Decentralized estimation of forest fire spread using reactive and cognitive mobile sensors

Guillaume Schlotterbeck, Clément Raievsky, Laurent Lefevre

Noname manuscript No.
(will be inserted by the editor)

Decentralized estimation of forest fire spread using reactive and cognitive mobile sensors

SCHLOTTERBECK Guillaume · RAÏEVSKY Clement · LEFÈVRE Laurent

Received: date / Accepted: date

Abstract This paper presents a study of the ability to build an observer for a complex system using a decentralized multi-agent system for the coordination of mobile sensors. The environment is modeled using a CA model representing forest fire spread. The initial distribution for the different species in the vegetation is generated using a Perlin algorithm. Implementation is realized on GPGPU. A coherence measure for the observation error is defined. The observation itself is realized with mobile sensors and a decentralized coordination of the trajectories. We analyze the balance between individual and collective behaviours of agents which is required to achieve the best performance with respect to the chosen coherence measure. Two kinds of agent' behaviour are studied: reactive and cognitive.

Keywords Mobile sensors · multi agent systems · state estimation · distributed parameters system · forest fire spread · cellular automata · reactive/cognitive agents

1 Introduction

In this paper, we present our work on quantifying the ability of a multi-agent system (MAS) to observe a dynamic 2D complex system. This study is a continuation to a paper which present an decentralized estimator of fire forest propagation with reactive multi-agents system (Schlotterbeck et al, 2016). By observing a complex system, we mean building a representation of its internal state based on partial and local measures of it. A multi-agent system (MAS) is a set of artificial entities (agents) which interact with each other. These entities' action selection is autonomous and based on a partial representation of their environment, on internal states, and on the interactions that occur between them. In the observation case, the agents have the sole task to gather information about their environment and can thus be seen as mobile sensors.

Observation of complex system is an interesting problem because understanding and control of this kind of systems critically depends on a dynamic representation of their states as precise as possible.

Sensors' placement is a crucial point of the 2D complex systems observation problem. Existing work tackling this point focus on methods to maximize an observability measure. For example, Waldraff et al (1998) use an observability matrix singular value based measure, an observability grammian singular value based measure or the Popov-Belevitch Hautus rank test. Empirical approaches based on genetic algorithms are also presented in Liu et al (2008). The main differences with our approach is that they are off-line and only consider static sensors.

To evaluate the ability of a MAS to observe a complex two dimensional system, we chose to explore different mobile sensors placement strategies in a simulated forest where a fire is propagating. To simulate this complex system, we used Cellular Automata (CA). CA are particularly suitable for modelling dynamic systems at a mesoscopic[1]level, especially to represent the spreading phenomenon (see El Yacoubi and El Jai, 2002). This approach to use CA to simulate a system to study estimation strategies was also used for traffic flow systems (Barbieri et al, 2013; Sartoretti et al, 2012). They

Univ. Grenoble Alpes, LCIS, F-26000 Valence, France
E-mail: guillaume.schlotterbeck@gmail.com
E-mail: clement.raievsky@lcis.grenoble-inp.fr
E-mail: laurent.lefevre@lcis.grenoble-inp.fr

[1] In terms of behavioural rules mimicking the physical interactions and translating them into a fully discrete universe (Chopard, 2012)

proposed methods to estimate it with decentralized approach. The CA they defined allowed them to analyse theoretically the efficiency of their approach.

Our main objective is both to determine and maximize the ability of a group of agents to build an estimation of the state of a complex dynamical system as close as possible to its actual state. To do so, we study the relationship between the collective behaviour of a group of autonomous mobile sensors and the difference between the estimate they build of a system and its actual state. In our simulations, the agents move in the simulated forest while it is burning and collectively build a virtual map of its state according to their local observations. This map is the estimate they build and changes only when the agents update it with their observations. Agents have a write-only access to this map and it is thus a omniscient tool, only used for evaluation purpose. As we are using the MAS approach, the behaviour of the agents is decentralized: each agent has its own local perceptions of the forest and acts accordingly. The challenge is thus to find a local behaviour that allows the group to reach its global goal to build an estimation as precise as possible.

We explored two main strategies to coordinate the agents' behaviour and compared them with two reference, pseudo-random behaviours. Agents using the first strategy have a "purely reactive" behaviour (Ferber, 1999; Franklin and Graesser, 1996). They only take into account their current perceptions to decide their next action. Neither communication nor internal representation were used. Their next action is a weighted combination of influences similar to the one used in boids (Reynolds, 1987). This strategy is described in Section 4.4.

Agents using the second strategy have three more capabilities compared to the previous ones. First they have an internal representation of the state of the environment. Second they are able to communicate with each other when they get close enough. Third, they have a model of the fire propagation which is the same as the one used in the simulated environment. Due to these abilities, these agents are considered as "cognitive" or "deliberative" (Franklin and Graesser, 1996).

The main contribution of this paper is to quantitatively compare decentralized mobile sensors coordination strategies in terms of estimation quality.

In Section 2, we present a formal definition of the CA used to simulate the environment. Then we present in Section 3 the observability measure we used to evaluate the different coordination strategies of the MAS which are described in Section 4. In Section 5, we describe the implementation of the whole system and the results obtained through simulations. Finally we analyse those results and draw some conclusion about our approach in Section 5.3.

2 Forest fire front propagation simulation using CA

The environment we defined is a 2D domain. Geometrically, it is a rectangular surface. The width of the environment is noted W and its heigh H.

2.1 Cellular Automaton Definition

Cellular Automata (CA) may be defined by a quintuple (L, S, N, a, C) based on the definition presented in Yacoubi (2008) where:

- L is a 2-dimensional lattice of cells c. $L = \{0, 1, ..., W-1\} \times \{0, 1, ..., H-1\}$. So a point c on this space have two components noted c_w and c_h. The total number of cells in the environment is defined as $N_c = W.H$. We denote $c_i \in [0; W.H-1]$ the index of the cell. We have $c_i = c_w + c_h.W$.
- S denotes a discrete composite state set: $S = \mathbb{N}^4$. The different sub-states are:
  - $x_T \in \mathbb{N}$: The quantity of remaining combustible.
  - $x_I \in \mathbb{N}$: The fire intensity (heat).
  - $x_F \in \mathbb{N}$: The necessary of neighbour's heat (intensity) to ignite the cell.
  - $x_R \in \mathbb{N}$: The neighbourhood's radius.

  Sub-states x_F and x_R are constant. They are similar to parameter linked to a cell.
  We note $x(c,t) \in S$ the state of the cell $c \in L$ at the time $t \in \mathbb{N}$.
  The global state at time t is noted $X(t) \in S^{W.H}$. The i^{th} element of $X(t)$ is $x(c_i, t)$. To simplify notation, we denote :
  - $x(c) = x(c,t)$ for the local state.
  - $x_T(c) = x_T(c,t)$ for sub-states.
  - $X_0 = X(0)$ for initial condition .
  - $X = X(t)$ for global state.
  - $X^+ = X(t+1)$ for next global state.
- N is a mapping which defines the cell's neighbourhood set[2]. It is given by:

$$N(c) = \{c' \in L \mid \|c' - c\| \leq x_R(c)\}$$

[2] Computational efficiency considerations led us to include the neighbourhood radius as a part of the state of a cell. Indeed, it varies with the type of vegetation. An alternative (and more usual) approach would be to define directly the neighbourhood of the largest possible one for all possible types of vegetation and the evolution rule accordingly.

