

**TELECOM**  
Paris



 **IP PARIS**

December, 2023

**Emergence**  
**in**  
**Complex Systems**  
Micro-studies

[teaching.dessalles.fr/ECS](https://teaching.dessalles.fr/ECS)

This document contains students' contributions written during the course "Emergence in Complex Systems" taught in November 2023.

We are very thankful to students for having be active participants and for the quality of their work.

*Jean-Louis Dessalles*

*Ada Diaconescu*



# Contents

<i>Name</i>	<i>First Name</i>	<i>Title</i>	<i>Page</i>
<b>Adriaensen</b>	Rik	The iterated Hawk-Dove dilemma	5
<b>Benammar</b>	Bilèl	Cellular Automaton's Game of Life	9
<b>Bugala</b>	Kacper	Simulating cellular automata with non-deterministic factor and increased number of states	11
<b>Cardot</b>	Aymeric	How should you signal yourself in a global emergency ?	15
<b>Caris</b>	Corentin	Analogy of Schelling's Segregationism and phase transition in the Ising model	17
<b>Castelo Cardoso</b>	Diogo	Segregationism: different initial states and moving for other reasons.	19
<b>Chardon</b>	Katia	A more realistic ant simulation	21
<b>Ciasnocha</b>	Michał	Rock-Paper-Scissors 2d cellular automata	23
<b>Clavé</b>	Mathieu	Multiple-choice Hawk-dove competition	29
<b>Da Silva</b>	Larissa	Nature shapes in emergence of fractals	33
<b>Montefusco</b>			
<b>Dalmard</b>	Alban	Analyze of the sensitivity to assumptions of the EMH	39
<b>De Thomassin de Montbel</b>	Rémy Kristen	Desire Paths	41
<b>Hooft</b>	Donna	Wolf vs Sheep (vs Grass)	45
<b>Kuc</b>	Jan	Gender property added to emergence of segregationism	53
<b>Lambert</b>	Thibault	Cocktail party	63
<b>Lefer</b>	Florine	Multiple-choice Hawk-dove competition	29
<b>Mamilo</b>	Missora	A quest for Enquist Hawk-Dove Dilemma implementation	65
<b>Martins braga</b>	Arthur	Emergence of segregationism: adding criteria to choose where to move	69
<b>Melga</b>	João	Nature shapes in emergence of fractals	33
<b>Metz</b>	Valentin	Guessing game	75
<b>Mille</b>	Pierina	High risk behavior in mammals to breed	77
<b>Molina Delgado</b>	Santiago	Exploring the correlation between group segregation and tolerance level of the population	81
<b>Muscillo</b>	Domenico	A more realistic analysis of social dynamics during a cocktail party	87
<b>Palmiro de Freitas</b>	Lucas	Emergence of segregationism: adding criteria to choose where to move	69
<b>Rios</b>	Maël	3D cellular automaton and visualization on Evolife	89
<b>Roux</b>	Léo	Dating-App Simulation Implementation	93
<b>Sumukha</b>	Shridhar	Wolf vs Sheep (vs Grass)	45
<b>Świąch</b>	Maciej	Ant Colony Algorithm, Obstacles added to the simulation	97
<b>Tasci</b>	Beste	Exploring the correlation between group segregation and tolerance level of the population	81
<b>Thireau</b>	Raphaël	Countering the growing effect of "filter bubble"	105
<b>Wimmer</b>	Benedikt	War of Attrition	107
<b>Wu</b>	Zhiheng	Draw a French flag by diffusion and thresholding	111





ATHENS course : **TP-09**

# Emergence in Complex Systems

Micro-study

teaching.dessalles.fr/ECS

Name: **Rik Adriaensen**

## The Iterated Hawk Dove Dilemma

### Abstract

Making an iterated version of the Hawk Dove Dilemma allows us to study strategies that take into account the previous moves of yourself and your opponent. In this study two approaches to encode these strategies in genes are investigated. In the first approach the genes control the weights of a simple network taking the previous moves of the players as inputs, in the second approach the genes are used as weights for combining different strategies. Here we study an iterated version of the Hawk Dove Dilemma, where individuals play the game for a number of times every time they meet. They are awarded the average score they achieved. It is now possible to have more interesting strategies, potentially taking into account your own and you partners' previous moves. The question is, how can we encode such strategies in genes?

## 1 Network Approach

I have tried out two different approaches. The first approach makes use of a very simple network that takes as input the previous move of yourself and your opponent as well as a bias. The input -1 corresponds to a hawk move, 1 to a dove move and the input is 0 if it is the first move of a series of encounter. These inputs and the bias are connected using three weights to a standard neuron with sigmoid activation. The output of this neuron is interpreted as the probability of an individual to play Hawk on the next move. Each of the three weights is controlled by a gene encoded with 6 bits. To go from a genes' relative value to the corresponding weight, I make use of a simple linear transformation to a number between -1 and 1.

