Complex flocking dynamics without global stimulus

Emmanuel Hermellin[1] and Fabien Michel[1]

[1]LIRMM - University of Montpellier, 161 rue Ada, 34095 Montpellier Cedex France
emmanuel.hermellin@lirmm.fr

Abstract

Murmuration, i.e. starlings gathering and swirling with extraordinary spatial coherence, is one of the most impressive kind of bird flocking. It is now well accepted that this collective behavior emerges from individual ones and that no global control is involved. In other words, every starling has an equivalent status in the flock and there is no leader deciding how the murmuration evolves. Considering this phenomenon, Reynolds' individual-based rules have been investigated and implemented a number of times to create compelling computer-animated models of the aerial movement of swarm-like flocks of starlings. Reynolds' model is considered as a classic Agent Based Model (ABM) and integrated as a flagship example in many ABM platforms.

Still, it turns out that implementing Reynolds' model is not sufficient per se in the sense that all murmuration simulations use tricks to achieve a convincing animation of this phenomenon. Especially, virtual leaders or points of interest are used to orientate the starlings, which somehow contradicts the no-global-control perspective, and thus suggests that murmuration dynamics is not yet fully grasped. This paper first highlight this aspect of existing murmuration simulations and then show that it is possible to obtain murmuration-like dynamics by only rethinking how Reynolds' are usually implemented. Especially, the proposed model does not require the existence of a virtual leader nor embed any global aspect.

The objective of this article is to show that it is possible to obtain complex coordinated flight dynamics using a very simple ABM and without adding external stimulus nor additional features, that is by only implementing Reynolds's rules thanks to the IRM4S modeling perspective (an Influence Reaction Model for Simulation). So, in this article, we will first focus on the existing implementations of flocking model to list the advantages and limits and then propose our solution based on the IRM4S approach.

Introduction

Grouping behavior is widely found in nature: Birds fly in flocks (Potts, 1984), fish swim in schools (Shaw, 1975), sheep move as a herd steered by a dog (Gueron et al., 1996) and insects swarm in an exhibit disorganized movements (Okubo, 1986). While engaging in this behavior, many animal groups display structural order, behaving in a way that they appear to move as a single coherent entity, while still changing their shape and direction. They show strong spatial coherence and are capable of fast, highly synchronized maneuvers, either spontaneously, or as a response to predator attacks. *Flocking* behavior is the behavior exhibited when a group of birds, called a flock, are foraging or in flight. It is an example of an emergent collective behavior: There is no leader, nor global control and this phenomenon emerges only from the local interactions.

To simulate this emergent collective behavior, individual based modeling, and especially Agent-Based Models (ABM) (Michel et al., 2009), have proved to be the most relevant compared to modeling approach based on mathematical equations (EBM, Equations Based Modeling). Indeed, as first highlighted by (Reynolds, 1987), producing an animated flock of birds presents significant difficulties. Scripting the path of a large number of individual objects using traditional computer animation techniques would be very complicated and would produce unrealistic movement. Given the complex paths that birds follow, it is uncertain that this specification could be made without error. Even if a reasonable number of suitable paths could be described, it is unlikely that the constraints of flock motion could be maintained.

It is therefore more relevant to consider each simulated bird as an independent actor that navigates according to its local perception in a dynamic environment. In this respect, Reynolds's work (Reynolds, 1987) was revolutionary because it is the first which considers each entity (called *boid*) of a flock as autonomous and in interaction. In Reynolds's modeling, the aggregate motion of the simulated flock is thus the result of local interactions between simulated birds, each of which relying on simple behavioral rules detailed in the next section. Reynolds's behavioral rules have since inspired a lot of research and is cited in numerous fields wherein collective motions are involved.

In the scope of Artificial Life, Reynolds's boids are considered as a classic ABM and is implemented as a flagship example in many ABM platforms (Hermellin and Michel, 2016b). Still, there is no consensus on how Reynolds's boids should be implemented and there exists numerous interpretations (see e.g. (Bajec et al., 2007)). Especially, it turns out that tricks are always used to obtain relevant and interesting group motions: Additional global rules, forces or point of interests are always added. Without such adjustments, the boids gather into a boring homogeneous flock that never changes its global direction. That is why a global direction, a bird leader with a specific behavior, or a global force are used to influence the flocks motion. So, many of the works that extend Reynolds's boids define *additional inputs* (or features) over Reynolds's rules:

- Anderson et al. (2003) introduce a complementary force to the alignment which change the leadership of the flock. This steer defines the chance of a boid to become a leader and try to escape.
- Husselmann and Hawick (2011) add new species (predator) in the simulation to influence the starlings and their behaviors.
- Su et al. (2009) add a virtual leader to force the global murmuration of the flock.
- Lai et al. (2005) introduce a constrained group of animations. Flocks of agents move on a predefined path.
- Anderson et al. (2003) describe a constraint-based system for flock behavior using group motion graphs.
- Delgado-Mata et al. (2007) use virtual fear to enhance flock behavior.
- All videos on Youtube use *additional inputs* over Reynolds' rules [1].

