omega(k) = \bigcup_{i \in \{0, \dots, n-k-1\}} \{K((i + k/2)D)\}.$$

- P_0 is true, as $\Omega(0) = \bigcup_{i \in \{0, \dots, n-1\}} \{K(iD)\}$.
- Suppose that P_k is true for $k \in \{0, \dots, n - 2\}$. Then, according to algorithm X , the processes at position $K((n-k-1 + k/2)D)$ moves to $K((n-k-1 + (k-1)/2)D)$, and $\forall i \in \{0, \dots, n-k-2\}$, the processes at position $K((i + k/2)D)$ move to $K((i + (k+1)/2)D)$. Thus, P_{k+1} is true.

²As this is a lower bound proof, our goal here is to exhibit one particular situation where no algorithm can solve the problem in less than $n - 1$ steps. Thus, we choose a worst-case configuration with a worst-case order.

Therefore, $\forall t \in \{0, \dots, n-2\}$, $|\Omega(t)| \geq 2$, and the processes are not gathered: contradiction. Thus, the result follows. \square

We now assume that the processes move according to the MM algorithm.

Lemma 19. *Let p and q be two processes. If there exists a time t where $M_p = M_q$, then at any time $t' > t$, $M_p = M_q$.*

Proof. Consider the configuration at time t . According to our new hypothesis, $C(p) = C(q)$. Let $K = m(M_p, M_{C(p)}) = m(M_q, M_{C(q)})$. According to the MM algorithm, p and q both move to K . Thus, at time $t+1$, we still have $M_p = M_q$. Thus, by induction, the result. \square

Lemma 20. *At any time t , if the processes are not gathered, there exists two processes p and q such that $M_p \neq M_q$, $p = C(q)$ and $q = C(p)$.*

Proof. Let $\delta = \min_{\{A,B\} \subseteq \Omega(t)} d(A,B)$. Let Z be the set of processes p such that $d(M_p, M_{C(p)}) = \delta$. Let $Z' = \bigcup_{p \in Z} \{p, C(p)\}$.

Let A be the point of Z' such that, $\forall M \in Z' - \{A\}$, $M < A$. Let p be a process at position A .

Let q be the largest element of N_p , that is: $\forall q' \in N_p - \{q\}$, $M_{q'} < M_q$. By definition, $M_p \neq M_q$. Thus, according to our new hypothesis, $q = C(p)$.

Then, note that p is also the largest element of N_q : $\forall p' \in N_q - \{p\}$, $M_{p'} < M_p$. Thus, $p = C(q)$. Thus, the result follows. \square

Lemma 21. *At any time t , if the processes are not gathered, then $|\Omega(t+1)| \leq |\Omega(t)| - 1$.*

Proof. Let p and q be the processes described in Lemma 20. Let $K = m(M_p, M_q)$. Then, according to Lemma 20, the processes at position M_p and M_q both move to position K . Let $X = \Omega(t) - \{M_p, M_q\}$. According to Lemma 19, the processes occupying the positions of X cannot move to more than $|X|$ new positions. Thus, $|\Omega(t+1)|$ is at most $|\Omega(t)| - 1$. Thus, the result follows. \square

Lemma 22. $\forall n \geq 2$, the MM algorithm solves the gathering problem in at most $n - 1$ steps.

Proof. According to Lemma 21, there exists a time $t \leq n - 1$ such that $|\Omega(t)| = 1$. Let A be the only point of $\Omega(t)$. Then, according to the MM algorithm, the processes do not move from position A in the following steps. Thus, the result follows. \square

Theorem 3. $\forall n \geq 2$, the MM algorithm solves the gathering problem in $n - 1$ steps, and no algorithm can solve the gathering problem in less than $n - 1$ steps.

Proof. The result follows from Lemma 18 and Lemma 22. \square

2.5 Fault tolerance

We now consider the case of *crash failures*: some processes may lose the ability to move, without the others knowing it. Let $C \subseteq P$ be the set of crashed processes (the other processes are called “correct”), and let $S_c = \bigcup_{p \in C} \{M_p\}$ (i.e., the set of positions occupied by crashed processes). Let $f = |S_c|$.

We prove the two following results.

- The gathering problem can only be solved when $f = 0$ (Theorem 4).
- The convergence problem can be solved if and only if $f \leq 1$. When $f \leq 1$, the MM algorithm solves it (Theorem 5).

Proof

We say that a process p is *attracted* if there exists a sequence of processes (p_1, \dots, p_m) such that $p = p_1$, $p_m \in C$, and $\forall i \in \{1, \dots, m-1\}$, $C(p_i) = p_{i+1}$. A *loop* is a sequence of correct processes (p_1, \dots, p_m) such that $C(p_m) = p_1$ and, $\forall i \in \{1, \dots, m-1\}$, $C(p_i) = p_{i+1}$. A *pair* is a loop with 2 processes. Let $\Omega' = \bigcup_{p \in P-C} \{M_p\}$ (i.e., the set of positions occupied by *correct* processes). Let $\Omega'(t)$ be the state of Ω' at time t .

Lemma 23. *Consider an algorithm for which there exists w such that $f_x(w) = w$ and $f_y(w) = 0$. Then, this algorithm cannot solve the gathering nor the convergence problem.*

Proof. Assume the opposite. Consider a situation where $\Omega = \{A, B\}$, with $d(A, B) = w$. Then, according to the algorithm, the processes at position A and B switch their positions endlessly, and neither converge nor gather: contradiction. Thus, the result follows. \square

Theorem 4. *The gathering problem can only be solved when $f = 0$.*

Proof. If $f \geq 2$, by definition, the processes cannot be gathered. Now, suppose $f = 1$.

Suppose the opposite of the claim: there exists an algorithm solving the gathering problem when $f = 1$. Let P be the following proposition: there exists two points A and B such that all crashed processes are in position A , and all correct processes are in position B .

Consider an initial configuration where P is true. As the algorithm solves the gathering problem, according to Lemma 23, the next position of correct processes cannot be A . Thus, P is still true at the next time step, with a different point B .

Therefore, by induction, P is always true, and the processes are never gathered: contradiction. Thus, the result follows. \square

Lemma 24. *If there exists a process p which is not attracted, then there exists a loop.*

Proof. Suppose the opposite: there is no loop. Let $p_1 = p$. $\forall i \in \{1, \dots, n\}$, let $p_{i+1} = C(p_i)$. We prove the following property P_i by induction, $\forall i \in \{1, \dots, n+1\}$: (p_1, \dots, p_i) are i distinct processes.

- P_1 is true.
- Suppose that P_i is true for some $i \in \{1, \dots, n\}$. As there is no loop, we cannot have $p_{i+1} \in \{p_1, \dots, p_i\}$. Thus, P_{i+1} is true.

Thus, P_{n+1} is true, and there are $n+1$ distinct processes: contradiction. Thus, the result follows. \square

Lemma 25. *All loops are pairs.*

Proof. Let (p_1, \dots, p_m) be a loop. Let $\delta = \min_{i \in \{1, \dots, m\}} d(M_{p_i}, M_{C(p_i)})$. Let Z be the set of processes of $\{p_1, \dots, p_m\}$ such that $d(M_{p_i}, M_{C(p_i)}) = \delta$. Let $Z' = \bigcup_{p \in Z} \{p, C(p)\}$.

Let p be the process such that, $\forall q \in Z'$ such that $M_p \neq M_q$, $M_p > M_q$. Let $q = C(p)$. As $C(q)$ is the closest neighbor of p , $C(q) \in Z'$. Then, according to the definition of p , $C(q) = p$.

Therefore, (p_1, \dots, p_m) is either (p, q) or (q, p) . Thus, the result follows. \square

Lemma 26. *If there exists a pair, then $|\Omega'(t+1)| \leq |\Omega'(t) - 1|$.*

Proof. According to the algorithm, two processes at the same position at time t are at the same position at time $t+1$. Let (p, q) be a pair. Then, according to the algorithm, the processes at positions M_p and M_q move to $m(M_p, M_q)$, and $|\Omega'(t+1)| \leq |\Omega'(t) - 1|$. \square

Lemma 27. *There exists a time t_A such that, for any time $t \geq t_A$, all correct processes are attracted.*

Proof. Suppose the opposite. Then, after a finite number of time steps, at least one correct process is not attracted. Thus, according to Lemma 24, there exists a loop. According to Lemma 25, this loop is a pair. Then, according to Lemma 26, $|\Omega'|$ decreases.

We can repeat this reasoning $n+1$ times, and we then have $|\Omega'| < 0$: contradiction. Thus, the result follows. \square

Lemma 28. *Suppose $f = 1$. Let p be an attracted process, and let L be the distance between p and the crashed processes. Then, $d(M_p, M_{C(p)}) \geq L/n$.*

Proof. Suppose the opposite: $d(M_p, M_{C(p)}) < L/n$. As p is attracted, there exists a sequence of processes (p_1, \dots, p_m) such that $p = p_1$, $p_m \in C$, and $\forall i \in \{1, \dots, m-1\}$, $C(p_i) = p_{i+1}$.

$\forall i \in \{1, \dots, m-2\}$, we have $d(M_{p_i}, M_{p_{i+1}}) \geq d(M_{p_{i+1}}, M_{p_{i+2}})$. Indeed, suppose the opposite. Then, $C(p_{i+1}) = p_{i+2}$, $d(M_{p_{i+1}}, M_{C(p_{i+1})}) > d(M_{p_{i+1}}, M_{p_i})$, and $C(p_{i+1})$ is not a closest neighbor of p_{i+1} : contradiction. Thus, $d(M_{p_i}, M_{p_{i+1}}) \geq d(M_{p_{i+1}}, M_{p_{i+2}})$.

