Complex flocking dynamics without global stimulus

Emmanuel Hermellin & Fabien Michel
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
ehermellin.um2@gmail.com fmichel@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 accepted that this collective behavior emerges from individual ones and that no global control is involved. In this scope, Reynolds' individual-based rules have been investigated a number of times. But it turns out that all murmuration simulations add tricks to these rules to achieve convincing animations of this phenomenon. Especially, virtual leaders or points of interest are used to orientate the starlings, which somehow contradicts the no-global-control perspective. In this research, we show that, thanks to the IRM4S modeling perspective (an Influence Reaction Model for Simulation), we have obtained complex coordinated flight dynamics using a very simple Agent-Based Model (ABM) and without adding external stimulus nor additional features.

Introduction

Flocking is the behavior exhibited when a group of birds, called a flock, are foraging or in flight. It is today understood as an emergent phenomenon: There is no leader nor global control and the corresponding swarm behavior emerges only from local interactions.

For achieving a believable animation of a flock of artificial birds (*boids*), Reynolds promoted a bottom-up modeling approach involving three simple individual-based rules illustrated in Figure 1 [4]:

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

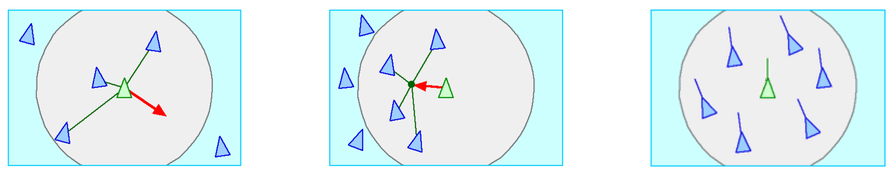

Figure 1: Reynolds's rules from left to right: *Collision Avoidance* (R1), *Flock Centering* (R2), *Velocity Matching* (R3).

Still, **there is no consensus on how boids should be implemented** and there exists numerous interpretations. Especially, various tricks are used to obtain relevant and interesting group motions: Additional global rules, forces or point of interests, etc. Without such adjustments, the boids gather into a boring homogeneous flock that never changes its global direction. As explained in [2], **usual boids-based swarming models lack the complexity of the flocking maneuvers of starlings**.

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

Main Objectives

1. Produce **large scale boids-based simulations exhibiting core aspects of murmuration dynamics**:
   - Property 1 (**P1**): **Sudden and spontaneous changes of direction** with potentially winding moves.
   - Property 2 (**P2**): **Splitting and merging of local subgroups**.
   - Property 3 (**P3**): **Change in shape and density of the swarms**.
2. Show that **this is possible using only Reynolds's rules**: **No external stimulus nor additional features**
3. Show how **this can be achieved thanks to IRM4S: an *Influence Reaction Model for Simulation***[3]

Our flocking model

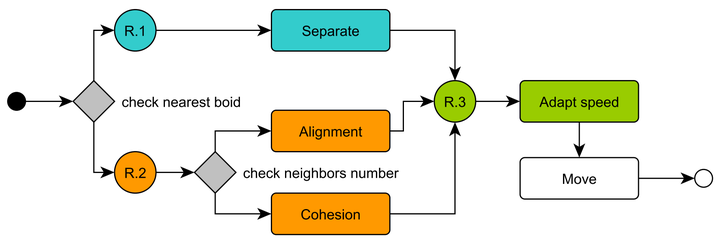

Figure 2: Flocking individual behavior process

Our model **does integrate R1, R2 and R3**, but also **follows the KISS principle** (Keep It Simple and Stupid) for creating a minimalist version so that we have **5 constants** (*fieldOfView*, *minimalSeparationDistance*, *cohesionThreshold*, *maximumSpeed* and *maximumRotation*) and **3 attributes** specific to each agent (*heading*, *velocity* and *nearestNeighborsList*).

IRM4S implementation using GPGPU

The IRM4S model

The *Influence Reaction Model for Simulation* **IRM4S [3] addresses some shortcomings of the usual representation of agent actions in ABM**. Indeed, modeling the action as direct modifications of the environment raises a number of issues. Especially, it does not allow to easily model simultaneous actions and interactions since the result of the actions is computed without considering others'.