All these boids-based models are even more complex when the objective is explicitly to simulate murmuration-like dynamics. Indeed, starlings murmuration are group motions involving sudden and spontaneous changes of direction, as well as the splitting and merging of local subgroups of starlings.

In this respect, the work done by the authors of (Hildenbrandt et al., 2010; Hemelrijk and Hildenbrandt, 2015) for modeling murmuration is actually the most advanced. As they explain in Hildenbrandt et al. (2010), usual boids-based swarming models lack the complexity of the flocking maneuvers of starlings. So, considering that studies of self-organization of large flocks of birds were missing Hildenbrandt et al. (2010), these authors proposed a complex 3D flocking model (involving more than twenty parameters), called StarDisplay, and validated it by comparing the produced patterns to empirical real-world data[2]. Some of the patterns produced by StarDisplay resemble remarkably of aerial display of starlings qualitatively and also quantitatively. Still, because it focuses on modeling accurately the very nature of murmuration dynamics for studying and understanding real flocks, StarDisplay relies on a very detailed and complex model which involves a lot of parameters which purpose is to increase the reliability of the simulations compared to the empirical data. Moreover, StarDisplay also involves additional global rules/forces[3] and the existence of a point of interest (e.g. a force steering the flock toward the roost).

While our main objective is not to simulate realistic murmuration, we are particularly interested in producing boids-based simulations exhibiting core aspects of murmuration dynamics, namely (1) sudden and spontaneous changes of direction with potentially winding moves (property tagged as P1 hereafter), (2) splitting and merging of local subgroups (P2), and change in shape and density (P3). More specifically, the goal of this paper is to show that this is possible by both using a very simple model and without requiring any global force nor input for defining the individual behaviors. So, the underlying idea is to ”strictly” use an Alife perspective, in the sense that the holistic emergent phenomena is the result of interactions between independent and entities (Liekens, 2003), thus following a pure bottom-up approach. We show in this paper how we have achieved this objective by relying on the IRM4S (Influence Reaction Model for Simulation) modeling perspective: A modeling approach dedicated to ABM which eases the representation of simultaneous interactions between individuals (Michel, 2007).

This paper details the creation of a flocking model, its implementation and also the resulting simulation obtained thanks to the IRM4S approach. More precisely, Section II presents the original Reynolds's model. Section III proposes an overview on how flocking has been implemented in MABS (Multi-Agent Based Simulation) and compares the resulting simulations. Section IV introduces the IRM4S modeling approach and describes the implementation of our flocking model. Section V analyses and discusses the results of our experiments. Finally, section VI concludes this paper.

Reynolds's boids (Reynolds, 1987)

At the origin of the work on flocking, Reynolds wanted to achieve a believable animation of a flock of artificial birds, namely *boids*. He remarked that it was not possible to use a scripted flock motion to achieve a realistic animation. It is more interesting to promote a bottom-up modeling approach where the global behavior of the system emerge from the interactions between entities. So, his idea was that boids have to be influenced by the others to flock in a coherent manner.

[1]`https://www.youtube.com/playlist?list=PLAsKueXMR2Aq_T0eo3j1uaJ-B2i0cTz6S`

[2]These data have been extracted from extensive video recordings done in Rome Ballerini et al. (2008).

[3]For instance, the behavior of the birds is influenced by a factor indicating the degree to which an individual is peripheral with respect to the whole flock.

So Reynolds proposes that each agent of the model is subjected to forces that make it move by taking into account the interactions with the others. To this end, each boid follows three behavioral rules:

1. *Collision Avoidance*: avoid collisions with nearby flockmates (here identified as R1);
2. *Flock Centering*: attempt to stay close to nearby flockmates (here identified as R2);
3. *Velocity Matching*: attempt to match velocity with nearby flockmates (here identified as R3).

Boids implementations in MABS platforms

Given that Reynolds's boids is one of the most representative ABM, we compare several implementations of this model that we can find in popular MABS platforms. Among the related implementations found, we only introduce models we were able to download and try with an open source code.