- a is a transition function which allows us to calculate the cell's state at time $t+1$. This function depends on the cell's neighbourhood state.
  It can be defined by $x(c, t+1) = a(x(N(c), t))$ and is described below.
  So, the global definition is $X^+ = A(X)$.
- C is the output operator. It allow us to construct a measurement vector noted Y from the state of the system.
  We have $Y(t) = Y = C \cdot X$
  Finally, we give the global definition of the cellular automaton :

$$\begin{cases} X^+ = A(X) \\ Y \quad = C \cdot X \end{cases} \tag{1}$$

The algorithm of the function $a(\cdot)$ is as follow: when a plant ignites, its heat intensity (x_I) is incremented at each time step and its remaining combustible (x_T) is decremented. To ignite, a plant has to be in a hot enough neighbourhood, *i.e.* when the sum of cells intensities (x_I) in its neighborhood (N) in range (x_R) is greater or equal to its flammability (x_F). When all combustible in a cell has burned, the fire is finished and the heat intensity falls to 0.

Our aim wasn't to design and test a realistic forest fire propagation model. The presented model is designed to be as simple as possible while displaying complex enough behaviours; making it interesting, not trivial, to observe.

The next section presents our approach to generate initial condition of the environment.

2.2 Initial condition

To evaluate the ability of a MAS to build a precise estimate, the observed system has to be sufficiently complex. To generate rich dynamics in a CA, the environment is required to exhibit local differences (Green, 1989). We therefore chose to use three virtual plants with specific dynamics and a probabilistic initial distribution of these plants in the environment based on Perlin Noise[3].

The three kinds of plant we defined are :

- `Tree`: Has the biggest starting amount of combustible ($x_T = 200$, *i.e.* a tree burns during 200 iterations). Has a flammability of 70 ($x_F = 70$) and a range of 2 ($x_R = 2$). *i.e.* A tree ignites if the sum of fire intensities within a range of 2 cells is at least 70.

[3] Perlin noise is an algorithm able to randomly generate coherent and continuous noise in N dimensions. For instance, it has been used to generate realistic heights for mountains in video games worlds (Parberry, 2014).

Fig. 1: Example of initial condition of the cellular automata

- `Shrub`: $x_T = 70$. It has a range of 1 and a flammability of 2.
- `Grass`: $x_T = 1$. $x_R = 1$ and $x_F = 1$.

Figure 1 shows an example of initial condition. White, black and gray pixels represent grass, trees, and shrub, respectively.

2.3 Implementation

In our CA implementation, all cells are synchronously updated using the same algorithm. The computation of this kind of SIMD (*Single instruction Multiple Data*) algorithm benefits greatly from GPGPU (General-purpose processing on graphics processing units) (see LAVILLE, 2013). Figure 2 shows six successive states of the fire front propagation. We can see in these snapshots that the shape of the fire front is chaotic and sufficiently complex to be interesting. The next section presents the observability problem we try to tackle and mathematical tools we use to evaluate the quality of our solution.

3 Observation quality evaluation

System observation consists in building a representation of the internal state of a system. Since the information available on a system is fundamentally incomplete and uncertain, the built representation is always an estimation.

To compensate this lack of information, control theory offer many mathematical methods based on a dynamic model of the system. These methods make it possible, for example and under some constraints, to determine some unmeasurable states and to predict the short-term evolution of the system's state. The estimation built from measures and mathematical methods from control theory are useful to control, stabilize, and identify systems.

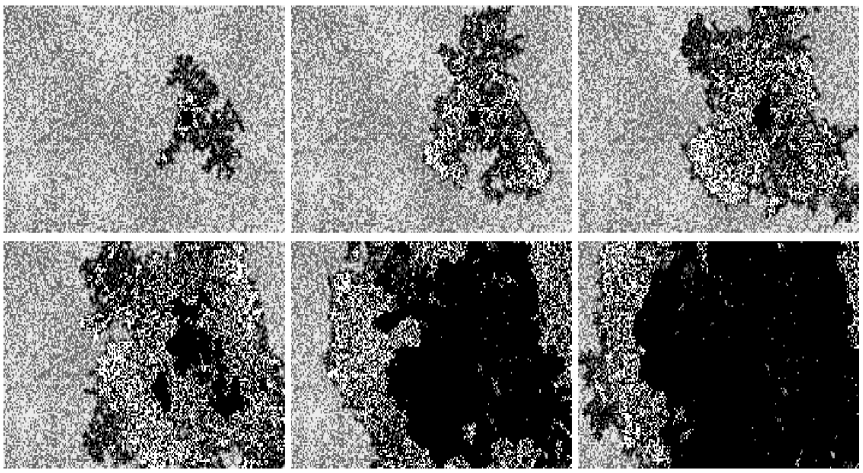

Fig. 2: Forest fire front propagation. Black pixels represent burned cells (ashes), white ones burning cells and gray ones plants. Among those, darkest pixels represent trees, and lightest ones grass.

One critical question in all these tasks is the placement of the sensors (see Waldraff et al, 1998).

Once again, control theory scientists have proposed mathematical methods which can determine the optimal positions of sensors in term of estimation quality. However, these methods only apply on some classes of problems and some of their applicability constraints are strong. Many real systems do not satisfy these constraints and in these cases there is not yet systematic methods to find optimal sensor placement. These systems are said unobservable and it is no possible to get a precise estimation of their states. The same problem appears when systems are too complex or too large.

Fire forest propagation, as we are modeling it, is of this kind and there is no applicable control theory method to get a precise estimate of it. Propagation and evolution rules are strongly non-linear because of the ignition thresholds (x_F). Further, the CA is constituted of hundreds of thousands of cells, giving huge dimensions to the global system's state vector. Each cell is directly or indirectly linked to all the others due to the neighbourhood propagation rule, making the system's evolution even more complicated to model.

While approaches based on genetic algorithms try to tackle this problem (Liu et al, 2008), they can not yet guarantee that the found solution is optimal.

Next section formally defines state estimation and describe a method to quantify the difference between an estimation and the actual state of a system.

3.1 Estimation definition

An estimate – estimation of a system's state – is a vector of the same dimension as the system's state vector X and is denoted $\hat{X}$. It is defined on $S^{W \cdot H}$.

An estimator or state observer is the set of processes used to determine $\hat{X}$ values over time. Estimators compute the state estimate from current and past measures and from a dynamic model of the system.

Given the CA's recursive evolution equation 1, the global definition of the observer is :

$$\hat{X}^+ = A_o(\hat{X},\ Y) \tag{2}$$

To maximize the estimator quality, the main objective is to find A_o such as $\hat{X}$ is the closet to X over time. To find this A_o, it is therefore required to have a measure of the difference between X and $\hat{X}$. The next part presents our approach to get this difference.

3.2 Evaluation of the difference between state and estimation

To get a measure of the difference between the simulated state and the agent-built estimation, we chose to average the difference over all environment cells at time t. This choice render the measured difference independent from the environment size and do not give any priority to any particular cells.

We denote this difference by $D(t)$.

The difference between the estimate and the "real" simulated state for a particular cell, also called the error on this cell, is denoted $d(\cdot, \cdot)$.

$D(t)$ is therefore defined as:

$$D(t) = \frac{1}{N_c} \sum_{i=0}^{N_c-1} d\ (x(c_i,t),\ \hat{x}(c_i,t)\ ) \tag{3}$$

There are several ways to define $d(\cdot,\cdot)$ depending on what is to be measured. To get the ratio of cells on which there are at least one difference between the values of the estimate and the state, the following definition should be used:

$$d(x,\hat{x}) = \begin{cases} 1 & \text{if } x_T \neq \hat{x}_T \text{ or } x_I \neq \hat{x}_I \\ 0 & \text{else} \end{cases} \tag{4}$$

This measure is the more frequently used in CA-related work (Yacoubi, 2008) and is the one we are using.

3.3 Estimation quality measure

In order to quantify the accuracy of an observer, we used an empirical approach since there is no analytical tools to do so. Our empirical approach is based on an offline evaluation of the average difference $D(t)$ between a simulated environment and an estimation of it, collectively built by a MAS.