So, **IRM4S relies on two notions: (1) influences and (2) reaction to influences**. Agents do not perform actions but produce influences ($\gamma(t) \in \Gamma$): **Influences do not directly modify the state of the environment** ($(\sigma(t) \in \Sigma)$ and nothing can be guaranteed about their result. This distinguishes the individual gestures (agent level) from what actually happens, that is the environment's reaction to all the influences (multi-agent level). Applying IRM4S thus requires a two phases mechanism that (1) collects the influences (influence phase), and then (2) computes the result of their combination (reaction phase).

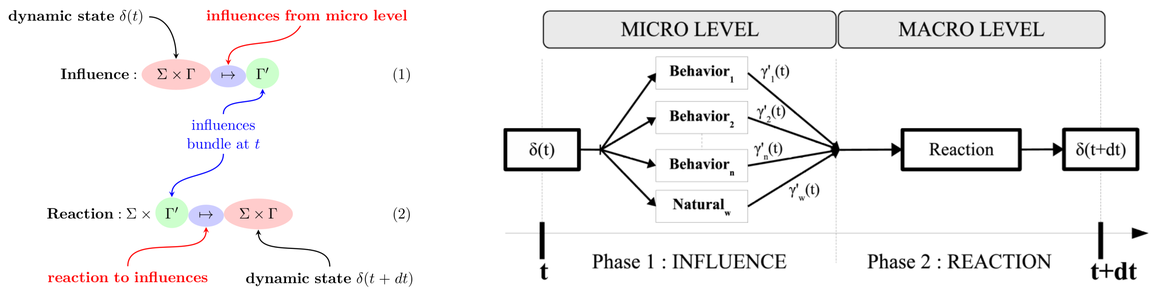

Figure 3: IRM4S: Influence Reaction Model for Simulation [3]

IMR4S Large Scale Simulations Using GPU Delegation

For achieving large scale simulations with our IRM4S flocking model, we used the GPGPU technology. To this end, **we follow the *GPU delegation* method** [1] which is about making an explicit distinction between the behaviors of the agents, handled by the CPU, and the environmental dynamics, managed by the GPU. Additionally, to boost performances in the scope of ABM, GPU delegation relies on identifying agent behaviors which can be transformed into environmental dynamics, thus allowing to transfer some computations from the CPU to the GPU. In our boids model, the agent's cohesion behavior (R.2) consists in averaging the neighbors' headings wrt a *FieldOfView*. According to GPU delegation, we transformed all these agent computations (CPU) into a single environmental dynamics computed by a GPU kernel (i.e. functions executed on the GPU by many threads in parallel).

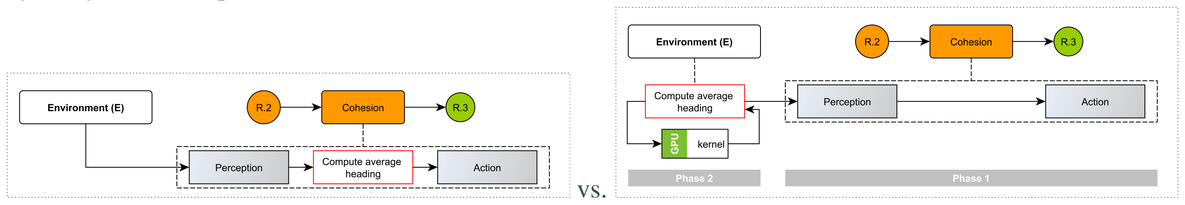

Figure 4: Comparison between R.2 implementations: Without IRM4S (left) and whith IRM4S/GPGPU (right).

So, **the environment globally processes the perception data that the agents will need locally**. So, for every simulation step, each agent put its heading in a 2D array (*headingArray*) which is then processed by the GPU kernel that simultaneously computes, for each cell (the environment is a 2D grid), the average of the headings of the surrounding agents. More precisely, each *thread(i,j)* of the GPU computes the average for a cell depending on its location (its identifiers: i and j in Algorithm 2). Once done, **the average orientations are available everywhere and the agents can access this data instantaneously**.