Description of various implementations

In NetLogo[4] (Sklar, 2007), all the agents move and try to get closer to their peers. If the distance between them and the nearest neighbor is too small, the agent tries to get away (avoid collision (R.1)), otherwise the agent aligns with its neighbors (R.2). However, there is no speed management (R.3): All the agents have the same velocity during the entire simulation.

In StarLogo[5] (Resnick, 1996), each agent searches for his closest neighbor. If the distance to his peer is too small, then it turns and gets away to avoid collision (R.1). Otherwise, it moves toward him and use his direction. The search for cohesion (R.2) is not explicitly expressed and the velocity of the agents is fixed throughout the simulation (R.3).

In GAMA[6] (Grignard et al., 2013), agents first look for a virtual target to follow (a moving object in the environment that initiates the flocking behavior). Once the agents have a target, they move according to three functions that implement Reynolds' rules: A separation function to avoid collision (R.1), a cohesion function (R.2) and an alignment function for speed and direction (R.3). The model differs from Reynolds' because the agents need a target to actually make the flocking.

In MasOn[7] (Luke et al., 2005), the implemented flocking model is quite different from the others because it uses the computation of several vectors to integrate R.1 and R.2. Each agent computes a motion vector composed of an avoidance vector (computed as the sum, over all neighbors, of vectors to get away from the neighbors (R.1)), a cohesion vector (a vector toward the "center of mass" of nearby flockmates), a momentum vector (a vector in the direction the flockmate went last time), a coherence vector (the direction where other flockmates are heading (R.2)), and a random vector. Velocity is not managed in this model (R.3).

[4]`https://ccl.northwestern.edu/netlogo/`
[5]`http://education.mit.edu/starlogo/`
[6]`https://code.google.com/p/gama-platform/`
[7]`http://cs.gmu.edu/~eclab/projects/mason/`

In Repast[8] (North et al., 2007), R.1 and R.2 are explicitly implemented. However, these two behavioral rules are successively executed by the agents in a single behavior. Moreover, in the cohesion behavior, the distance between the agents is fixed and does not result from the interactions. Finally, we note that R.3 is not implemented.

In FlameGPU[9] (Richmond et al., 2010), the integrated flocking model uses GPGPU (General-Purpose on Graphics Processing Units), an HPC (High Performance Computing) technology. In this model, the three Reynolds' rules are clearly implemented but in a single function. R.1, R.2 and R.3 are executed successively in the main behavior of each agent. Moreover, these rules use some barycenters (e.g. the flock center) to compute the next position according to flockmates, the speed, etc.

Resulting simulations

Looking at the resulting simulations of the flocking models presented above, we notice big differences between them in terms of flocking dynamics and movement believability. Indeed, the two most convincing simulations are those proposed by FlameGPU and GAMA for which flocks show spatial coherence and synchronized fast maneuvers. In the opposite, the other simulations are very simple with poor dynamics and coordination.

From these simulations, it appears that the most believable ones, and more specifically those exhibiting P1 or P2, are those that not only implement all Reynolds's rules, but more remarkably those that use additional features or global parameters (as summarized in Table 1): Virtual point of interest in GAMA and barycenters in FlameGPU.

The difficulty of implementing dynamics such as P1, P2 or P3, and the fact that some global features and parameters are required for this, can be explained to some extent by the fact that there is no formalization of Reynolds's boids. Indeed, despite a large amount of works, no formal definition has ever been presented (Bajec et al., 2007). There is no precise definition either from a mathematical, behavioral, technical point of view, etc. So, the three flocking rules expressed in (Reynolds, 1987) are subject to broad interpretation which in turn complicates their analysis and implementation. As an example, the second rule is also known as the *alignment* rule when interpreted as an attempt to have the same attitude as nearby flockmates. The problem is that expressions like attempt to stay close and attempt to match velocity have broad meanings.

[8]`http://repast.sourceforge.net/`
[9]`http://www.flamegpu.com/`

Platform	Main characteristics	Additional features	Resulting dynamics
NetLogo	R.2 is implemented as "*alignment*" behavior		Simple
StarLogo	Only R.1 is implemented		Poor
GAMA	All rules are implemented	Virtual target / obstacles	Suitable (P1 & P2)
MasOn	R.1 and R.2 are reinterpreted into a global vector		Simple
Repast	R.1 and R.2 integrated into a single behavior		Simple
Flame GPU	All rules are implemented	Some barycenters are used	Convincing (P1 & P2 & P3)

Table 1: Comparison between flocking implementations in common MABS platforms.

Moreover, most of the existing implementations are not available nor reusable, especially considering murmuration dynamics since academic research is mainly oriented toward classic boids flocking dynamics (Hildenbrandt et al., 2010).