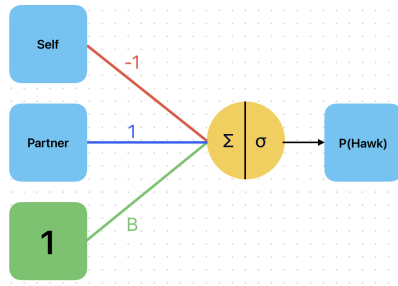


Figure 1: The implemented network.

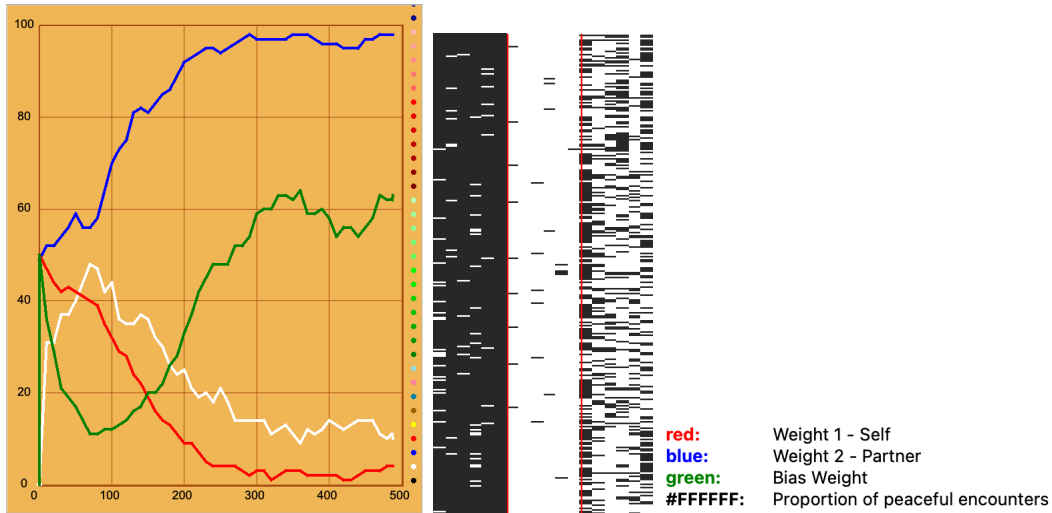


Figure 2: Results Network Approach; PieToShare = 10; CombatCost = 40; NbEncounters = 10

The results of this approach can be seen in figure 2. We observe that the gene controlling the weight of the own move is selected to be close to minimal and the gene controlling the weight of the partners' move to be almost maximal. Furthermore we observe an inverse correlation between the bias weight and the amount of peaceful encounters. This makes sense as the population will be more biased towards playing dove if the bias weight is low and vice versa, particularly on the first move of an encounter. This weight of the bias stabilises around a certain value, 0.6 in this case, which depends on how large the pie to share is and the combat cost.

Which strategy is encoded here? First we observe there is not much variation throughout the population for the first two genes but there is quite a bit of variation in the bias gene. This is not apparent from the averages alone and it means the competition between hawks and doves has not disappeared. On the initial move and each time both players make the same move, the probability of playing Hawk will depend solely or very strongly on the bias weight alone. However once the players play opposite moves, each player will have a very high probability of playing its opponents move on the next turn. This means these individuals will often get stuck in alternating patterns of playing dove and hawk. This is not a very effective strategy, it has similarities with the Tit-for-Tat strategy, but as the proportion of peaceful encounters indicates, it clearly doesn't optimise the average pay-off of the population. I believe this approach has only a limited ability to express different strategies in genes. That is why I have experimented with a second approach.

## 2 Ensemble Approach

Here each individual has access to five different strategies: Random, Hawk, Dove, Tit-For-Tat and Spiteful. The Tit-For-Tat strategy is to always play Dove on the first move but than mimic the other player. The Spiteful strategy is to always play Dove, until the other player plays Hawk once, after which it plays Hawk forever. Each individual has a gene, encoded with five bits each, for each of these strategies. On every move the five strategies propose a probability of playing Hawk. These probabilities are combined using a linear weighted sum, using the appropriate gene values as weights. The results of three runs with the same parameters are shown in the figure 4.

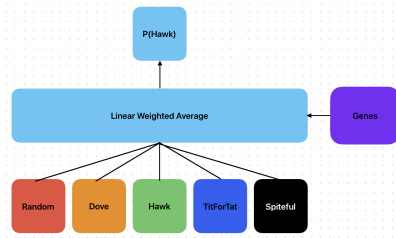


Figure 3: Schematic of the Ensemble approach.