The estimation quality q of a particular MAS behaviour is calculated according to the following equation:

$$q = \frac{1}{t_f+1} \sum_{t=0}^{t_f} D(t) \tag{5}$$

where t_f is the time at which we consider the simulation as completed, when there is no burning cell in the environment anymore, in our case.

The next section describes the MAS implementation we used as an estimator of the simulated fire front.

4 Distributed estimation using a multi-agent system

4.1 Definition

We define an agent as an entity moving in the environment simulated by the CA described earlier. The agents' curvilinear speed v_0 is constant and the same for all agents. This hypothesis put aside the problem of the agents' linear speed regulation. These agents may be seen as small unmanned planes, flying at constant speed. The position vector p of agent i at time t is :

$$p_i(t) = \begin{pmatrix} p_{iw}(t) \\ p_{ih}(t) \end{pmatrix}$$

and is defined on $[0; W[\times[0; H[$.

At each time step, only the agent's *direction vector*, $v_i(t)$, can change and is defined as a unit vector:

$$v_i(t) = \begin{pmatrix} v_{iw}(t) \\ v_{ih}(t) \end{pmatrix}$$

The complete definition of $v_i(t)$ is given by equation 6.

Accordingly, at each time step the new agent's position is given by :

$$p_i(t+1) = p_i(t) + v_0.v_i(t)$$

Agents' behaviour is inspired by the work of Reynolds (1987) on flocking behaviours which is based on the combination of simple rules. In our case, agents' direction $v_i(t)$ is thus computed from a linear combination of several $v_{ik}(t)$, each determined according to a simple rule. We used a weighted average to compute $v_i(t)$ according to the $v_{ik}(t)$ The weights are denoted α_k and are the same for all agents. The following equation define $v_i(t)$:

$$v_i(t) = normalize\left(\sum_{k \in S_r} \alpha_k.v_{ik}(t) \right) \tag{6}$$

where S_r is the set of used rules. A particular MAS behaviour being a combination of specific rules among all the developed rules, with associated weights.

We supposed that agents can move through burning cells, we can consider that they are either immune to fire or flying over the environment. We made this hypothesis to avoid introducing behaviours specific to the fire front propagation use case.

Agents have a circular field of view with radius r_a. Only cells inside the circle $(p_i(t), r_a)$ are sensed by agent i at the time t. We also define the measure operator C_i of agent i as a diagonal matrix. Diagonal elements relative to a measured cell are equal to 1, 0 otherwise. We then define C as the measure operator of the MAS as a whole:

$$C(t) = \bigcup_{i=1}^{N_a} C_i(t) \tag{7}$$

where N_a is the number of agents and $\bigcup$ the inclusive OR operator applied on each components of each $C_i(t)$.

Finally, we define the global measure as :

$$Y(t) = C(t) \cdot X(t) \tag{8}$$

In the remaining of the text, we will use C for $C(t)$ and C_i for $C_i(t)$.

In the following, we will define four different MASs, two used as references, and two other, used to study the ability of a MAS to be used as a distributed estimator of a 2D dynamical system. The MAS types are the following:

- **Reference agents**: these agents have a random behaviour, their moving direction is smoothly and randomly varied over time. These agents have no memory ability and do not take their perception of the environment into account.
- **Reference agents (zoned)**: these agents are similar to previous one except that they can only move in restricted sub-division (zones) of the environment.
- **Reactive agents**: these agents' direction is based solely on their instantaneous perceptions, they do not have a model of their environment and do not exchange information with each other. The observer constituted by these agents is said *static* because the information they add to the estimator is not automatically updated over time by any predicting rules.
- **Cognitive agents**: the behaviour of these agents depends on their current perceptions, on an internal, dynamic model of their environment, and on the information they exchange with each other. The observer they constitute is said *dynamic* because they update the information they put in the estimator according to the state of their model even without direct perception of the environment.

4.2 Random agents

To get a meaningful reference behaviour that can be used to compare the estimation quality of different MAS behaviours we used two kinds of random, exploratory behaviours:

- All agents move randomly over the entire environment. This behaviour is denoted as random.
- All agent moves randomly but their moves are restricted to a particular subdivision of the environment. This simple division of space can be considered as a predefined global, centralized coordination strategy. While it is trivial to implement in the chosen simulation environment, it is important to note that not all environments may be easily split between agents. Furthermore, this strategy has the drawbacks of centralized approaches such as making the system vulnerable to one agent failure. This behaviour is denoted as random-zoned.

These two behaviours are obtained by combining two direction-changing rules: *inertia* and *randomness*. Those rules are described in the next two subsections.

4.2.1 Inertia rule

This rule represents the influence of the agent's moment of inertia on its direction change. The overall agent's trajectory is strongly influenced by this rule since it determine the maximum curvature radius of their trajectory. The weight of this rule in the combination (*cf.* eq.6) which determines the agent's direction at the next time step is denoted α_1. When α_1 is small, agent's direction can change easily and its trajectory can get more curled. This kind of behaviour can be implemented by quad-copters drones for example. Conversely when α_1 is large, agent's direction tend to stay constant, its trajectory is thus more smooth. This kind of behaviour looks more like the one of a plane.

The contribution of this rule is simply the direction (unit) vector of the agent at the previous time step:

$$v_{i1}(t) = v_i(t-1)$$

4.2.2 Random rule

This rule gives a uniform random direction vector:

$$v_{i2}(t) = \begin{pmatrix} \cos(\theta_r(t)) \\ \sin(\theta_r(t)) \end{pmatrix}$$

where θ_r is a uniform random angle within $[0; 2\pi[$.

The "*random*" reference behaviour results from the linear combination of this two rules using the definition of equation 6 with only rules 1 and 2 in S_r. The resulting behaviour is a smooth random walk of the agents in the environment.

We used the Nelder Mead method to explore the parameter space and to find appropriate values for the weight of these rules (α_1 and α_2) in the combination. This optimisation method is described in Section 4.6.

Now that we have defined the rules at play in the reference behaviours, we will present the way we compute the static estimate $\hat{X}$ from the agents' perceptions.

4.3 Static distributed observer

To quantify the ability of the MAS to build a correct estimation, we chose to use the global measure Y defined by equation 8. The MAS built estimate is thus an aggregation of the instantaneous measures made by the agents. While this kind of estimation is hardly possible to obtain in real conditions due to communication and synchronisation problems, we can use it because all agents are simulated. The agents do not have access to this estimate because it would have compromised the decentralized property of our approach.

The formal definition of the cellular automata observer we defined is as follow:

$$\begin{cases} \hat{X}^+ = A_{os}(\hat{X},\ Y) \\ Y \quad = C \cdot X \end{cases}$$

Since the information used to build the state estimation $\hat{X}$ is gathered from direct perception and without any information about the dynamic of the system, the observer constituted by the agents is said *static* in automatic control terminology. Furthermore, the global estimate $\hat{X}$ is updated with agents' perceptions only at currently perceived cells. Estimate of all unperceived cells is not changed. In this case, the observer evolution function A_{os} can be simplified. We obtain the following expression for $\hat{X}^+$:

$$\begin{cases} \hat{X}^+ = (I - C).\hat{X} + Y = (I - C).\hat{X} + (C).X \\ Y \quad = C \cdot X \end{cases}$$

where I the identity matrix with adequate dimensions.

From the definition of C (equations 7), we see that according to the values of its elements (0 or 1), we select either an element of X or an element of $\hat{X}$ to compute elements of $\hat{X}^+$.

This global estimation is used as a reference to compare the other, more sophisticated, MAS behaviours. The next section describes the first of these behaviour, implemented with reactive agents.

4.4 Reactive agents

Unlike random agents, reactive ones use their perceptions to determine their direction. They are called reactive because their moves are only determined by their perceptions and not on an internal model of their environment. As for the random agents and unlike the random-zoned ones, there is no predefined global coordination strategy among the reactive agents. The global estimate $\hat{X}$ is built in the exact same way as exposed in the previous section.