Thus, $\forall i \in \{1, \dots, m-1\}$, $d(p_i, p_{i+1}) < L/n$. Therefore, $d(p_1, p_m) \leq (m-1)L/n < L$: contradiction. Thus, the result follows. \square

Lemma 29. *Let $f = 1$, and let X be the position of crashed processes. Let $L = \max_{p \in P} d(X, M_p)$. Let $L(t)$ be the value of L at time t . Suppose that all correct processes are attracted. Then, for any time t , $L(t+1) \leq k(n)L(t)$, where $k(n) = \sqrt{1 - 1/(2n)^2}$.*

Proof. At time $t+1$, let p be a process such that $d(X, M_p) = L(t+1)$. Let K be the position of p at $t+1$. Then, according to the algorithm, at time t , there exists two processes q and r at position A and B such that $K = m(A, B)$.

Let $L' = \max(d(X, A), d(X, B))$. Let $q' \in \{q, r\}$ be such that $d(X, M_{q'}) = L'$. Then, according to Lemma 28, $d(M_{q'}, M_{C(q')}) \geq L'/n$. Let r' be the other process of $\{q, r\}$. Then, the position of r maximizing $d(X, K)$ is such that $d(X, M_{r'}) = L'$.

Therefore, according to the Pythagorean theorem, $(L(t+1))^2$ is at most $L'^2 - (L'/(2n))^2$, and $L(t+1) \leq k(n)L' \leq k(n)L(t)$. Thus, the result follows. \square

Lemma 30. *If $f = 1$, the MM algorithm solves the convergence problem.*

Proof. According to Lemma 27, there exists a time t_A after which all correct processes are attracted. We now suppose that $t \geq t_A$. Let $\epsilon > 0$. Let X be the position of crashed processes, and let $L = \max_{p \in P} d(X, M_p)$. As $k(n) = \sqrt{1 - 1/(2n)^2} < 1$, let M be such that $k(n)^M L < \epsilon$. Then, according to Lemma 29, at time $t_A + M$, all processes are at distance at most ϵ from X . Thus, the result follows. \square

Theorem 5. *The convergence problem can be solved if and only if $f \leq 1$. When $f \leq 1$, the MM algorithm solves it.*

Proof. When $f \geq 2$, there exists at least two crashed processes that will stay at the same position forever. Thus, the convergence problem cannot be solved.

When $f \leq 1$, according to Lemma 30, the MM algorithm solves the convergence problem. Thus, the result follows. \square

2.6 Future works

This first work can be the basis for many extensions. For instance, we could consider a more general scheduler (e.g. asynchronous). We could investigate how

resilient this model is to crash or Byzantine failures. We could also consider the case of voluminous processes, that cannot be reduced to one geometrical point.

3 Learning to gather

In Section 3.1, we give a state of the art of reinforcement learning in multi-agent systems w.r.t. the gathering problem. In Section 3.2, we present the Q-learning technique (with eligibility trace), then a precise formulation of the gathering problem in a Q-learning framework. In particular, we describe which state and actions are used to model the gathering problem in Q-learning. In Section 3.3, we explicit the numerical parameters used to implement our model. For pedagogical reasons, we first present results for a default setting; then, we show that the learned behaviors can be reused with more agents.

3.1 State of the art

Reinforcement learning [73, 46] consists in taking simple feedback from the environment to guide learning. The general idea is to associate rewards and penalties to past situations in order to learn how to act in future ones. The principle differs from that of supervised learning [42, 46] by the nature of the feedback. In supervised learning, an agent is taught how to perform precisely on several examples. In reinforcement learning, the agent only gets an appreciation feedback from the environment. For instance, in dog training, dogs are rewarded when doing correct actions and punished when behaving badly. The advantage here is the possibility to have a feedback in situations where the correct behavior is unknown. Several successful AI approaches use reinforcement learning, one spectacular example being the performance of AlphaGo [70] defeating the world Go champion Lee Sedol.

So far, reinforcement learning has mainly been used in situations with only one learning agent (*single-agent* systems), with important results [44, 38, 43, 48, 60, 68].

Multi-agent systems involve numerous independent agents interacting with each other. Many works on multi-agent reinforcement learning consider problems where only 2 or 3 agents are involved [10, 16, 27, 59, 65, 76, 80]. Some deal with competitive games (e.g. zero-sum games) [1], where agents are rewarded at the expense of others. Other tackle collaborative problems, but the reward is global and centralized [75]. The algorithm proposed in [21] achieves convergence, safety and targeted optimality against memory-bounded adversaries in arbitrary repeated games. [63] presents the first general framework for intelligent agents to learn to teach in a multiagent environment.

The domain of evolutionary robotics [36] studies how the behavior of agents can evolve through “natural selection” mechanisms, with [18] or without [56] communication. In this paper, we focus on behaviors that can be learned “within a lifetime”, through rewards and punishments.

In general, communication mechanisms are used to share information among agents [17, 51, 55, 62, 67, 69, 81] in order to increase the learning speed. Still, in some cases, communication between independent agents is difficult, impossible, or at least very costly [79, 9]. In these situations, it might be useful to devise a learning process that does not rely on communication.

Yet, so far, very few approaches considered a genuinely distributed setting where each agent is rewarded individually, and where agents do *not* communicate. In [61], the problem and the constraints are similar to our work, but the rewards are given for taking an *action* instead of reaching a *state*. Consequently, the final behavior is predetermined by the model itself. In [19], even if the constraints are similar (cooperative task, no communication and individual rewards), the problem tackled is fundamentally different: the task only requires the cooperation of agents by groups of two (not of all agents simultaneously).

3.2 Model

3.2.1 Q-learning

As recalled in the previous section, the goal of reinforcement learning is to make agents *learn* a behavior from reward-based feedback. In this paper, we work with a widely used reinforcement learning technique called *Q-learning* [77, 73, 80, 51, 20, 38]. More specifically, we use Q-learning with *eligibility trace* [58, 73] as explained in what follows.

Q-learning was initially devised for single agent problems. Here, we consider a multi-agent system where each agent has its own learning process. We describe in the following the learning model of *one* agent taken independently.

Let A be a set of *actions*, and let S be a set of *states* (representing all the situations in which the agent can be). The sets A and S contain a finite number of elements. In each state s , the agent may choose between different actions $a \in A$. Each action a leads to a state s' , in which the agent receives either a positive reward, a negative reward or no reward at all. The objective of Q-learning is to compute the cumulative expected reward for visiting a given state. Intuitively, this is materialized by the fact that *learning*, in Q-learning, is all about updating the Q-value using the mismatch between the previous Q-value and the observed reward.

Let $\pi : S \rightarrow A$ be the *policy function* of an agent – i.e., a function returning an action to take in each state.

Let X_t^{π, s_0} be the state in which the agent is after t steps, starting from state s_0 and following the policy π . In particular, $X_0^{\pi, s} = s$.

Let $r : S \rightarrow \mathbb{R}$ be the *reward function* associating a reward to each state.

The *cumulative expected reward* over a period $I = \llbracket 0, N \rrbracket$ of state s is

$$\sum_{t \in I} \mathbb{E}(\gamma^t r(X_t^{\pi, s}))$$

where $\gamma \in [0, 1]$ is a *discount parameter* modulating the importance of long term rewards. The long term rewards become more and more important when γ is close from 1.

When predicting the best transition from one state to another (by taking a given action) is difficult or impossible, it is useful to compute a cumulative expected reward of a couple (s, a) .

Under the assumption that each couple (s, a) is visited an infinite number of time, it is possible, following the law of large numbers, to estimate without bias the expected cumulative reward by sampling [77], i.e by trying state-action couples and building an estimator of the expected reward. We denote this estimator $Q(s, a)$, and call it the *Q-value* of the state-action couple (s, a) . The following formula is the usual update rule to compute an estimator of the Q-value.

$$Q_{t+1}(s, a) = (1 - \eta)Q_t(s, a) + \eta(r(X_t) + 1 + \gamma \max_{a'}(Q_t(s, a')))$$

if action a is taken in state s at step t .

$$Q_{t+1}(s', a') = Q_t(s', a')$$

otherwise.

Here, η is a parameter called the *learning rate* that modulates the importance of new rewards over old knowledge. Q_t is the estimate of the cumulative expected reward after t samples.

A complementary approach to get better estimations of Q-values with fewer samples is to use *eligibility trace* [58, 73]. The idea is to keep trace of older couples (s, a) until a reward is given, and to propagate a discounted reward to the couples (s, a) that led to the reward several steps later. Formally, for each state s , a value (eligibility) $e_t(s)$ is attributed. e is initialized at $e_0(s) = 0$ for every state s then updated as follows:

if $s_t = s$,

$$e_{t+1}(s) = \gamma\lambda * e_t(s) + 1$$

otherwise,

$$e_{t+1}(s) = \gamma\lambda * e_t(s).$$

Using the eligibility trace, Q-values are updated by the scaling of the update rule described above with eligibility values. The factor $\gamma\lambda$ used in the update of the eligibility acts as a discount in time: older visited states get less reward than recent visited states.

In addition to update rules for learning, we need a policy for choosing actions. An ϵ -greedy policy π is a stochastic policy such that: (1) with probability $(1 - \epsilon)$, $\pi(s) = a$ when (s, a) yields the highest expected cumulative reward from state s , and (2) with probability ϵ , a random action is chosen in A . The parameter ϵ is called the *exploration rate* and modulates the trade-off between *exploration* of new and unknown states (to obtain new information) and *exploitation* of current information (to sample valuable states more precisely and thus be rewarded).