Our flocking model

In the scope of this research, we want our model to have two major characteristics: (1) be as simple as possible while exhibiting group motions having qualities related to P1, P2 and P3 and (2) not use any stimulus, feature or point of interest related to a global point of view.

To achieve this goal, we took inspiration from all the implementations we have studied (2D), so that our model does integrate R.1, R.2 and R.3, while also following the KISS (Keep It Simple and Stupid) principle in the aim of creating a minimalist version. In our model, we have two types of parameters: 5 constants (*fieldOfView*, *minimalSeparationDistance*, *cohesionThreshold*, *maximumSpeed* and *maximumRotation*) and 3 attributes specific to each agent (*heading*, *velocity* and *nearestNeighborsList*).

So, the behavioral model of an agent only relies on evolving the speed and the heading according to its local neighbors [10]. To this end, the proximity with local neighbors is first tested and then Reynolds's rules are triggered accordingly. More specifically, if there is no agent around then it continues to move in the same direction. Otherwise, the agent checks if the neighbors are not too close. Depending on the proximity between entities, agents separate (R.1), align with other entities or create cohesion (R.2). Then agents adapt their speed (R.3) and move. Figure 1 summarizes the global behavior process.

The global behavior process for each entity is divided into three behaviors:

- (R.1) Separation: When an agent is too close from another one, it separates. This behavior consists in retrieving the heading of both agents. If these two directions lead to a collision, agent rotates to avoid its neighbor.

- (R.2)
  - Align: When an agent comes closer to others, it tries to align itself with them, by adjusting his direction according to its nearest neighbor.
  - Cohesion: When multiple agents are close without having to separate, each agent retrieves the directions of its neighbors and adjusts its own heading based on the average direction computed within its field of view.

- (R.3) Speed Adaptation: During all the simulation, every agent modifies its speed according to that of its neighbors.

[10]The orientation is an angle in degree (between 0 and 360) which gives the heading of the agent according to the landmark fixed in the environment.

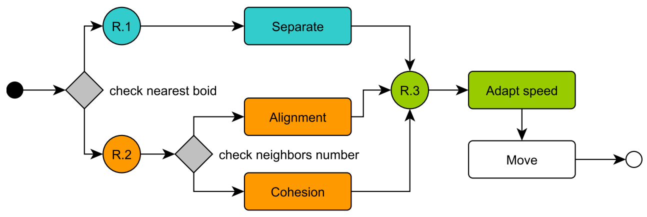

Figure 1: Flocking global behavior process.

IRM4S implementation

The IRM4S model

The IRM4S model (Influence Reaction Model for Simulation) (Michel, 2007) is an adaptation of the formalism of (Ferber and Müller, 1996) for multi-agent based simulations. Its purpose is to address some shortcomings of the usual representation of agent actions in ABM. Indeed, agent actions are usually modeled as direct modifications of the environment, which may raise a number of issues at both the modeling and implementation levels. In the scope of this paper, one crucial issue is that it does not allow to easily model simultaneous actions and interactions since the result of the action of each entity is computed without considering others'. Without going into the details of the underlying conceptual aspects of IRM4S, it is important to note that the modeling issues which are addressed by IRM4S not only integrate qualitative differences in outcomes under synchronous vs. asynchronous updating of agents as studied by Huberman and Glance (1993), but go beyond as IRM4S also deals with the representation of complex interaction schemes (Michel et al., 2004). In particular, one major idea is that an agent should not be allowed to compute the result of its actions because it cannot know exhaustively the environmental settings and especially the other actions which take place at the same time.

So, IRM4S relies on two notions: (1) influences and (2) reaction to influences. So, an agent does not perform actions but produces influences: Influences do not directly modify the environment and, from the agent's point of view, nothing can be guaranteed about their result. This perspective enables to distinguish the individual gestures (agent level) from what actually happens considering the other gestures: The environment reaction to all the influences (multi-agent level). So, the reaction cannot be computed without knowing all the influences which are produced at the same time. Applying IRM4S thus requires a two phases mechanism that (1) collects the influences (influence phase), and then (2) compute the result of their combination (reaction phase).

While IRM4S can be implemented using a sequential programming approach, one major issue is that its two-phases computation mechanism requires a lot of computing resources. So, since we targeted simulations with thousands of boids, it became necessary to use an HPC technology: GPGPU (a massively parallel programming approach for performing intensive computations on GPU)[11]. However, the GPGPU technology requires adopting a new programming approach which is very tricky to use for implementing ABM. That is why we used the approach proposed in (Michel, 2013; Hermellin and Michel, 2016a), namely GPU delegation, which is specifically designed for modeling and implementing MABS using GPGPU.