We defined 3 rules for this type of agents. The first one is the inertia rule defined on the part 4.2.1. The second one allows the agents to move according to the perception of the environment's limits. The last rule modifies the direction of the agents according to the proximity of the other agents.

4.4.1 Boundary avoidance rule (α_3)

This rule makes agents move away from the environment's boundaries. This repulsion is inversely proportional to the distance of the agent to the boundary. The formal definition of the contribution of this rule to the agent's direction is:

$$v_{i3}(t) = \begin{pmatrix} \frac{1}{p_{iw}(t)} \\ \frac{1}{p_{ih}(t)} \end{pmatrix} - \begin{pmatrix} \frac{1}{W - p_{iw}(t)} \\ \frac{1}{H - p_{ih}(t)} \end{pmatrix}$$

The weight of this rule in the rule-combining equation is α_3

Figure 3 shows the agents' trajectories with inertia rule's weight $\alpha_1 = 1$ and varying boundary avoidance rule's weight α_3. We used $N_a = 20$, $W = 512$, $H = 256$, $V_0 = 2$.

4.4.2 Agents avoidance rule (α_4)

This rule makes agents avoid each other in the same way charged particles repel each others. The closer the distance, the stronger the repelling force. Formally, the contribution of this rule to the agents' direction vector is defined as:

$$v_{i4}(t) = \sum_{j=0; j \neq i}^{N_a - 1} \frac{p_i(t) - p_j(t)}{\|p_i(t) - p_j(t)\|^3}$$

The intuitive idea behind this rule is to increase the average distance between agents and to distribute them evenly on the environment. The figure 5 shows this rule's influence on the average distance between agents. When this rule is disabled ($\alpha_4 = 0$), we measure an average distance between agents equal to 85 cells with: $N_a = 10$, $W = 512$, $H = 256$, $V_0 = 2$, $\alpha_1 = 1$. We got the results of figure 5 with the same conditions.

It is clear from Figure 5 that there is a positive correlation between the weight of the agents avoidance rule and the average distance between agents. We can also observe that when α_4 is large enough ($\alpha_4 > 10^4$), a saturation of the average distance between agents appears. The figure 4 shows such configuration (with $\alpha_4 = 10^5$). Agents' positions are stable and depend only on the environment's dimensions ratio (1/2 in this case) and the number of agents ($N_a = 20$). This configuration is probably the best one to maximize the average distance in these conditions.

Note that we could have formulated the combination of these rules as a sum of forces applied to the agents, as in classical mechanics. However this would have create trajectory dynamics and associated trajectory tracking control problems. In this work, we rather considered an ideal control for the drones which make feasable instantaneous changes in speed and direction.

As for the random behaviours, we used the Nelder Mead method to find suitable values of the different weights of the three rules defining the reactive agents' behaviour. The way we build the global estimate $\hat{X}$

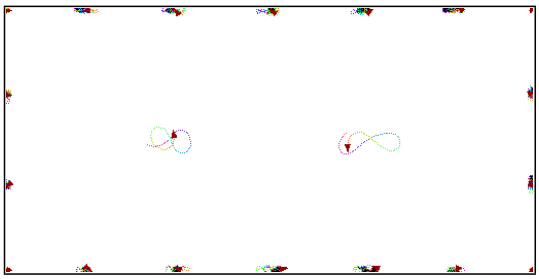

Fig. 4: Capture of a simulation with a extreme influence of the agents avoidance rule.

for the reactive agents is the same as for the reference behaviour which is exposed in Section 4.3.

4.5 Cognitive Agents – Dynamic Observer

Agents are said cognitive when they can build, update, and use an internal representation of their environment. The agents we are presenting in this section are not purely cognitive, their decision is based on their internal model and on their current perception. We gave them the ability to store their perception in an internal memory, to make prediction on the evolution of their environment, and to exchange information. Cognitive agents described in this section have three more abilities than the reactive ones:

- Each agent i has its own global estimation of the system state, $\hat{X}_i$.
- Agents can send data to agents that are within their communication range.
- They know the evolution laws of the environment, so they can predict its evolution.

These abilities are detailed in the next sections.

4.5.1 Internal state estimation

Contrary to reactive agents which share a write-only, global estimation $\hat{X}$, each cognitive agent has an internal one, $\hat{X}_i$, which it can read and modify. So each agent store an estimate of the system's state for all the cells of the environment. Since the estimates are not shared, they can differ for a given cell.

In addition to the estimated state, each cell c in the internal model of agent i has an associated confidence value $h_i(c,t)$ within $[0;1]$. As a convention, a confidence of 1 means that the cell's state is perfectly known. Conversely, a confidence of 0 means that the agent has no information about the cell. Typically, a cell that is currently being perceived has an associated confidence of 1. Cells that have never been observed have a confidence of 0.

At each simulation iteration, cognitive agents behaviour has two steps: 1) A perception step and 2) a transmission step.

In the perception step, agents update their internal estimate $\hat{X}_i$ according to their immediate perceptions. So for each perceived cell, its estimate value is overwritten by the perceived one and the confidence value is set to 1.

During the transmission step, agents that are within communication range r_c will share their estimates for all the cells. An agent will update its estimate about a cell according to the value given by another agent only if the other agent has a greater confidence in its estimate about this cell. When updating its estimate value about a cell according to another agent's value, an agent will set the confidence of this cell to 90% of the transmitted value. As an example, considering the situation where agent i considers cell c_{42} as burning with a confidence of 0.2 and that agent j considers this cell as hashes with a confidence of 0.75. If they come close enough to each other, they will share their estimation of all cells. Since agent j has a greater confidence in its estimate about cell c_{42} $(0.75 > 0.2)$, agent i will set its estimate of this cell to "hash" and associate a confidence value of 0.675 $(0.75 \cdot 90\%)$. In this example, after the transmission step, nothing will have change about agent j estimate of cell c_{42}, and agent i will have update its estimate and confidence value about this cell. This updating process is executed on all environment cells.

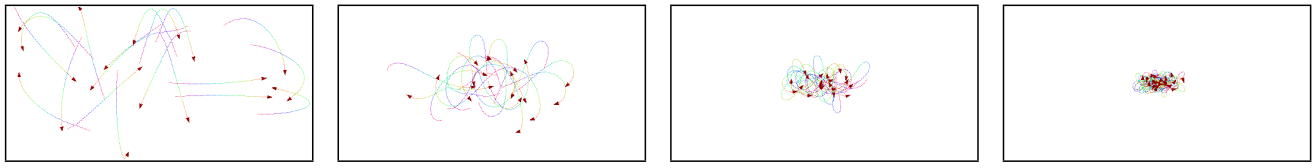

Fig. 3: Influence of boundary avoidance weight, α_3, on agents' trajectories. Triangles represent agents and curves represent trajectories. From left to right, values of α_3 are: 1, 16, 64, and 256

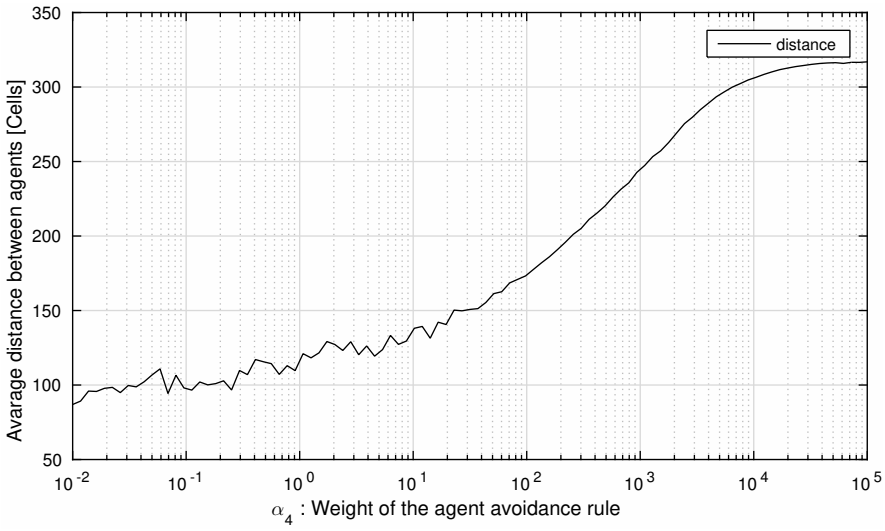

Fig. 5: Influence of the agents avoidance rule on the average distance between agents.