3.2.2 Setting

We consider a *ring* topology. This is a simple topology for a bounded space that avoids non-realistic borders effects (i.e no need to "manually" replace an agent in the middle of the states-space if the agent reaches the border in the case of a square for example). There are n positions $\{0, \dots, n - 1\}$. $\forall k \in \{0, \dots, n - 2\}$, positions k and $k + 1$ are adjacent, and positions $n - 1$ and 0 are also adjacent. Each agent has a given position on the ring. This space has only one dimension, but our results may be extended to higher dimension spaces by applying the approach independently on each dimension.

The time is divided into discrete steps $1, 2, 3, \dots$. At the beginning of a given step t , an agent is at a given position. The possible actions are: *go left* (i.e. increase position), *go right* (i.e. decrease position) or *do not move*.

The current state of each agent is determined by the relative positions of other agents. However, we cannot associate a state to each combination of position of other agents, because of "combinatorial explosion". Thus, in order to limit the maximal number of states, each agent perceives an *approximation* of the positions of other agents. Besides, a state must not depend on the number of agents, in order to have a scalable model and to tolerate the loss of agents.

Thus, our state model is the following. The space is divided into groups of close positions called *sectors*. Each agent does not perceive the exact number of agents per sector, but the *fraction* of the total population in each sector. A *state* is given by the knowledge of the fractions of the total population in each sector with a precision of 10% (i.e. the possible values are multiples of 10%, rounded so that the sum of the fractions equals 100%). The choice of 10% precision here is an arbitrary value to reduce computational cost, this value can be optimized as an hyper-parameter.

The delimitation of the sectors is not absolute but *relative* to the position of each agent: each agent has its own sector delimitation centered around itself.

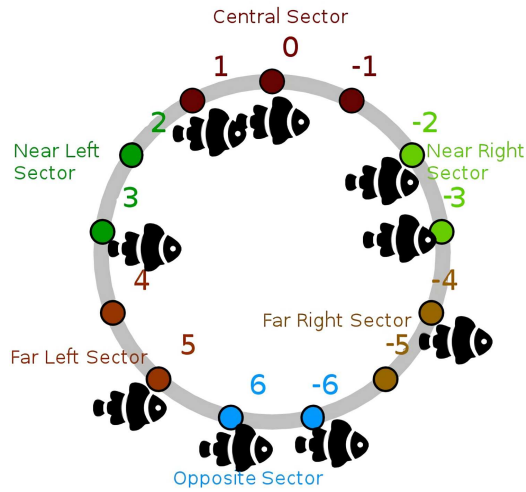


Figure 1: Default sector delimitation (on a ring of size 13). The *Central* sector contains 3 positions centered around 0 (i.e., the position of the current agent). *Near* sectors contain two positions each, adjacent to the Central sector. Same for *Far* sectors and the *Opposite* sector.

This delimitation is set to 6 sectors (as for the precision value of 10% described above, this choice can be left as an hyper-parameter, but optimizing it is out of the scope of this work). The first sector is centered around the agent position (its size corresponds to the size of the neighborhood where we expect the other agents to gather). This sector is the *Central* sector. The agents in the central sector of a given agent are called its *neighbors*. Two more sectors are adjacent to the central sector, the *Near Right* and *Near Left* sectors. The *Far left* and *Far Right* are a second layer after the near sectors. Finally, the *Opposite* sector is the sector diametrically opposed to the Central one. The exact size of each sector is a parameter of the problem, as well as the number of agents and the number of positions.

An example of sectors delimitation is given in Figure 1, for a ring of size 13.

3.2.3 Rewards

Each agent is rewarded if it has a large enough number of neighbors (i.e., more than a certain fraction of the total population is in its central sector). Each agent is penalized if it has not enough neighbors (i.e., less than a certain fraction of the population is in its central sector).

3.2.4 Learning process

The learning phase is organized as follows:

- The initial positions of agents are random, following a uniform distribution.
- At each step, each agent decides where to go with a ϵ -greedy policy.
- When all the decisions are taken, all the agents move simultaneously.
- After moving, they consider their environment, get rewards and update their Q-values with respect to these rewards.
- The learning phase is subdivided in *cycles* of several steps. At the end of each cycle, the position of agents is reset to random positions. This ensure that the environment is diverse enough to learn a robust behavior. After position reset, the agents can move again for another cycle.

The duration of a cycle is set proportional to the size of the ring (e.g. 5 times the size of the ring) in order to give enough time to the agents to gather: this time depends on the distance they have to travel, and this distance depends on the size of the ring. To update Q-values, Q-learning with eligibility traces is used. Eligibility traces are reset at the end of each cycle, and each time, a reward is given to an agent.

3.2.5 Problem

Intuitively, the goal is to make the agents learn a *gathering* behavior, that is: within a reasonable time in a same cycle, the agents become (and remain) reasonably close to each other. This criteria is voluntarily informal, and its satisfaction will be evaluated with several metrics in the next section.

More precisely, the problem consists in computing, for each agent, a value $Q(s, a)$ for each couple state-action (s, a) . This value indicates which action a to take in state s in order to increase the likelihood of obtaining a reward. For instance, the description given in subsection A states that, with probability $1 - \epsilon$, action a is chosen if it maximizes $Q(s, a)$ among all possible actions from state s .

Our objective is to verify experimentally that the Q-values learned in this fashion lead to an efficient gathering of the agents, i.e, that reinforcement learning, with rewards being given to actions that improve an agents' neighbourhood situation, lead to efficient gathering behaviours at the level of the group.

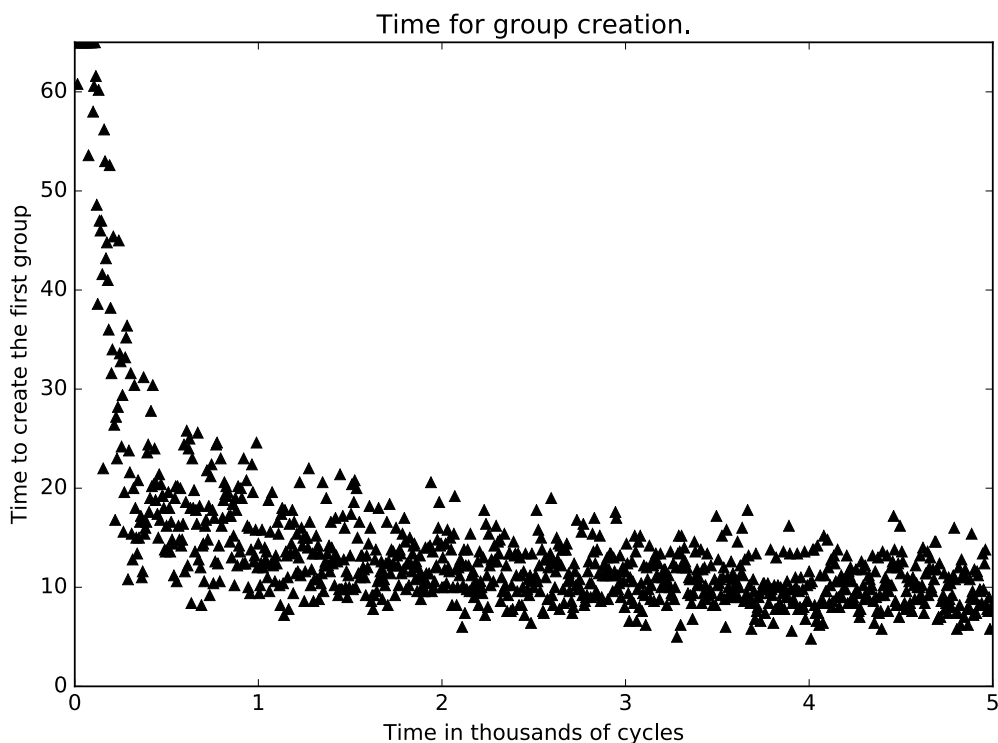


Figure 2: Time needed to form a group from random initial positions for 10 agents on a ring of size 13. Each point is the mean over 5 cycles of the average time to form a group (in the following, we simply say “average over X cycles”).

3.3 Results

We consider a ring of size 13, with a sector division such as described in Figure 1. A *group* exists if at least one agent has more than 80% of the population as neighbors. An agent is given a reward of value 100 if the fraction of neighbors is more than 80% of the population, and a penalty of value -5 if it is 10% or less.

The exploration rate is $\epsilon = 0.1$, the learning rate is $\eta = 0.1$ and the discount factor is $\gamma = 0.95$. The duration of a cycle is 65 steps (around 5 times the size of the ring), and the duration of the learning phase is 5000 cycles.