Algorithm 1: Simulation scheduling

```
while simulationRunning do
    /* Phase 1:  INFLUENCE                    */
    foreach agent in listOfAgents do
        perceive();
        flock();
        fillEnvironment(flockCentering[],heading);
    end
    /* Phase 2:  Environment's REACTION       */
    executeGPUKernel( computeAverage(flockCentering[]) );
end
```

Algorithm 2: *computeAverage* GPU kernel

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

Moreover, it turns out that achieving *GPU delegation* naturally implies a two-phases mechanism matching the one required when implementing IRM4S. So, one simulation step is composed of two distinct phases (see Algorithm 1 and Figure 4): *Influence*, *phase 1*) and *Reaction*, *phase 2*.

Experimental results

To trial our model, we also implemented it without IMR4S and tested both versions with different environment sizes (256×256 and 512×512), number of boids (4000 → 10000) and field of view values (5 → 10).

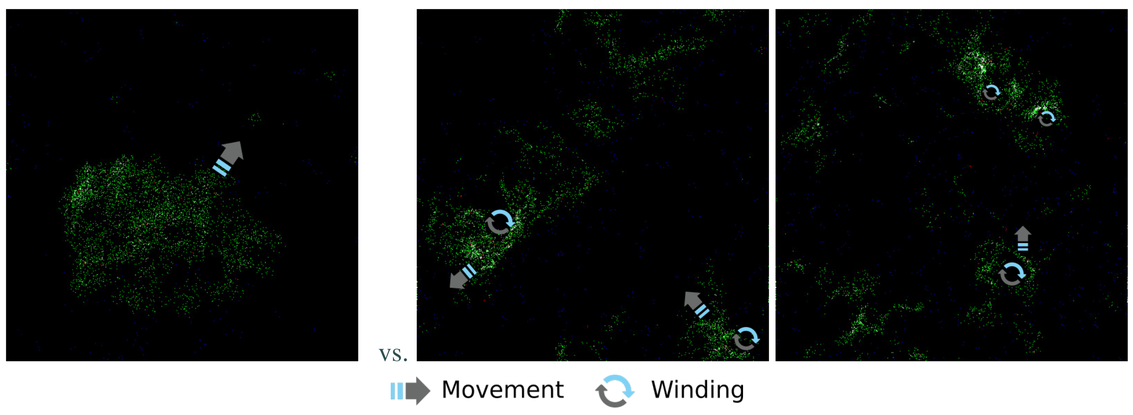

Figure 5: Resulting flocking simulations without IRM4S (the one on the left) and with IRM4S (the two on the right).

One major result is that, while based on exactly the same behavioral model, collective dynamics are qualitatively very different depending on the implementation approach which is used (Figures 5 and 6).

Without IRM4S, the flocking behavior works well but the global dynamics is rather simple and boring: It does not produce clearly any of P1, P2 or P3. Contrarily, with IRM4S, 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.

This is due to the underlying modeling principles on which relies IRM4S. Without IRM4S, the boids perceive and act directly one after another so that the swarm quickly converges toward a common value for the agents' headings (Figure 6). 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.

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, the system does not globally converge to a particular steady state.

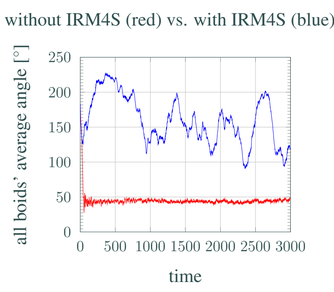

Figure 6: Swarm's average direction

Conclusions

- All existing boids simulations complete Reynolds's rules with *global stimuli* or *additional inputs/features* to obtain murmuration-like dynamics such as P1, P2 or P3. We showed that thanks to IRM4S, such tricks are not required to obtain complex coordinated flight dynamics, even with a very simple ABM.
- Beyond that result, this research (1) shows that a lot of ABM dynamics remain to be explored and (2) is another example of the fact that complex behavioral models are not required to obtain complex dynamics, which is crucial in the scope of Artificial Life.

References

[1] Emmanuel Hermellin and Fabien Michel. 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, 2016. IFAAMAS.

[2] Hanno Hildenbrandt, Cladio Carere, and Charlotte K Hemelrijk. Self-organized aerial displays of thousands of starlings: a model. *Behavioral Ecology*, 21(6):1349–1359, 2010.

[3] Fabien Michel. The IRM4S Model: The Influence/Reaction Principle for Multiagent Based Simulation. In *Proc. of the 6th Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007*, AAMAS '07, pages 903–905. ACM, 2007.

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