4.5.2 Dynamic observer

The third ability that cognitive agents have over the reactive ones is to know the evolution laws of the system they observe. This ability allows them to predict future state of the system and thus to update their estimation at places they are not observing. In the work presented here, we made the hypothesis that agents have a perfect knowledge of the dynamic of the system. They can use the CA that is updating the environment to update their internal estimate $\hat{X}_i$.

Since there is no global shared estimate $\hat{X}$ that we can use anymore to compute the estimation quality q defined in equation 5, we have to define one from the $\hat{X}_i$. As these estimates vary according to a model of the evolution laws of the environment, the observer constituted by the agents is said *dynamic*.

The formal definition of how agent i internal estimate $\hat{X}_i$ is updated is:

$$\begin{cases} \hat{X}_i^+ = A_{od}(\hat{X}_i,\ Y_i) \\ Y_i \ \ = C_i \cdot X \end{cases}$$

To compute the next state of a cell's estimate, the same rules are applied as those used for the fire front CA. These rules are represented by the matrix A defined in Section 2. These rules are applied on the internal state estimate $\hat{X}_i$, on each cell, except when a direct or indirect measure is available. Finally, the expression of the dynamic estimator is:

$$\begin{cases} \hat{X}_i^+ = A_{od}(\hat{X}_i,\ Y_i) = A((I - C_i').\hat{X}_i + Y_i) \\ Y_i \ \ = C_i' \cdot X \end{cases}$$

where C_i' is the direct and indirect measure operator.

In addition to update $\hat{X}_i$ at each time step, cognitive agents update the confidence value of their estimate's cells. We made the following three hypothesis about the confidence evolution:

1. If there is no new information about a cell (direct observation or values from other agents), its associated confidence value will slowly decrease, this is the **forgetting factor**.
2. The confidence associated with a cell decreases rapidly if there are adjacent cells with low confidence because of the uncertainty on these cells' state. This is the **uncertainty factor**.
3. Conversely, the confidence of a cell can grow slowly if its adjacent cells have a high confidence value. This correspond to the idea that if an agent knows one cell with certitude, it will be able to correctly predict the value of the adjacent cells. This is the **certainty factor**.

These hypothesis about the confidence evolution can be modeled using the principles of heat diffusion and dissipation on a surface. Confidence being equivalent to heat, dissipation correspond to the forgetting factor, and diffusion the other two factors. The heat propagation and dissipation equation in continuous time is:

$$\frac{\partial h(c,t)}{\partial t} = k_1 \cdot \left(\frac{\partial^2 h(c,t)}{\partial {c_w}^2} + \frac{\partial^2 h(c,t)}{\partial {c_h}^2} \right) - k_2 \cdot h(c,t) \quad (9)$$

where k_1 is the diffusion coefficient, k_2 the dissipation one, and c_w, c_h the components of the cell's position vector c.

After choosing appropriate values for k_1 and k_2 and transposing the previous equation in discrete time and

space, we optain:

$$h(c,t+1) = k_1 \cdot \Big(h(c_{up},t) + h(c_{left},t) + h(c_{down},t) + h(c_{right},t) - 4 \cdot h(c,t)\Big) - k_2 \cdot h(c,t) \quad (10)$$

where $k_1 = 0.2499$, $k_2 = 0.0004$, and c_{up}, c_{left}, c_{down} and c_{right} are the upper, left, down, and right adjacent cells of c.

The following constraints were taken into account while choosing k_1 and k_2 values. To ensure that the equation 10 is numerically stable, it is necessary to have $0 \leq 4 \cdot k_1 + k_2 \leq 1$. Also, in order to get the behaviour associated with the *forgetting factor*, k_2 must be greater than 0. Finally, we empirically chose the greatest value for k_1 that made cells' certainty behave coherently with the *uncertainty* and *certainty* factors.

For cell on the environment boundaries, where equation 10 is not defined[4], we set their state at the value of the closest cell for which the equation is applicable. In this case, we consider that $\frac{\partial h(c,t)}{\partial c} = 0$.

In Section 3 we exposed a quantitative measure of the difference $D(t)$ between a state X and an estimation $\hat{X}$ of this state (equation 3). In the dynamic estimator case, $\hat{X}$ is not directly available, unlike in the reference and reactive agents cases. We need to define the way $\hat{X}$ is determined from the internal $\hat{X}_i$ of the cognitive agents.

For each cell in $\hat{X}$ we take the value of the $\hat{X}_i$ that has the highest confidence value. This global estimate is still a ubiquitous tool, used for evaluation purposes only, and not accessible to the agents, to preserve the decentralized property of our approach. Using this global estimate, we can compute the estimation quality criterion q defined by equation 5 in Section 3.3. In both static and dynamic estimator cases, we build the global estimate $\hat{X}$ at each time step.

The next section explains how cognitive agents use their internal $\hat{X}_i$ to improve their trajectories in order to maximize the quality of their observations.

4.5.3 Confidence improving rule (α_5)

The behaviour of the cognitive agents combines the three rules of the reactive agents (inertia, boundary avoidance, and agents avoidance) with a new one: the confidence improving rule. The aim of this rule is to lead the agent toward the part of the environment where its confidence in its estimation is the lowest. Three implementations of this rule have been considered, two taking global information from $\hat{X}_i$ into account and one taking only local information into account:

1. The first global approach based on $\hat{X}_i$ is to compute the overall cells positions' barycentre, weighted by 1 minus the confidence value. We obtain the position where the confidence is the lowest on average. The direction of the vector $v_{i5}(t)$ corresponding to this rule is toward this point.
2. Another global approach is to make the agent go toward the cell with the lowest confidence. Each time the agent reaches this point, it sets the cell's confidence to 1 and move on to the next cell with the lowest confidence.
3. The local approach we considered is to identify the direction around the agents where the confidence is the lowest. This approach is similar to the first one but the barycentre is computed only on the cells within some neighbourhood of the agent.

We only implemented the third solution for the following reasons: the second solution is problematic when there are several cells with the minimum confidence value (which is frequent, especially at the beginning of the simulation, when almost all cells have not been measured and have a confidence of 0). The second solution is also problematic since the barycentre of the low confidence cells may be a very well explored area. For example, if the agent is at the center of the rectangular environment with all cells' confidence values being 0 ($h_i(c,t) = 0$ for all cells except the perceived ones). In this case, the average position weighed by $(1 - h_i(c,t))$ is the current agent's position. Let's suppose that the agent moves to the left. It will then perceive cells on its left, making the average uncertain position moves to the right. This will lead the agent to the right, then to the left, *etc.* A not very interesting, oscillating over already visited cells, exploratory behaviour.