3.3.1 Results for 10 agents

We first consider a population of 10 agents. To assess the quality of the learned behavior, we compute several metrics. We first consider the time needed to form a group from random initial positions, and see how it evolves during the learn-

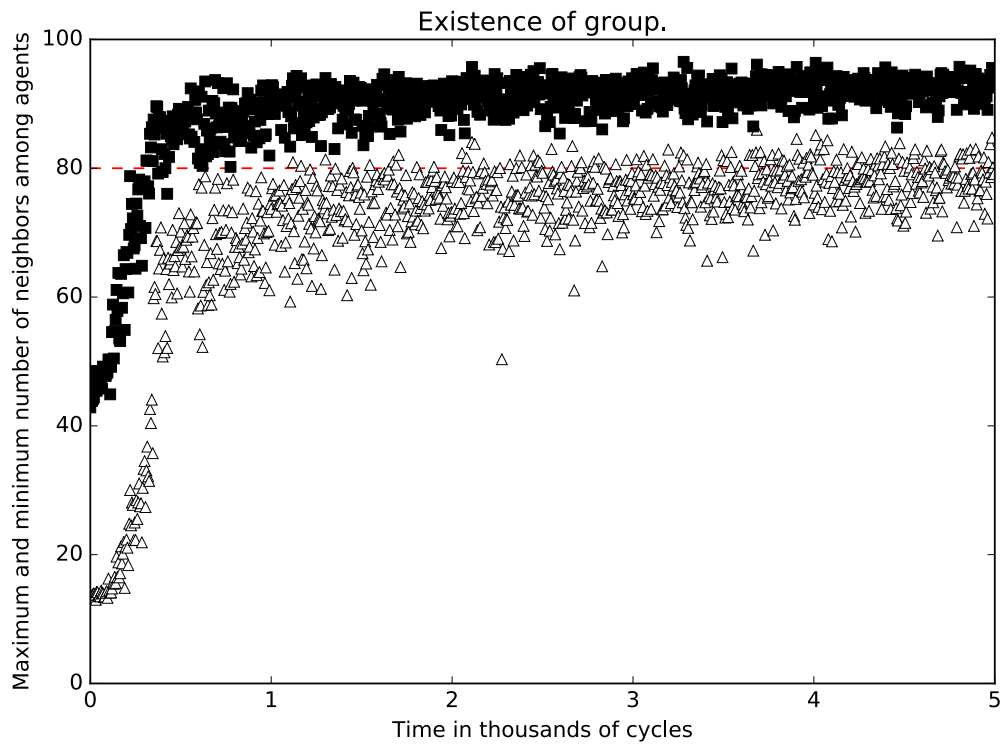


Figure 3: Maximum and minimum number of neighbors, in percent of the total population, during learning, for 10 agents on a ring of size 13. Maximum is black squares and minimum is white triangles. The dashed line is the minimum number of neighbors needed to be considered in the group: 80% of the total number of agents. Each point is an average over 325 steps, including time before creation of the first group.

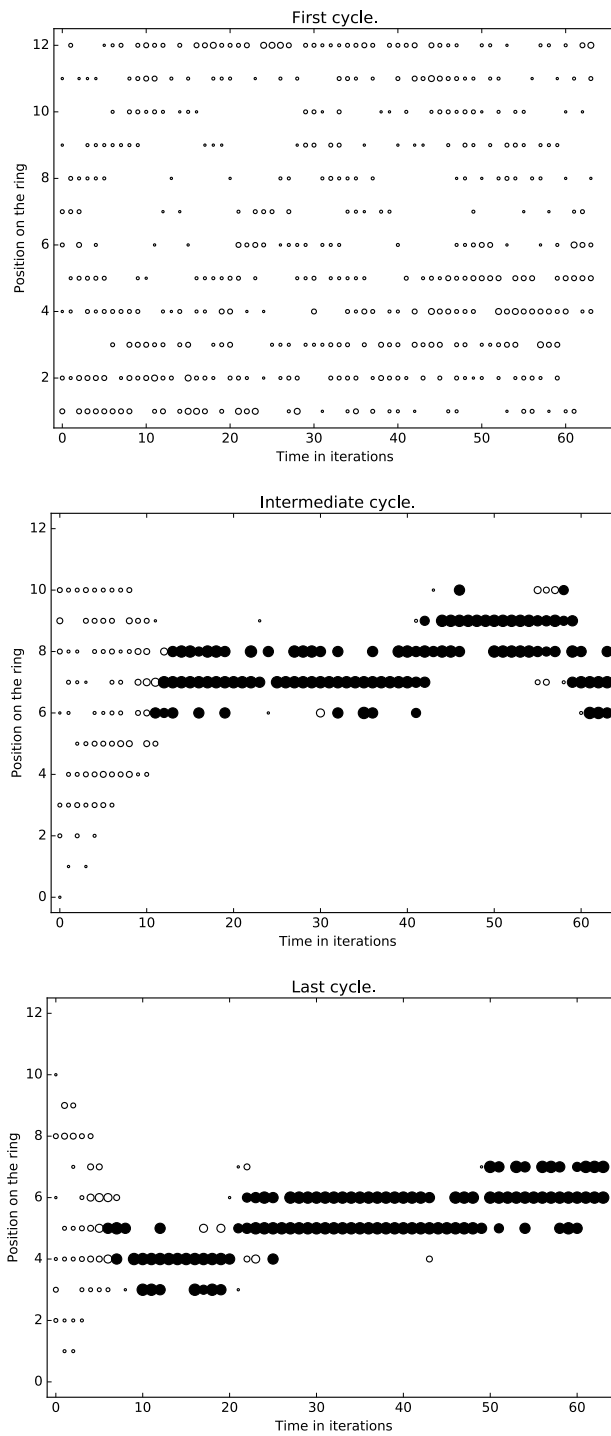


Figure 4: Evolution over time of the number of neighbors at each position of the ring during a cycle. Larger dots represent a higher number of neighbors. Positions where agents are considered to be in the group are in black, others in white.

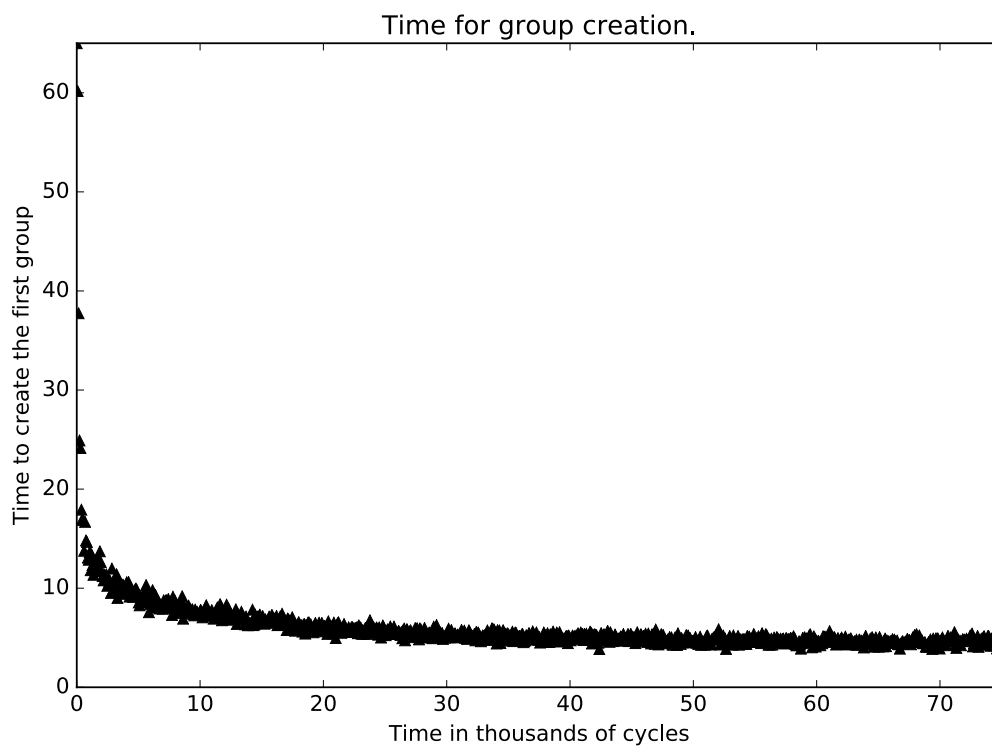


Figure 5: Time needed to form a group from random initial position for 10 agents on a ring of size 13. Each point is an average over 75 cycles. Learning phase is 75 000 cycles long.

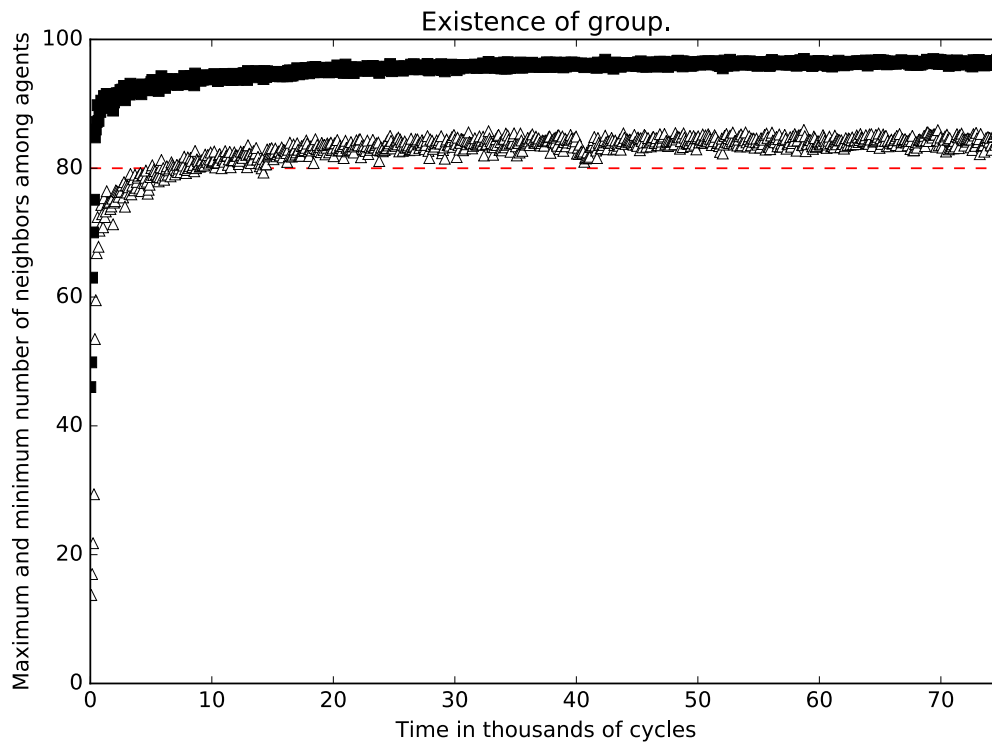


Figure 6: Maximum and minimum number of neighbors, during learning, for 10 agents on a ring of size 13. Maximum is black squares and minimum is white triangles. The dashed line is the minimum number of neighbors needed to be considered in the group (i.e. 80% of the total number of agents). Each point is an average over 75 cycles (4875 steps), including time before creation of the first group. The learning phase is 75 000 cycles long.

ing phase. Then, to ensure that groups are not only formed but also maintained, we observe the evolution of the number of neighbors among the population. To evaluate the learning qualitatively, we look at the exact behavior of agents at the beginning, middle and end of the learning phase. Finally, we study the impact of a longer learning phase.