Implementation using GPU delegation

To implement our IRM4S flocking model using GPGPU, we follow the *GPU delegation* principle which is mainly about making an explicit distinction between the behaviors of the agents, handled by the CPU, and the environmental dynamics, managed by the GPU. Moreover, it turns out that achieving this naturally creates a two-phases mechanism which therefore matches the one required when implementing an IRM4S approach. Indeed, contrarily to a usual modeling for which agent actions are directly committed in the environment sequentially, one step of this simulation is composed of two distinct phases (see Algorithm 1 and Figure 2):

- The agents activation phase (influence, *phase 1*): The agents produce influences.
- The data processing phase (reaction, *phase 2*): The environment reacts to influences and computes data (local average of neighbors' heading) that will be used by the agents in the next simulation step.

[11]To further introduce GPGPU, we put on GitHub some simple examples (addition of a vector, computation of π, etc.) and their parallelization on CPU and GPU: `https://github.com/ehermellin/IntroHPC`. Owens et al. (2007) also proposes an interesting overview of this technology.

And because the reaction phase requires to make local computation, everywhere in the environment, this can be done in a very efficient parallel way with GPGPU.

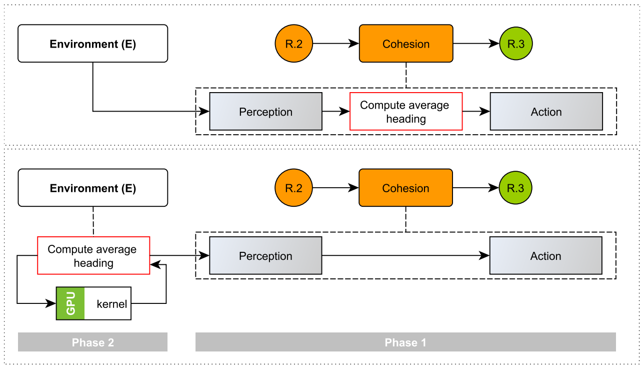

Figure 2: Comparison between R.2 implementation without IRM4S (top) and whith IRM4S/GPGPU (bottom).

Algorithm 1: Flocking model, simulation scheduling.

```
init($environment$,$agents$);
while $simulationRunning$ do
    /* Phase 1: Agents' activation */
1   foreach $agent$ in $listOfAgents$ do
2       percept();
3       live();
4       fillEnvironment($flockCentering$[],$heading$);
5   end
    /* Phase 2: Environment's activation */
6   executeGPUKernel(
      computeAverage($flockCentering$[]) );
end
```

Concretely, in the model we propose, the cohesion behavior (R.2) contains a computation that consists in collecting and processing data: Averaging the orientations of neighbors according to a particular *FieldOfView*. All the agents perform this computation in their own behavior and use the result to adapt their direction (see Figure 2). That is why, according to the GPU delegation method, we transform all these agent computations into a single environmental dynamics computed by a GPU kernel[12]. So, the environment is in charge of processing, in a global way, the perception data that the agents will need locally. To this end, at each simulation step, each agent put its heading value in a 2D array (*headingArray*) according to its position. Then this array is processed by the GPU kernel that simultaneously, for each cell, according to a *fieldOfView*, performs the average of the headings of the surrounding agents. More precisely,

[12]*Kernels* are functions executed on the GPU. These kernels are executed by many GPU threads (identified with unique id, see below) in parallel.

each *thread(i,j)* of the GPU computes the average for a cell depending on its location in the global GPU grid (its identifiers: *i* and *j* in Algorithm 2). Once done, the average headings are available in every cells of the environment. So, the agents can access this data instantaneously with respect to their position and thus adapt their movement accordingly[13].

Algorithm 2: Kernel that compute average heading.

```
input  : $width$, $height$, $fieldOfView$, $headingArray$ and
         $nearestNeighborsList$
output: $flockCentering$ (the average of directions)
1 $i = blockIdx.x * blockDim.x + threadIdx.x$ ;
2 $j = blockIdx.y * blockDim.y + threadIdx.y$ ;
3 $sumOfHeading, flockCentering = 0$ ;
4 if $i < width$ and $j < height$ then
5     $sumOfHeading =$ getHeading($fieldOfView$,
        $headingArray[i, j]$);
6 end
7 $flockCentering[i, j] =$
    $sumOfHeading / sizeOf(nearestNeighborsList)$ ;
```

Experimental results and discussion

Experimental Protocol

To trial our model, we also implemented it using a usual sequential programming approach and tested both versions using different environment sizes (256×256 and 512×512) while varying the number of agents (4000, 8000 and 10000) and the field of view (between 5 and 10). We have executed dozens of simulations of each version using different initial configurations with respect to the agents' location and orientation. So, the agents initially are either evenly distributed or placed in the center and have either the same heading or a random one.