The third solution allows the agent to avoid such situations. The most uncertain cell in average is looked for within the neighbourhood of the agent. So it will always go forward, because cells on its back will always have a high confidence value since they have just been perceived. The formal expression of the contribution $v_{i5}(t)$ of this third implementation of the confidence improvement rule on the agent direction is:

$$v_{i5}(t) = \left(\frac{1}{|R_{5i}(t)|} \sum_{c \in R_{5i}(t)} (1 - h_i(c,t)) \cdot c\right) - p_i(t) \quad (11)$$

where $R_{5i}(t) = \{c \in L \mid r_a < \|c - p_i(t)\| < 2 \cdot r_a\}$: the cell set on which the barycentre is computed. This set is a ring around the agent. The ring's smaller radius r_a is equal to the agent's field of view. We chose this value

[4] Cells on the left boundary have no left adjacent cell for example.

for the minimal radius because cells within the agent's field of view are directly perceived and have therefore a confidence value of 1, making them useless to include in a minimum finding process. The ring's larger radius has been arbitrarily set to $2r_a$. The weight $(1-h_i(c,t))$ of each cell is proportional to the uncertainty associated with it. So $v_{i5}(t)$ points toward the point of lowest confidence within the agent's neighbourhood. As for the other rules, we define α_5 as the weight used in the weighted average which determines the agent's direction.

The following section presents the method we used to find values of α_k which maximize our estimation quality criterion q.

4.6 Rules weights optimization (Nelder-Mead)

In order to find values for the rules' weights α_k that minimize the criterion q, we used the Nelder-Mead optimization method. It is a generic numeric method which minimize a scalar continuous function defined on a multidimensional space.

For example, in the case of the cognitive agents, we used this method to find the values of α_3, α_4 and α_5 (α_1 being constant and set to 1) that give the best estimation quality. In this case the parameter space's dimension is three.

The Nelder-Mead method is also called "Downhill Simplex Method". Indeed, it makes use of simplexes which are a generalization of triangles in any dimension. In a space of dimension 1, a simplex is a segment, in a 2-dimensional space, it's a triangle, in a 3-dimensional space, a tetrahedron. More generally, in an N-dimensional space, a simplex has $N+1$ vertices.

The simplex used by the Nelder-Mead method is defined in the parameter space. Each of its vertex represent a specific set of values for the α_k. Once an initial simplex chosen, each vertex of it is associated with the value of the criterion q obtained after several simulation carried out with the corresponding α_k. Then, the Nelder-Mead method gives rules to move the simplex's vertices according to their associated values in order to make the simplex converge toward a local minimum. In our case, this minimum is a configuration of rules combination's weights that maximize the quality of the estimate built by the agents.

Figure 6 shows an execution of this algorithm on an example 2-dimensional parabolic function f centred in $(0.9; 0.6)$. We note $s = \begin{pmatrix}\alpha_1 & \alpha_2\end{pmatrix}^T$, the 2D input of this function which is defined by:

$$f(s) = \sqrt{(2\alpha_1 - 1,8)^2 + (2\alpha_2 - 1,2)^2}$$

Each step of this algorithm requires several simulations with varying initial condition to get a significant value for q. Furthermore the complexity of the parameter space's exploration increases dramatically with the number of its dimensions. Consequently, we have to keep the number of weights, and thus the number of rules, low to be able to find interesting values for the α_k. We chose this empirical method to evaluate the quality of the estimation because the analytical relationship between the rules weights and q is extremely complex to establish.

5 Results and discussions

This section presents two analyses. First, we study the effect of the number of agent on the global behaviour. For each kind of agents (random-zoned, random, reactive, and cognitive), we made experiments for 1, 4, 16 and 64 agents and studied the evolution of our global criteria q. Second, we analyse the evolution of the estimation quality over the duration of a simulation in a 4-agent configuration. Table 1 presents the initial condition and parameters values used in the results presented here.

5.1 Influence of the number of agents

This section presents our analysis of the effect of the number of agents on the estimation quality criterion q defined by equation 5 in Section 3.3, for each behaviour we defined.

We did simulations with 1, 4, 16, and 64 agents for each behaviour (random, random-zoned, reactive, and cognitive). For each agent number-behaviour combination, we did eight simulations with different initial conditions to get more robust results. The figure 7 synthesizes these results using a logarithmic scale. Using this logarithmic scale for the x and y axis allowed us to detect a linearity between the agents count and the q criteria when the number of agents is lower or equal to 16. This case correspond to a low density of agents in the environment and is more realistic than when this density is too high. In our case, we fixed the limit at sixteen agents. When agent density is too high, they perceive almost all the environment at each time step and their behaviour has less influence than the range of their field of view.

5.1.1 Low agent density

According to the observed linearity on the log-log graph, and in order to find the relationship between the crite-

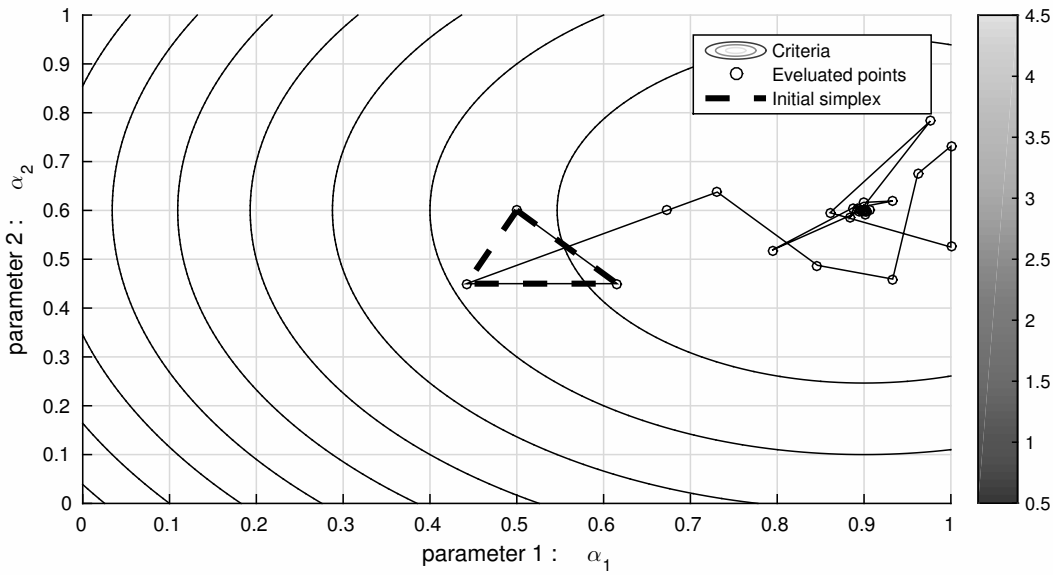

Fig. 6: Execution of the Nelder-Mead algorithm to find the minimum of an example function.

Parameter	**Symbol**	**Value**
Iteration count	t_f	~ 3700
Environment width	W	256
Environment height	H	256
Field of view (cells)	r_a	9
Communication range (cells)	r_c	64
Speed of agents (cells/iteration)	v_0	2
Initial position of agents	$p_i(0)$	$(26; 26)$
Initial direction of agents	$v_i(0)$	$\begin{pmatrix}\cos(2\pi i/N_a)\\ \sin(2\pi i/N_a))\end{pmatrix}$

Table 1: Initial condition and parameters

rion q and the number of agents N_a when it is adequate (not too high), we used the following expression:

$$\log_2(q) = k_1 \cdot \log_2(N_a) + k_2 \tag{12}$$

where k_1 and k_2 are the coeficients of the linear relationship we want to identify.

We used the least squares method to find the k_1 and k_2 that made the previous linear relationship match our experimental points for 1, 4, and 16 agents.

To obtain a direct relationship between q and N_a, without the log, we can transform equation 12 into:

$$q^{-1} = k_3 \cdot N_a^{k_4} \tag{13}$$

where $k_3 = 2^{-k_2}$ and $k_4 = -k_2$. We introduced the inverse of q because a high value of it indicates a low estimation quality.

As the right side of equation 13 is the product of k_3 and $N_a^{k_4}$, we can interpret k_3 as the absolute performance of a behaviour because it is independent from N_a and k_4 can be seen as the improvement brought by each additional agent. The values of k_3 and k_4 we computed for each MAS behaviour are presented in Table 2.