Time to form a group. Figure 2 shows the time that agents need to gather and form the first group (i.e., at least one agent is rewarded), starting from random initial positions. We observe that this time decreases during the learning phase and stabilizes around 10 steps.

Number of neighbors. Figure 3 shows the minimal and maximal number of neighbors over all agents. When the maximal number of neighbors is above 80%, it means that a group exists. When the minimal number of neighbors is above 10%, it means that no agent is isolated; when it is above 80%, it means that all agents are in the group. We observe that the agents learn, not only to gather, but also to *maintain* the group and avoid being isolated. Indeed, the maximum number of neighbors is higher than 80% of the total number of agents, and the minimum is higher than 10%. We also observe that the minimum number of neighbors is close to 80% at the end of the learning phase. It means that even the agents that are not *always* in the group are often in it.

Note that these average values include the iterations starting from the beginning of each cycle, where the agents are not yet gathered (i.e. around 10 iterations at the end of the learning phase).

Qualitative evolution. Figure 4 contains three plots that show the qualitative evolution of the learning for three cycles, at the beginning, middle and end of the learning phase.

In the first figure (beginning of the learning phase), we observe that the agents are quite uniformly distributed: the circles are white and small, indicating few neighbors and no significant group formation.

In the second figure (middle of the learning phase), we observe that the agents converge to a same position, forming a group in approximately 10 steps. The large black circle indicate that at least 80% of the total number of agents are neighbors of the position, i.e. that a group exists. We can see that this group is maintained after its formation until the end of the cycle. We also observe that the group itself is slowly moving during the cycle, while being maintained. We notice that there are very few agents outside the group after its formation.

In the third figure (end of the learning phase), we observe that agents still converge to form a group, but the group is formed earlier than before (around 7

steps). The group is still maintained and still moves during the cycle. We can notice even less agents outside the group than before.

Longer learning phase. We finally study the impact of a longer learning phase: 75 000 cycles instead of 5000.

Figure 5 is the equivalent of Figure 2 for a longer learning phase. At the end of the learning, the agents are gathering faster (around 5 steps) and are less often outside of the group.

Figure 6 is the equivalent of Figure 3 for a longer learning phase. We observe that the minimum number of neighbors goes above 80%, which means that all the agents are in the group most of the time.

3.3.2 Scalability and comparison with a hardcoded algorithm

In the section, we explore the scalability and robustness properties of the aforementioned learning scheme. We show that the agents that have learned Q-values with default parameters in 75 000 cycles are able to gather with more agents *without* any new learning: we can take several agents that have learned in groups of 10 until we obtain a group of 100.

In a second time, we compare this behavior with a *hardcoded* gathering algorithm (i.e., where the behavior is written in advance and not learned).

- First, we compare the learned behavior to an algorithm that uses the *exact* and *absolute* positions of all the agents (by opposition to relative positions and approximations used during learning). With this algorithm, agents always move towards the barycenter [22, 64] of all the agents. As this algorithm has an exact view on the environment, the performances are 50% better.
- We then make a fairer and more meaningful comparison with an algorithm that uses the same perceptions as the learning algorithm. With an equally constrained perception of the environment, we get results that are similar to the learned algorithm (the learned algorithm even slightly better in terms of “time to form a group”). We thus show that, even with a relatively simple learning scheme, we can reach the same performances as a hardcoded behavior.

Note that, since the agents have already learned a behavior, there is no more “progression” visible on the plots.

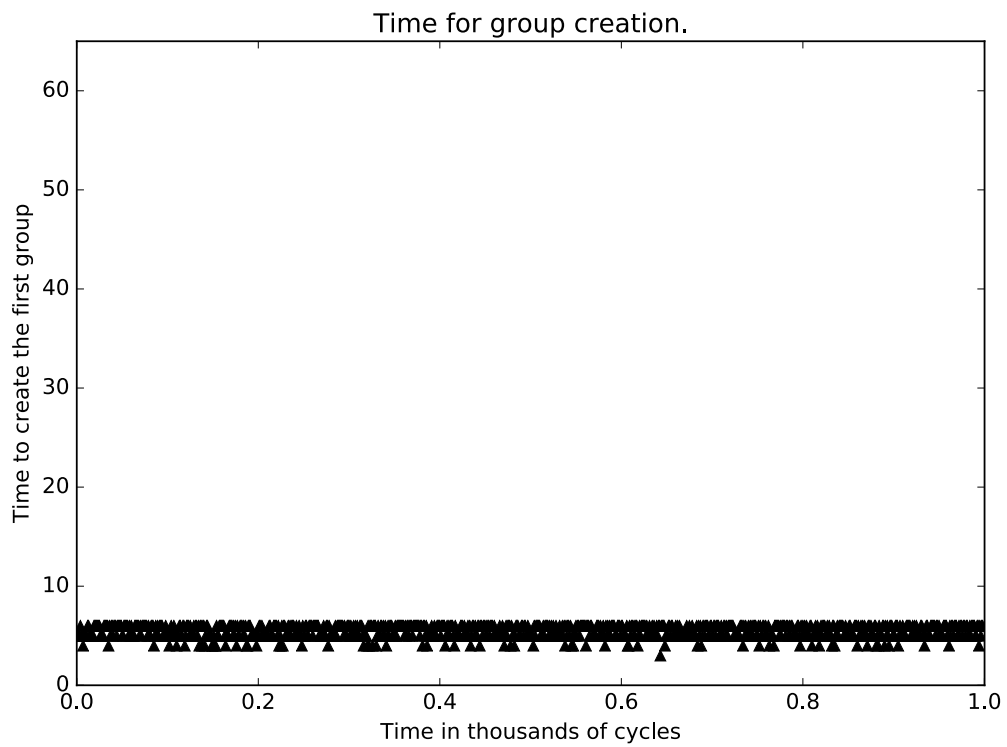


Figure 7: Time needed to form a group from random initial positions for 100 agents on a ring of size 13 with (hardcoded algorithm). Average is 5.4 steps, median is 5.0 steps and standard deviation is 0.6.

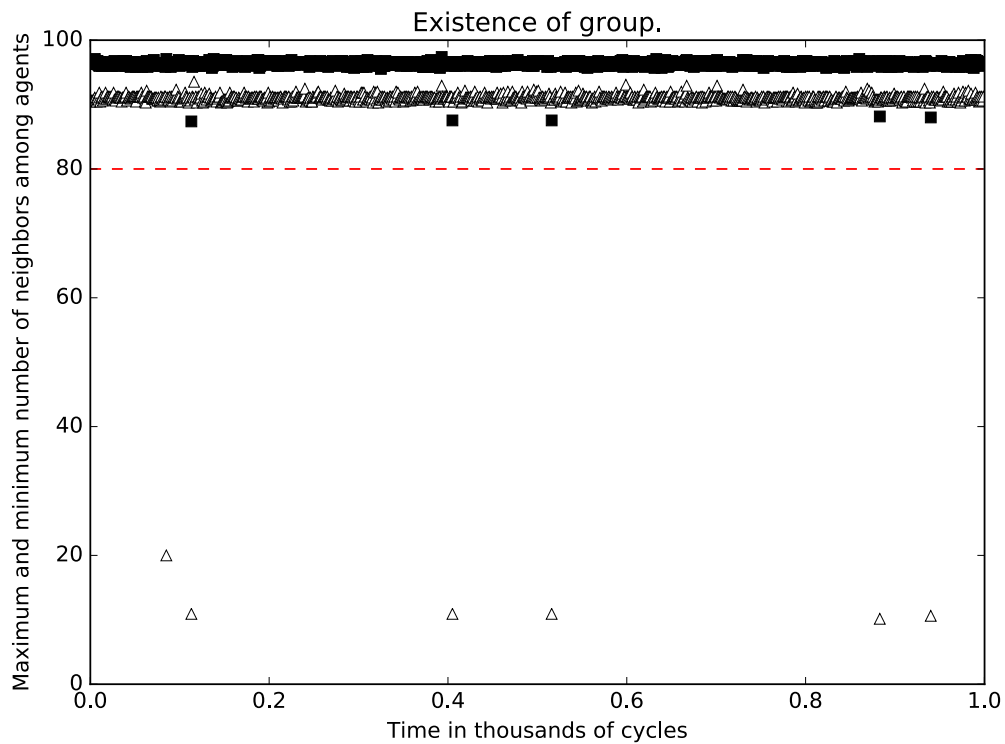


Figure 8: Maximum and minimum number of neighbors for 100 agents on a ring of size 13 (hardcoded algorithm). Maximum is black squares and minimum is white triangles. The dashed line is the minimum number of neighbors needed to be considered in the group. Each point is an average over a cycle (65 steps). Average is 90.6%, median is 91.1% and standard deviation is 6.1% for the minimum number of neighbors. Average is 96.3%, median is 96.3% and standard deviation is 0.7% for the maximum number of neighbors.