Results and discussion

Analyzing the simulations, the main result is undoubtedly that, while they are based on exactly the same behavioral model, the resulting collective dynamics are very different depending on the approach which is used (see Figure 4).

In the CPU version, the flocking behavior works well but the global dynamics is rather simple and boring in the sense that it does not produce clearly any of P1, P2 or P3. Contrarily, with the IRM4S version, the global dynamics is much more chaotic, complex and does exhibit simultaneously P1, P2 and P3. Indeed, one can see different groups of boids which are able to split or merge, while flocking according to directions that may suddenly change.

It turns out that this is mainly due to the underlying modeling principles on which relies IRM4S (influences and reactions). In the CPU version, the boids perceive and act directly one after the other. So, a simulation step leads to the convergence of the system towards a common value for the orientation of the agents. The first boid modifies its direction according to its perception. The second boid does the same but may perceive the new direction of the first one and thus take it into account computing its own new direction, so does the third, and so on. Therefore, after only a few simulation steps, the boids eventually act in a very homogeneous world: All boids move in the same direction, modulo a small random variation on the individual directions (see Figure 3).

On the contrary, using IRM4S, all the data for the perceptions are pre-processed by the environment (as reaction to the influences) and all the agents perceive the same state of the world for a unique timestamp t. Therefore, using IRM4S, the headings of the agents do not globally converge to a particular value (see Figure 3). So local flocking dynamics can be observed and the system does not globally evolve toward an equilibrium.

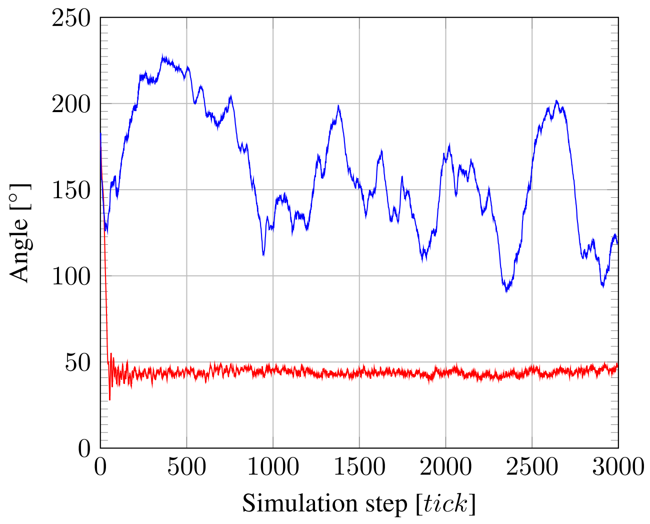

Figure 3: Average heading of all boids without IRM4S (red) and with IRM4S (blue).

Without an IRM4S approach, we would probably have modified parts of the boids behaviors, added *global stimulus* or *additional inputs* for obtaining a more convincing flocking exhibiting all the desired properties. Here, considering strictly the same behavior, it is the collective dynamics that we can make more complex, thanks to IRM4S. One can have an overview of the resulting simulations thanks to a set of videos that shows both versions in action: `https://www.youtube.com/playlist?list=PLAsKueXMR2Aox2cVERnnR-cmpA8g_cXcG`.

Conclusion

The collective behaviors exhibited by starlings, called murmuration, are part of swarm phenomena which are both fascinating and hard to explain. In this paper, we have high-

[13]The source codes and other resources can be found at `https://github.com/ehermellin/Flocking_GPGPU_TurtleKit`