Agent type	**Approximation of q**
Random-zoned	$q^{-1} = 1.27 \cdot {N_a}^{0.59}$
Random	$q^{-1} = 1.31 \cdot {N_a}^{0.57}$
Reactive	$q^{-1} = 1.52 \cdot {N_a}^{0.71}$
Cognitive	$q^{-1} = 2.39 \cdot {N_a}^{0.90}$

Table 2: Relationship between N_a and q when $N_a < 16$

Overall, it is important to note that: 1) Cognitive agents are basically better than those using a static estimator : k_3 is the highest for cognitive agents. 2) Each additional cognitive agent brings more performance to the observer than other types of agents. These observations are in line with our hypotheses and confirm the value of adding cognitives abilities to agents.

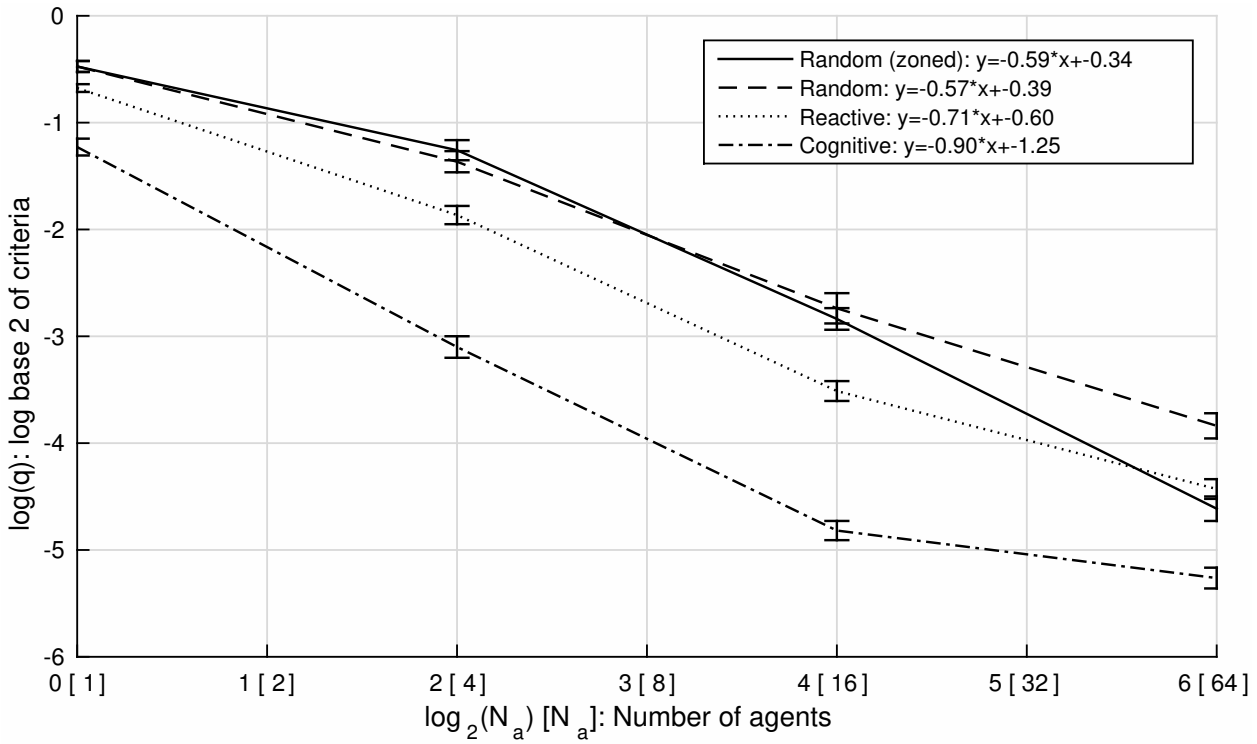

Fig. 7: Relationship between the number of agents and the estimation quality criteria for different MAS behaviours.

Regarding the agents using the static estimator, we observe that the reactive ones are more efficient than random ones. This result was expected because they use avoidance rules (boundary and agents) and confirms that a behaviour based on direct perceptions gives better performances than simple random walks.

5.1.2 High agent density

In our simulation condition, when there are more than sixteen agents, it becomes difficult to draw conclusions. It is clear that when the agent density is sufficiently high, the estimator becomes perfect since the agents are then able to perceive the entire environment at each time step. Figure 7 seems to support this hypothesis: the criteria values are getting closer and seems to converge toward a horizontal asymptote. Additional experiments would be required to support this intuitive observation.

5.2 Estimation error evolution over the duration of a simulation

This section presents our observations about the evolution of the difference between the MAS-built estimate and the CA-simulated state, $D(t)$, defined by equation 3, over the duration of a simulation. The evolution of a fire front propagation is monotonous: the environment changes slowly at the beginning, when there are only a few burning cells, the change rate increases dramatically when the fire front becomes large and fall toward zero when only hashes remain. Our simulations stops when there is no changes anymore in the cells' state. The criteria q we studied in the previous sections does not account for the evolution of the difference between the collective estimate and the simulated state over the simulation duration since it is based on a cumulative sum of this error over the whole duration of the simulation.

To get insight into the different dynamics of the error along the simulation duration, we carried out twelve simulations with different forest compositions (other initial parameters staying equal) for each kind of agent behaviour. There were four agents in these simulations. Figure 8 presents the average value of $D(t)$ over the twelve simulations for each behaviour. "Burned cells" curve shows the ratio of burning cells relative to the total number of cells.

The most interesting observation is made considering iterations between 1000 and 2000. During this time interval, the error of all behaviours but the cognitive one show a significant increase. This increase is due to the fast modification of the environment. It is indeed during this time lapse that there is the greatest number

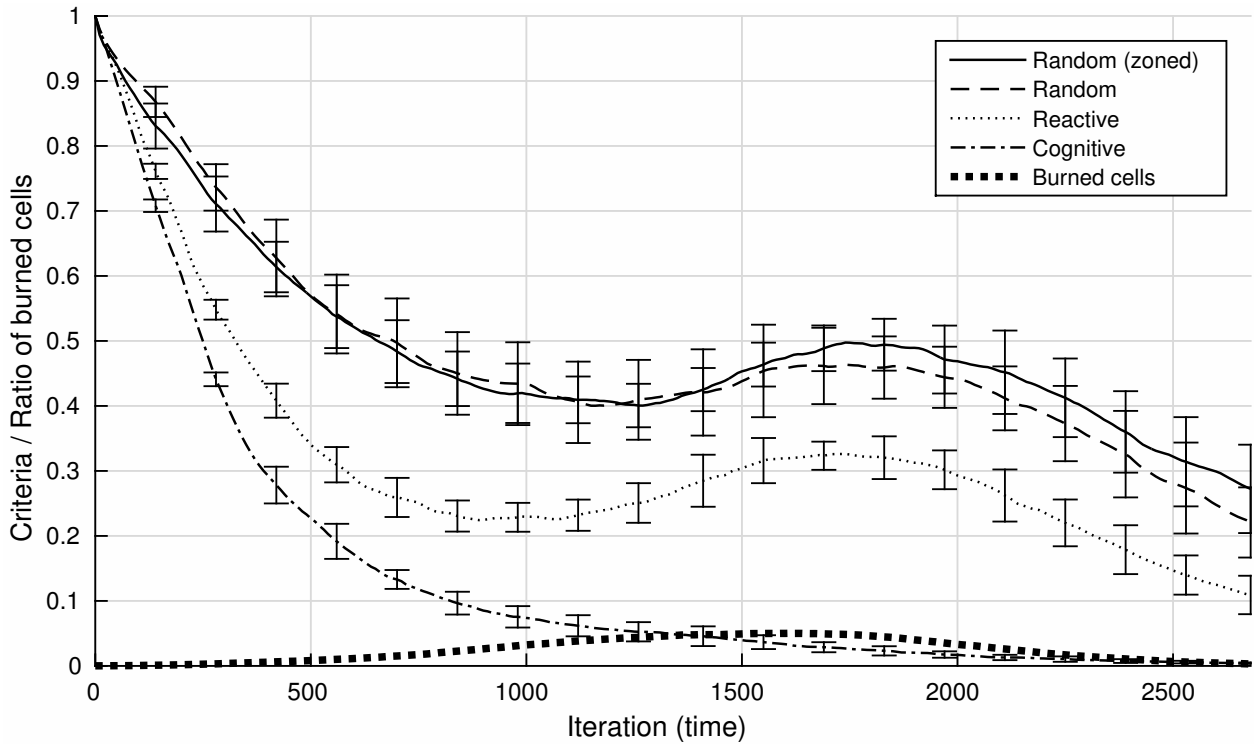

Fig. 8: Average value of $D(t)$ over twelve simulations for each MAS behaviour against the ratio of burning cells to the total number of cells N_c.