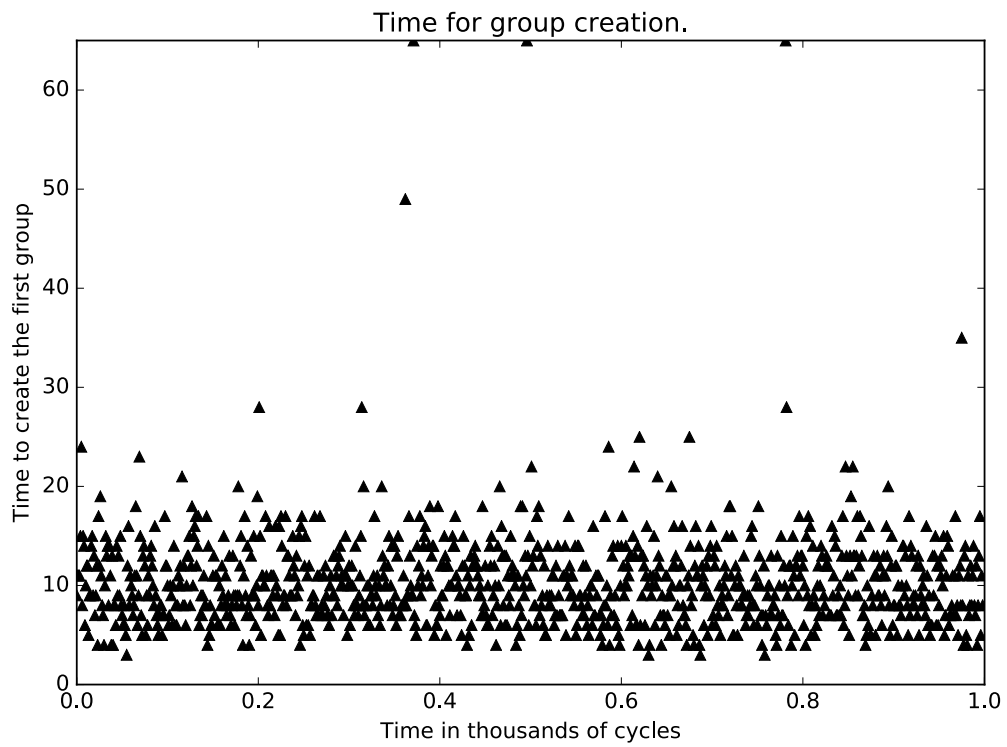


Figure 9: Time needed to form a group from random initial positions for 100 agents on a ring of size 13 (learned behavior). Average is 10.4 steps, median is 10.0 steps and standard deviation is 5.1.

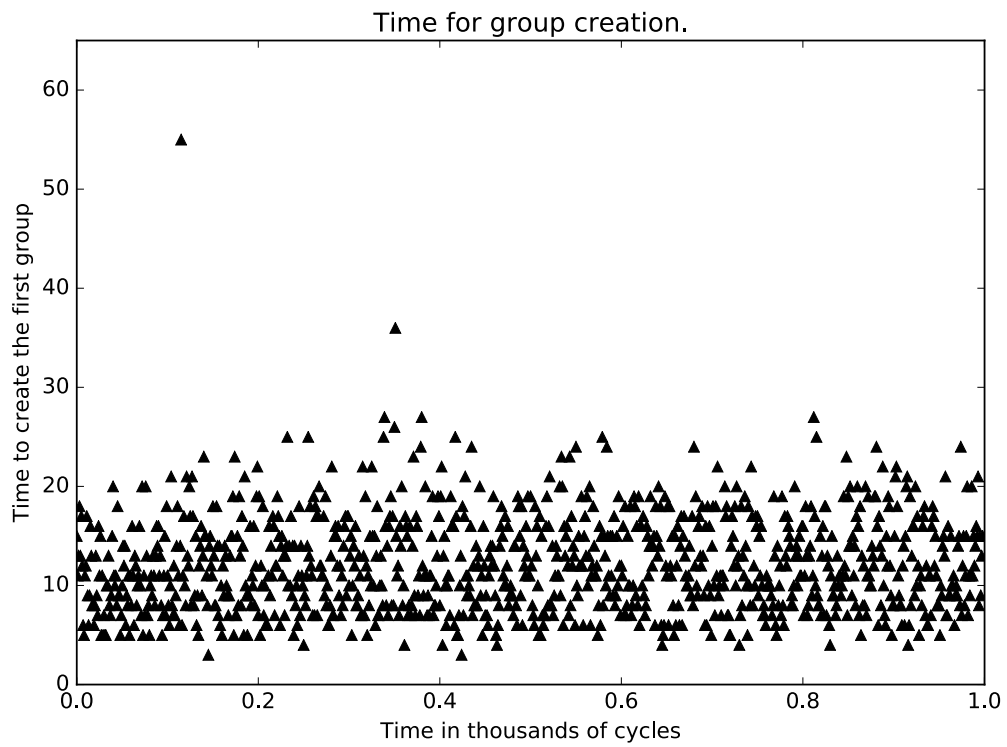


Figure 10: Time needed to form a group from random initial positions for 100 agents on a ring of size 13 (Q-hardcoded algorithm). Average is 12.1 steps, median is 11.0 steps and standard deviation is 4.9.

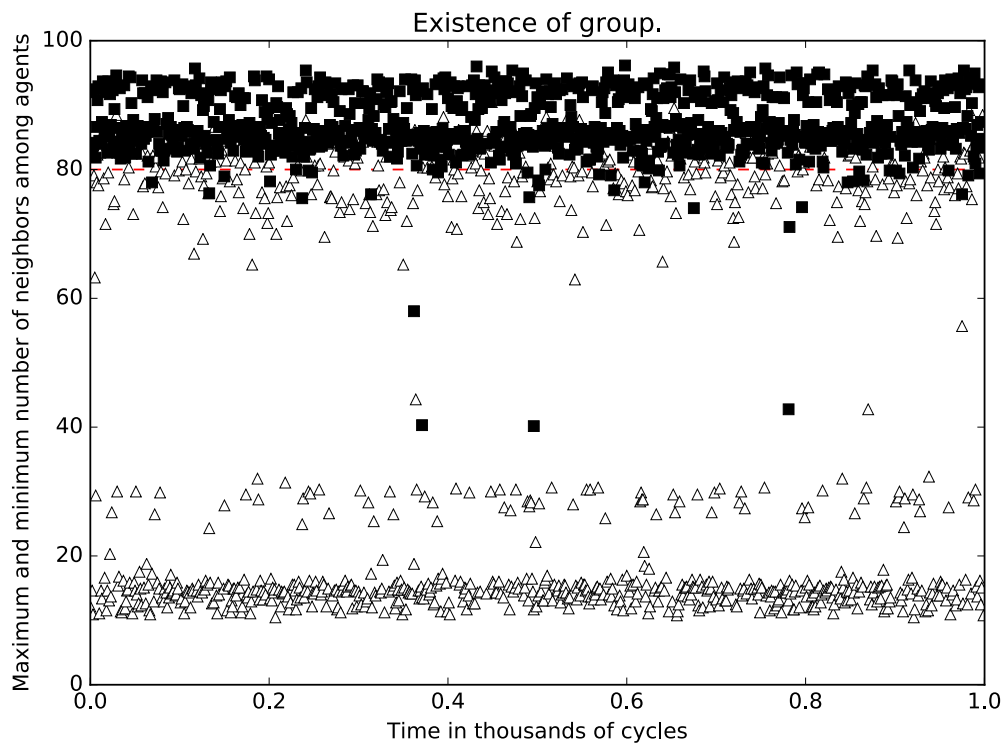


Figure 11: Maximum and minimum number of neighbors for 100 agents on a ring of size 13. Maximum is black squares and minimum is white triangles (learned behavior). The dashed line is the minimum number of neighbors needed to be considered in the group. Each point is an average over a cycle (65 steps). Average is 40.4%, median is 16.3% and standard deviation is 31.0% for min neighbor. Average is 87.1%, median is 86.0% and standard deviation is 5.1% for max neighbor.