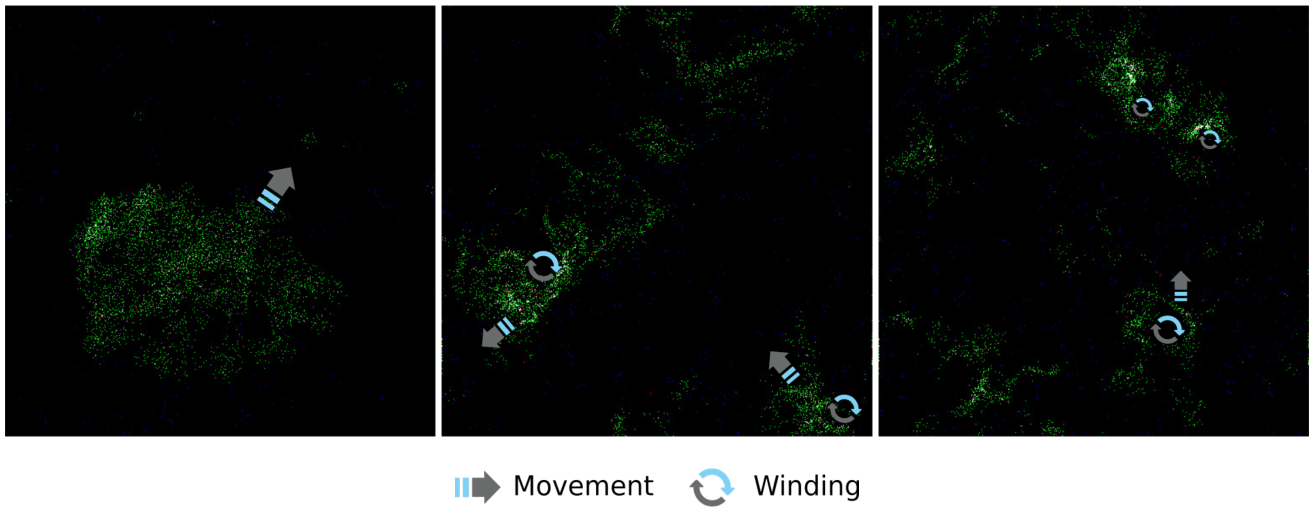

Figure 4: Resulting flocking simulations without IRM4S (the one on the left) and with IRM4S (the two on the right).

lighted that all boids simulations enhance Reynolds's rules with *global stimulus* or *additional inputs/features* to obtain murmuration-like dynamics such as P1, P2 or P3. In this respect, the use of virtual leaders or points of interest are commonly used for orienting the boids and influencing their movements. For instance, without these tricks, boids simulations only produce coordinated flights that do not show sudden and spontaneous changes in direction as observed in murmuration.

In this paper, we have shown that it is possible to obtain complex coordinated flight dynamics using a very simple ABM and without adding external stimulus nor additional features, that is by only implementing Reynolds's rules. Especially, we argue on the idea that this is possible thanks to the use of an approach such as IRM4S, which is an alternative to usual modeling approaches relying on the notions of influence and reaction. Moreover, to deal with the amount of computing resources that IRM4S requires, we have used a massively parallel programming approach, namely GPGPU, which enabled us to experiment our model in a number of simulations, using huge boids populations.

Experimenting our flocking model, the results have proved that the global dynamics of the flocks are much more chaotic and surprising when IRM4S is implemented. Our approach therefore enable to obtain complex dynamics even if only individual behaviors are represented. Moreover, this work confirms the crucial role of the update regime in ABM as described in Huberman and Glance (1993); Michel et al. (2004) for instance.

So, beyond the fact that we are able to produce boids dynamics having characteristics such as P1, P2 and P3, we consider that obtaining a so huge difference between the two versions is as a major result of this research and think that this more globally promotes the urge of revisiting usual multi-agent based simulations and how we implement them. Indeed, because of our way of conceiving or implementing simulations, we enclose ourselves in a pattern of thought that limits us, so that a lot of dynamics from ABM remain to be explored. And this is crucial from an ALife perspective in the sense that this represents another example of the fact that we do not necessarily need complex behavioral models to obtain complex dynamics. This represents a promising research perspective both for both ALife and ABM.

References

Anderson, M., McDaniel, E., and Chenney, S. (2003). Constrained animation of flocks. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 286–297, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Bajec, I. L., Zimic, N., and Mraz, M. (2007). The computational beauty of flocking: boids revisited. *Mathematical and Computer Modelling of Dynamical Systems*, 13(4):331–347.

Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., and Zdravkovic, V. (2008). Empirical investigation of starling flocks: a benchmark study in collective animal behaviour. *Animal Behaviour*, 76(1):201 – 215.

Delgado-Mata, C., Martinez, J. I., Bee, S., Ruiz-Rodarte, R., and Aylett, R. (2007). On the use of virtual animals with artificial fear in virtual environments. *New Generation Computing*, 25(2):145–169.

Ferber, J. and Müller, J.-P. (1996). Influences and reaction: a model of situated multiagent systems. In *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 72–79.

Grignard, A., Taillandier, P., Gaudou, B., Vo, D., Huynh, N., and Drogoul, A. (2013). GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation. In Boella, G., Elkind, E., Savarimuthu, B., Dignum, F., and Purvis, M., editors, *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, volume 8291 of *Lecture Notes in Computer Science*, pages 117–131. Springer Berlin Heidelberg.