of burning cells which quickly turn forest into hashes. Conversely, the estimation quality of cognitive agents is always improving. This property is extremely interesting since the improvement brought by cognitive agents regarding the overall estimation quality q is magnified if we only consider the time lapse during which the environment is changing the most. This time lapse being the one during which it is the most interesting to get a precise estimate. The constant decrease in the difference between the simulated state and the estimate built by the cognitive agents indicates that they are quite resilient to the change rate of the environment.

Another interesting observation can be made on the first two hundred time steps. During this interval, there is still only a few burning cells, the environment is almost not changing. So the main reason of the increase in estimation quality comes from the initial exploration of the environment by the MAS, and not from the correction of estimation mistakes. This gives us information about the rate at which the various behaviours explore an unknown environment. Table 3 presents the exploration rate of each behaviour during these first time steps.

While these results depend on the size of the environment, we can still draw some partial conclusions. Cognitive agents have the best exploration rate. This is not surprising since they use information about the direction where their estimate is the worse, leading them toward unexplored areas. Furthermore, as they share information about explored areas, they tend to avoid them.

Agent Type	**Exploration rate**
Random zoned	0.11%
Random	0.09%
Reactive	0.17%
Cognitive	0.20%

Table 3: Exploration rate, expressed in environment portion per time step, of the different MAS behaviours

Reactive and cognitive agents have the ability to avoid each other. This ability prevent them to simultaneously explore the same area. That's why they have better exploration rates than the random agents.

As expected, random behaviours have lower exploration rates than the others because they don't use any information to improve their ability to explore their environment. The two kind of random behaviours have similar exploration rates. We can suppose that this is because this rate depend only on the range of their field of view.

All these results support the hypothesis that knowledge and communication improve the ability of a MAS to build a precise estimate of a dynamical system.

5.3 Conclusion and perspectives

This paper describes tools to evaluate the abilities of multi-agent systems with different coordination mechanisms to build an estimation of the state of a simulated 2D environment modeling the propagation of a forest fire front. Those MASs have to coordinate the trajectories of mobile sensors in order to maximize the quality of the collectively built estimation. To do so, we used a completely decentralized approach to the problem of dynamic complex system observation and that's what distinguishes our work from existing approaches which either use centralized methods or examine static systems.

Each different MAS behaviour is based on a particular combination of elementary behaviours. The first one models the agents' inertia and ensures smooth trajectories. The second and third behaviours use agents' perceptions to make them avoid the environment boundaries and each other, respectively. The last one utilizes cognitive abilities added to the agents: communicating with each other, building an internal model of their environment, predict the evolution of this model.

To evaluate and compare the different behaviours resulting from the possible combinations, we defined a distance measure D between the CA-simulated state of the forest fire front and the estimate collectively built by the agents. We used this distance in two ways: first we defined a criteria q which is the average value of this distance over the whole duration of a simulation; second we studied the variation of this distance throughout the duration of a simulation. The q criteria allowed us to find appropriate values for the different parameters of agents' behaviours and to quantify the ability of these behaviours to build a correct estimate of the observed state. The experiments we carried out support our hypotheses according to which taking more information into account in the agents' behaviour improves the quality of their estimation. Indeed, reactive agents, which use only their perceptions to choose their direction, were able to build better estimate than agents with random walk-types of behaviour. Furthermore, cognitive agents are even more efficient than reactive one.

When we studied the variation of D throughout the duration of a simulation, we found that cognitive agents were far less sensible to the change rate of the environment than the reactive and random agents. Indeed they were able to improve their estimation of the system state during the whole simulation, even when the environment was changing quickly, which is not the case of the other kinds of agents. All experiments were carried out using GPGPU, allowing us to carry out many simulations with varying initial condition and behaviour. On an Intel i5 CPU ans NVidia 830M GPU, a simulation of ~ 3000 steps with with 8 agents in a 100000 cells environment takes ~ 7 seconds for random and reactive agents and takes ~ 13 minutes for cognitive ones.

The main research avenue we plan to explore is to study the effect of some parameters on the estimation efficiency. For example the influence of the range of agents' field of view, or the link between their communication range and their avoidance behaviour, seem to be interesting factor to study to get more insight into the relationships between agents' behaviour and their ability to build collectively an estimation of the state of a dynamical system. Other future works includes the validation of the observed properties in more complex (*e.g.* periodic) systems such as chemical reactions, or population simulations.

References

Barbieri B, Sartoretti G, Falcone JL, Chopard B, Gander M (2013) Traffic Prediction Based on a Local Exchange of Information. Journal of Cellular Automata 8:429–441

Chopard B (2012) Cellular automata modeling of physical systems. Computational Complexity: Theory, Techniques, and Applications pp 407–433, DOI 10.1007/978-1-4614-1800-9{_}27

El Yacoubi S, El Jai A (2002) Cellular automata modelling and spreadability. Mathematical and Computer Modelling 36(9-10):1059–1074, DOI 10.1016/S0895-7177(02)00259-5, URL `http://www.sciencedirect.com/science/article/pii/S0895717702002595`

Ferber J (1999) Multi-agent systems: an introduction to distributed artificial intelligence, vol 1. Addison-Wesley Reading

Franklin S, Graesser A (1996) Is it an agent, or just a program?: A taxonomy for autonomous agents. In: International Workshop on Agent Theories, Architectures, and Languages, Springer, pp 21–35

Green DG (1989) Simulated effects of fire, dispersal and spatial pattern on competition within forest mosaics. Vegetatio 82(2):139–153, DOI 10.1007/BF00045027

LAVILLE G (2013) Thèse de Doctorat - Exécution efficace de systèmes multi-agents sur GPU

Liu W, Gao WC, Sun Y, Xu MJ (2008) Optimal sensor placement for spatial lattice structure based on

genetic algorithms. Journal of Sound and Vibration 317:175–189, DOI 10.1016/j.jsv.2008.03.026

Parberry I (2014) Designer Worlds: Procedural Generation of Infinite Terrain from Real-World Elevation Data. Journal of Computer Graphics Techniques (JCGT) 3(1):74–85

Reynolds CW (1987) Flocks, herds, and schools: A distributed behavioral model. In: Computer Graphics, SIGGRAPH '87 Conference Proceedings, vol 4, pp 25–34

Sartoretti G, Falcone JL, Chopard B, Gander MJ (2012) Decentralized method for traffic monitoring. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7495 LNCS:464–473, DOI 10.1007/978-3-642-33350-7-48

Schlotterbeck G, Raïevsky C, Lefèvre L (2016) Decentralized estimation of forest fire spread using mobile sensors. In: International Conference on Cellular Automata, Springer, pp 334–343

Waldraff W, Dochain D, Bourrel S, Magnus A (1998) On the use of observability measures for sensor location in tubular reactor. Journal of Process Control 5(6):497–505

Yacoubi SE (2008) A mathematical method for control problems on cellular automata models. International Journal of Systems Science 39(5):529–538, DOI 10.1080/00207720701847232