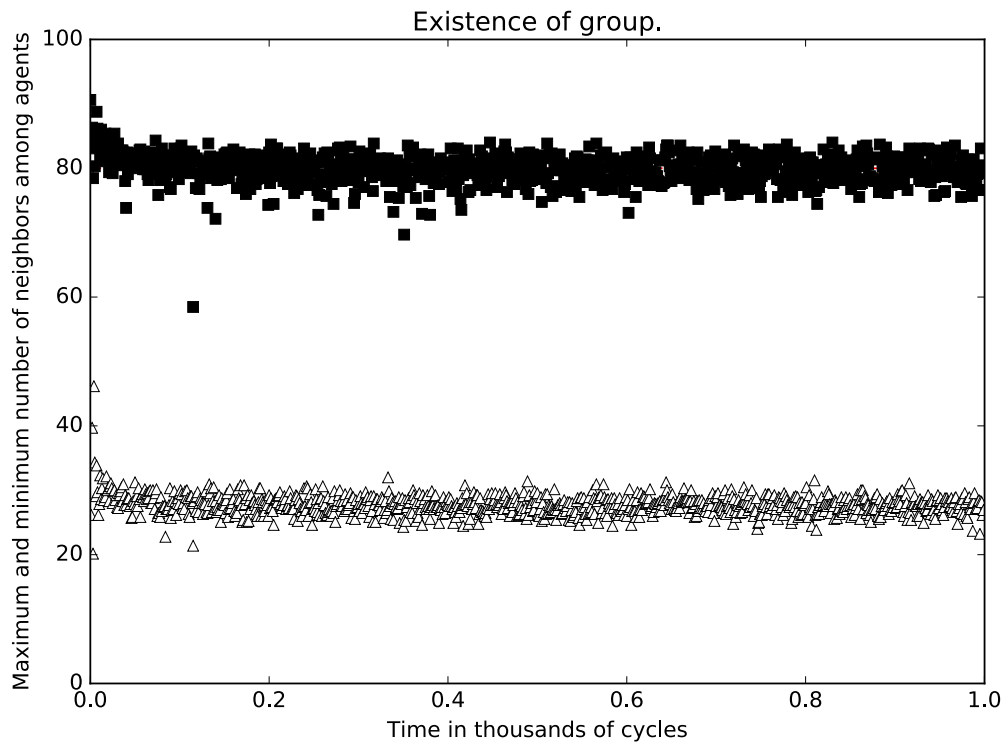


Figure 12: Maximum and minimum number of neighbors for 100 agents on a ring of size 13 (Q-hardcoded algorithm). Maximum is black squares and minimum is white triangles. The dashed line is the minimum number of neighbors needed to be considered in the group. Each point is an average over a cycle (65 steps). Average is 27.8%, median is 27.8% and standard deviation is 2.3% for the minimum number of neighbor. Average is 79.9%, median is 80.0% and standard deviation is 2.3% for the maximum number of neighbor.

Time to create a group for 100 agents. On Figure 9, we can see the time needed to form a group for 100 agents on a ring of size 13. Compared to the case with 10 agents, the time needed to form a group including 80% of the population is higher (around 10 steps in average). But the agents are still able to gather in a short time (the worst case is no more than 50 steps) most of the time: 997 times over 1000.

Number of neighbors for 100 agents. On Figure 11, we observe that the maximum number of neighbors is higher than 80% most of the time, which means that a group exists most of the time. We also observe that the minimum number of neighbors is often low. This means that a few agents, even if not isolated, are unable to join the main group.

Performances of the hardcoded algorithm. On Figure 7 and 8, we can observe that the hardcoded algorithm is better than the learned behavior. In average, the agents gather in 5 steps with a standard deviation of 0.6. Moreover, the maximum and minimum number of neighbors are very high (average: resp. 96% and 91%). However, these good results are only possible because this algorithm uses the exact and absolute positions of other agents.

Fairer comparison. To make a fairer comparison between hardcoded algorithm and learned behavior, we try to impose to the hardcoded algorithm the same constraints that were imposed to the learning algorithm: relative position, sector approximation and action choice with Q-values. To do so, we compute Q-values with the help of the hardcoded algorithm. Each agent decide how to act according to the hardcoded algorithm, and Q-values are computed along the sequence of actions determined by the hardcoded algorithm. It allows each agent to compute Q-values for couples (s, a) of states and actions. We call this algorithm the *Q-hardcoded algorithm*: the desired behavior is known in advance, but we imposes the same perception constraints to the agents than the learned behavior.

In Figure 10, we observe that the time needed to form a group has the same distribution as the learned behavior in Figure 9. The average time is even slightly better for the learned behavior (10 steps) than for the Q-hardcoded algorithm (12 steps). However, the standard deviation is slightly higher for learned behavior (5.1) than for the Q-hardcoded algorithm (4.9).

In Figure 10, we represent the distribution of the number of neighbors. Here again, we observe that the distribution is better for the learned behavior (Figure 11) than for the Q-hardcoded algorithm (Figure 12): the average of the maximum number neighbors is better (87% versus 80%) as well as the average of the

³Note that the figures are intentionally numbered to keep figures 9 and 10 (resp. 11 and 12) side by side, in order to have a clearer comparison between these figures.

minimum number of neighbors (40% versus 28%)⁴. However, the distribution of the number of neighbors is more sparse for the learned behavior.

3.4 Future works

In order to extend this work, it might be interesting to investigate how this multi-agent behavior emerges from the individual behavior of each agent, the difference of behavior between agents, and to quantify the importance of diversity in the behavior of agents.

Another direction to continue this work would be to devise a way for agents to design or learn their *own* approximations of their environment. This could be done through unsupervised learning [45], or with the help of the reward feedback from the environment (or by a combination of both). This automatic design of the perception approximation could allow to systematically find a good compromise between the reduction of the learning space and the capacity to perceive meaningful differences and learn complex tasks. Neural networks may be a good modular framework to model these approximations functions.

A major challenge would be to find a way to reuse the behavior learned with the old approximation, instead of re-learning the behavior from scratch whenever a change occurs in the approximation. The relative dynamics of the two timescales (one for the evolution of the approximation, and one for the evolution of the behavior) would also be of a particular importance.

Acknowledgment. This work was supported in part by the Swiss National Science Foundation (Grant 200021_169588 TARBDA).

References

- [1] O. Abul, F. Polat, and R. Alhajj. Multiagent reinforcement learning using function approximation. *IEEE Trans. Syst., Man, Cybern. C*, 30(4):485–497, 2000.
- [2] Yehuda Afek, Noga Alon, Omer Barad, Eran Hornstein, Naama Barkai, and Ziv Bar-Joseph. A biological solution to a fundamental distributed computing problem. *Science*, 331(6014):183–185, 2011.
- [3] Chrysovalandis Agathangelou, Chryssis Georgiou, and Marios Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 250–259, 2013.

⁴Many light triangles are between 60% and 80% on Figure 11, which explains the higher average value.

- [4] Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 1070–1078, 2004.
- [5] Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. Comput.*, 36(1):56–82, 2006.
- [6] Luzi Anderegg and Mark Cieliebak. The weber point can be found in linear time for points in biangular configuration. 02 2003.
- [7] Hideki Ando, Yoshinobu Oasa, Ichiro Suzuki, and Masafumi Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. Robotics and Automation*, 15(5):818–828, 1999.
- [8] Hideki Ando, Yoshinobu Oasa, Ichiro Suzuki, and Masafumi Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. Robotics and Automation*, 15(5):818–828, 1999.
- [9] Ronald C Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.
- [10] Mostafa D. Awgheda and Howard M. Schwartz. Exponential moving average based multiagent reinforcement learning algorithms. *Artificial Intelligence Review*, 45(3):299–332, oct 2015.
- [11] Ozalp Babaoglu, Geoffrey Canright, Andreas Deutsch, Gianni A. Di Caro, Frederick Ducatelle, Luca M. Gambardella, Niloy Ganguly, Márk Jelasity, Roberto Montemanni, Alberto Montresor, and Tore Urnes. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.*, 1(1):26–66, September 2006.
- [12] Pieter Beyens, Maarten Peeters, Kris Steenhaut, and Ann Nowe. Routing with compression in wireless sensor networks: a q-learning approach. In *Proceedings of the 5th European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS)*, 2005.
- [13] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *J. Discrete Algorithms*, 36:50–62, 2016.
- [14] Kálmán Bolla, Tamás Kovács, and Gábor Fazekas. Gathering of fat robots with limited visibility and without global navigation. In *Swarm and Evolutionary Computation - International Symposia, SIDE 2012 and EC 2012, Held in Conjunction with ICAISC 2012, Zakopane, Poland, April 29-May 3, 2012. Proceedings*, pages 30–38, 2012.
- [15] Zohir Bouzid, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Optimal byzantine-resilient convergence in uni-dimensional robot networks. *Theor. Comput. Sci.*, 411(34-36):3154–3168, 2010.

- [16] Bruno Bouzy and Marc Métivier. Multi-agent learning experiments on repeated matrix games. In *Proceedings of the 27 th International Conference on Machine Learning*,, 2010.
- [17] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intell*, 7(1):1–41, jan 2013.
- [18] Nicolas Bredèche, Evert Haasdijk, and Abraham Prieto. Embodied evolution in collective robotics: A review. *Front. Robotics and AI*, 2018, 2018.
- [19] Olivier Buffet, Alain Dutech, and François Charpillet. Shaping multi-agent systems with gradient reinforcement learning. *Auton Agent Multi-Agent Syst*, 15(2):197–220, jan 2007.
- [20] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, mar 2008.
- [21] Doran Chakraborty and Peter Stone. Convergence, targeted optimality, and safety in multiagent learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 191–198, 2010.
- [22] Benjamin Charlier. Necessary and sufficient condition for the existence of a fréchet mean on the circle. *ESAIM: Probability and Statistics*, 17:635–649, 2013.
- [23] Mark Cieliebak. Gathering non-oblivious mobile robots. In *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, pages 577–588, 2004.
- [24] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Solving the robots gathering problem. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, pages 1181–1196, 2003.
- [25] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Solving the robots gathering problem. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, pages 1181–1196, 2003.
- [26] Mark Cieliebak and Giuseppe Prencipe. Gathering autonomous mobile robots. In *SIROCCO 9, Proceedings of the 9th International Colloquium on Structural Information and Communication Complexity, Andros, Greece, June 10-12, 2002*, pages 57–72, 2002.
- [27] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, (s 746):752, 1998.
- [28] Reuven Cohen and David Peleg. Robot convergence via center-of-gravity algorithms. In *Structural Information and Communication Complexity, 11th International Colloquium, SIROCCO 2004, Smolenice Castle, Slovakia, June 21-23, 2004, Proceedings*, pages 79–88, 2004.