Gueron, S., Levin, S. A., and Rubenstein, D. I. (1996). The dynamics of herds: From individuals to aggregations. *Journal of Theoretical Biology*, 182(1):85 – 98.

Hemelrijk, C. K. and Hildenbrandt, H. (2015). Diffusion and topological neighbours in flocks of starlings: Relating a model to empirical data. *PloS one*, 10(5):e0126913.

Hermellin, E. and Michel, F. (2016a). Gpu delegation: Toward a generic approach for developping mabs using gpu programming. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 1249–1258, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Hermellin, E. and Michel, F. (2016b). GPU Environmental Delegation of Agent Perceptions: Application to Reynolds's Boids. In Gaudou, B. and Sichman, J. S., editors, *Multi-Agent Based Simulation XVI*, volume 9568 of *Multi-Agent Based Simulation XVI*, pages 71–86. Springer International Publishing.

Hildenbrandt, H., Carere, C., and Hemelrijk, C. K. (2010). Self-organized aerial displays of thousands of starlings: a model. *Behavioral Ecology*, 21(6):1349–1359.

Huberman, B. and Glance, N. (1993). Evolutionary games and computer simulations. *Proceedings of the National Academy of Sciences of the United States of America*, 90(16):77167718.

Husselmann, A. V. and Hawick, K. A. (2011). Simulating Species Interactions and Complex Emergence in Multiple Flocks of Boids with GPUs. In *Int. Conference on Parallel and Distributed Computing and Systems*, pages 100–107. IASTED.

Lai, Y.-C., Chenney, S., and Fan, S. (2005). Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 281–290, New York, NY, USA. ACM.

Liekens, A. (2003). Artificial life. In *Encyclopedia of Computer Science*, pages 93–96. John Wiley and Sons Ltd., Chichester, UK.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). MASON: A Multiagent Simulation Environment. *Simulation*, 81(7):517–527.

Michel, F. (2007). The IRM4S Model: The Influence/Reaction Principle for Multiagent Based Simulation. In Durfee, E. H., Yokoo, M., Huhns, M. N., and Shehory, O., editors, *Proc. of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007*, AAMAS '07, pages 903–905. ACM.

Michel, F. (2013). Translating agent perception computations into environmental processes in multi-agent-based simulations: A means for integrating graphics processing unit programming within usual agent-based simulation platforms. *Systems Research and Behavioral Science, Special Issue: Conceptual Models Of Complex Systems*, 30(6):703–715.

Michel, F., Ferber, J., and Drogoul, A. (2009). Multi-agent systems and simulation: a survey from the agents community's perspective. In Uhrmacher, A. and Weyns, D., editors, *Multi-Agent Systems: Simulation and Applications*, Computational Analysis, Synthesis, and Design of Dynamic Systems, pages 3–52. CRC Press - Taylor and Francis.

Michel, F., Gouaïch, A., and Ferber, J. (2004). Weak interaction and strong interaction in agent based simulations. In Hales, D., Edmonds, B., Norling, E., and Rouchier, J., editors, *Multi-Agent-Based Simulation III*, volume 2927 of *Lecture Notes in Computer Science*, pages 43–56. Springer Berlin Heidelberg.

North, M., Tatara, E., Collier, N., and Ozik, J. (2007). Visual agent-based model development with Repast Simphony. In *Agent 2007 Conference on Complex Interaction and Social Emergence*, pages 173–192, Argonne, IL, USA. Argonne National Laboratory.

Okubo, A. (1986). Dynamical aspects of animal grouping: Swarms, schools, flocks, and herds. *Advances in Biophysics*, 22:1–94.

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krger, J., Lefohn, A. E., and Purcell, T. J. (2007). A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum*, 26(1):80–113.

Potts, W. (1984). The chorus-line hypothesis of manoeuvre coordination in avian flocks. *Nature*, 309(5966):344–345.

Resnick, M. (1996). StarLogo: An Environment for Decentralized Modeling and Decentralized Thinking. In *Conference Companion on Human Factors in Computing Systems*, CHI '96, pages 11–12, New York, NY, USA. ACM.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34.

Richmond, P., Walker, D., Coakley, S., and Romano, D. M. (2010). High performance cellular level agent-based simulation with FLAME for the GPU. *Briefings in bioinformatics*, 11(3):334–47.

Shaw, E. (1975). Naturalist at large-fish in schools. *Natural History*, 84(8):40.

Sklar, E. (2007). NetLogo, a Multi-agent Simulation Environment. *Artificial Life*, 13(3):303–311.

Su, H., Wang, X., and Lin, Z. (2009). Flocking of multi-agents with a virtual leader. *IEEE Transactions on Automatic Control*, 54(2):293–307.