- [29] Reuven Cohen and David Peleg. Robot convergence via center-of-gravity algorithms. In *Structural Information and Communication Complexity, 11th International Colloquium, SIROCCO 2004, Smolenice Castle, Slovakia, June 21-23, 2004, Proceedings*, pages 79–88, 2004.
- [30] Jurek Czyzowicz, Leszek Gasieniec, and Andrzej Pelc. Gathering few fat mobile robots in the plane. *Theor. Comput. Sci.*, 410(6-7):481–499, 2009.
- [31] Gianlorenzo D’Angelo, Gabriele Di Stefano, Ralf Klasing, and Alfredo Navarra. Gathering of robots on anonymous grids and trees without multiplicity detection. *Theor. Comput. Sci.*, 610:158–168, 2016.
- [32] F. F. Darling. Bird flocks and the breeding cycle; a contribution to the study of avian sociality. *Oxford, England: Macmillan*, 1999.
- [33] Xavier Défago, Maria Gradinariu Potop-Butucaru, Julien Clément, Stéphane Messika, and Philippe Raipin Parvédy. Fault and byzantine tolerant self-stabilizing mobile robots gathering - feasibility study -. *CoRR*, abs/1602.05546, 2016.
- [34] Anders Dessmark, Pierre Fraigniaud, Dariusz R. Kowalski, and Andrzej Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46(1):69–96, 2006.
- [35] Yoann Dieudonné and Franck Petit. Self-stabilizing deterministic gathering. In *Algorithmic Aspects of Wireless Sensor Networks, 5th International Workshop, ALGOSENSORS 2009, Rhodes, Greece, July 10-11, 2009. Revised Selected Papers*, pages 230–241, 2009.
- [36] Stéphane Doncieux, Nicolas Bredèche, Jean-Baptiste Mouret, and A. E. Eiben. Evolutionary robotics: What, why, and where to. *Front. Robotics and AI*, 2015, 2015.
- [37] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73 – 81, 1997.
- [38] David J. Finton. When do differences matter? on-line feature extraction through cognitive economy. *Cognitive Systems Research*, 6(4):263–281, dec 2005.
- [39] Paola Flocchini, Evangelos Kranakis, Danny Krizanc, Nicola Santoro, and Cindy Sawchuk. Multiple mobile agent rendezvous in a ring. In *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, pages 599–608, 2004.
- [40] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.*, 337(1-3):147–168, 2005.
- [41] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.*, 337(1-3):147–168, 2005.
- [42] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

- [43] Matthew R. Glickman and Katia P. Sycara. Evolutionary search, stochastic policies with memory, and reinforcement learning with hidden state. In *ICML*, 2001.
- [44] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS'14*, pages 3338–3346, Cambridge, MA, USA, 2014. MIT Press.
- [45] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.
- [46] Simon Haykin. *Neural Networks and Learning Machines Third Edition*. 2008.
- [47] Junius Ho, Daniel W Engels, and Sanjay E Sarma. Hiq: a hierarchical q-learning algorithm to solve the reader collision problem. In *International Symposium on Applications and the Internet Workshops (SAINTW'06)*, pages 4–pp. IEEE, 2006.
- [48] T. Horiuchi, A. Fujino, O. Katai, and T. Sawaragi. Fuzzy interpolation-based q-learning with profit sharing plan scheme. In *Proceedings of 6th International Fuzzy Systems Conference*. Institute of Electrical & Electronics Engineers (IEEE), 1997.
- [49] Tomoko Izumi, Taisuke Izumi, Sayaka Kamei, and Fukuhito Ooshita. Time-optimal gathering algorithm of mobile robots with local weak multiplicity detection in rings. *IEICE Transactions*, 96-A(6):1072–1080, 2013.
- [50] Sayaka Kamei, Anissa Lamani, Fukuhito Ooshita, and Sébastien Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In *Structural Information and Communication Complexity - 18th International Colloquium, SIROCCO 2011, Gdansk, Poland, June 26-29, 2011. Proceedings*, pages 150–161, 2011.
- [51] Soumya Kar, José M. F. Moura, and H. Vincent Poor. Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus+innovations. *IEEE Transactions on Signal Processing*, 61(7):1848–1862, apr 2013.
- [52] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring. *Theor. Comput. Sci.*, 411(34-36):3235–3246, 2010.
- [53] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390(1):27–39, 2008.
- [54] Dariusz R. Kowalski and Andrzej Pelc. Polynomial deterministic rendezvous in arbitrary graphs. In *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings*, pages 644–656, 2004.
- [55] Hung Manh La, Ronny Lim, and Weihua Sheng. Multirobot cooperative learning for predator avoidance. *IEEE Transactions on Control Systems Technology*, 23(1):52–63, jan 2015.

- [56] Paul Levi and Serge Kernbach. *Symbiotic Multi-Robot Organisms - Reliability, Adaptability, Evolution*, volume 7 of *Cognitive Systems Monographs*. Springer, 2010.
- [57] Shouwei Li, Friedhelm Meyer auf der Heide, and Pavel Podlipyan. The impact of the gabriel subgraph of the visibility graph on the gathering of mobile autonomous robots. In *Algorithms for Sensor Systems - 12th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2016, Aarhus, Denmark, August 25-26, 2016, Revised Selected Papers*, pages 62–79, 2016.
- [58] John Loch and Satinder P Singh. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML*, pages 323–331, 1998.
- [59] Liam MacDermed. Scaling up game theory: Achievable set methods for efficiently solving stochastic games of complete and incomplete information. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, 2011.
- [60] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
- [61] Koichiro Morihiko, Tejiro Isokawa, Haruhiko Nishimura, and Nobuyuki Matsui. Characteristics of flocking behavior model by reinforcement learning scheme. In *2006 SICE-ICASE International Joint Conference*. Institute of Electrical & Electronics Engineers (IEEE), 2006.
- [62] Jean Oh. *Multiagent Social Learning in Large Repeated Games*. PhD thesis, Pittsburgh, PA, USA, 2009. AAI3414065.
- [63] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P. How. Learning to teach in cooperative multiagent reinforcement learning. *CoRR*, abs/1805.07830, 2018.
- [64] Xavier Pennec. Probabilities and statistics on riemannian manifolds: Basic tools for geometric measurements. In *NSIP*, pages 194–198. Citeseer, 1999.
- [65] HL Prasad, Prashanth LA, and Shalabh Bhatnagar. Two-timescale algorithms for learning nash equilibria in general-sum stochastic games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1371–1379. International Foundation for Autonomous Agents and Multiagent Systems, 2015.
- [66] Giuseppe Prencipe. On the feasibility of gathering by autonomous mobile robots. In *Structural Information and Communication Complexity, 12th International Colloquium, SIROCCO 2005, Mont Saint-Michel, France, May 24-26, 2005, Proceedings*, pages 246–261, 2005.
- [67] Hussein Saad, Amr Mohamed, and Tamer ElBatt. Cooperative q-learning techniques for distributed online power allocation in femtocell networks. *Wirel. Commun. Mob. Comput.*, 15(15):1929–1944, feb 2014.

- [68] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1889–1897, 2015.
- [69] Z. Shi, J. Tu, Q. Zhang, X. Zhang, and J. Wei. The improved q-learning algorithm based on pheromone mechanism for swarm robot system. In *Proceedings of the 32nd Chinese Control Conference*, pages 6033–6038, July 2013.
- [70] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [71] Herbert A Simon. Why should machines learn? In *Machine learning*, pages 25–37. Springer, 1983.
- [72] Kazuo Sugihara and Ichiro Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *J. Field Robotics*, 13(3):127–139, 1996.
- [73] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Netw.*, 9(5):1054–1054, sep 1998.
- [74] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999.
- [75] Ming Tan. Multi-agent reinforcement learning: Independent versus cooperative agents. In *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 330–337, 1993.
- [76] Ben-Nian Wang, Yang Gao, Zhao-Qian Chen, Jun-Yuan Xie, and Shi-Fu Chen. A two-layered multi-agent reinforcement learning model and algorithm. *Journal of Network and Computer Applications*, 30(4):1366–1376, nov 2007. Competitive.
- [77] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- [78] Trevor J. Willis, Russell B. Millar, and Russell C. Babcock. Detection of spatial variability in relative density of fishes: comparison of visual census, angling, and baited underwater video. *Marine Ecology Progress Series*, 198:249–260, 2000.
- [79] Ping Xuan, Victor Lesser, and Shlomo Zilberstein. Communication in multi-agent markov decision processes. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 467–468. IEEE, 2000.
- [80] Zhen Zhang, Dongbin Zhao, Junwei Gao, Dongqing Wang, and Yujie Dai. FMRQ—a multiagent reinforcement learning algorithm for fully cooperative tasks. *IEEE Trans. Cybern.*, pages 1–13, 2016.

- [81] Mortaza Zolfpour-Arokhlo, Ali Selamat, Siti Zaiton Mohd Hashim, and Hossein Afkhami. Modeling of route planning system based on q value-based dynamic programming with multi-agent reinforcement learning algorithms. *Engineering Applications of Artificial Intelligence*, 29:163–177, mar 2014.