Immediately it becomes clear that these individuals are able to achieve a much higher average pay-off over the population. The amount of peaceful encounters is significantly higher. The Dove strategy seems to be most popular on average which explains this high amount of peaceful interactions. The second most popular strategy is Tit-For-Tat. This all makes sense, as Tit-For-Tat and Dove work very well together. When they encounter each other, they will both always cooperate. The Hawk strategy however is almost never selected as it works badly against both Tit-For-Tat and Spiteful. Depending on the run the Spiteful strategy is favoured more or less. The same is true for the Random strategy. These results are similar to a conclusion from [0]. In that paper they look at the best strategies for playing a repeated version of the Prisoners' Dilemma and conclude that they are usually a combination between Tit-For-Tat and Spiteful. The results I find are similar.

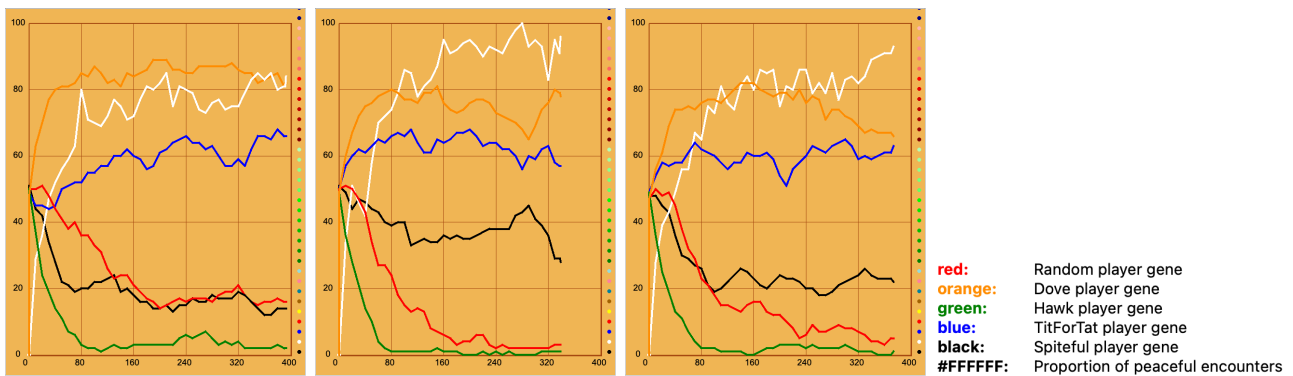


Figure 4: Results Ensemble Approach; PieToShare = 10; CombatCost = 40; NbEncounters = 7

## Bibliography

Philippe Mathieu, Jean-Paul Delahaye. New Winning Strategies for the Iterated Prisoner's Dilemma. Rafael Bordini, Edith Elkind, Gerhard Weiss, Pinar Yolum. 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), May 2015, Istanbul, Turkey. pp.1665-1666, 2015, Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015). <http://www.aamas2015.com/hal-01199149>

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)

Name: **Bilèl Benammar**

---

## Cellular Automaton's Game of Life

### **Abstract**

I attempted to implement John Conway's Game of Life in the Cellular Automaton of Evolife by going from a 1D Automaton to a 2D Automaton and using the right rule in it.

### **Problem**

As well as for the 1D Automaton, we are trying to observe an emerging phenomena using only simple rules and simple cells, but this time in a 2D environment. Emergence may express through exotic complex figures obtained but also through the creation of structures that move indefinitely on the grid, coming from static binary cells.

### **Method**

I first made a program which generates the specific rule number that corresponds to the set of rules of The Game of Life. Then, I modified the 1D Cellular Automaton program in order to make it process 2D elements and not simple values, and attempted to use a new observer in order to display an image through time. If I had been successful, I then would have used my new 2D Automaton to make it run the specific rule that I wanted and would have observed the characteristic structures obtainable in The Game of Life.

### **Results**

I got an application that could generate a rule and apply it to an already existent state. However, I did not manage to correctly display the states of the 2D space, thus I got no visual result.

## ***Discussion***

This program could be improved obviously by managing to code correctly the observer and the display of the 2D states, but I also could have used either Matplotlib or the observer of the 'Segregation' application to do it. I tried with both methods but did not manage to have a successful display because of a lack of time.

## ***Bibliography***

Conway's Game of Life ([Conway's Game of Life - Wikipedia](#))

Emergent Phenomena ([Telecom Paris - JL Dessalles - Artificial Intelligence \(enst.fr\)](#))



ATHENS course : **TP-09**

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)

Name: **Kacper Bugala**

---

## Simulating cellular automata with non-deterministic factor

### Abstract

The aim of this work was to analyse if the secret of the beauty in patterns generated by Cellular Automata algorithm lies in its immaculate math rules. In this project the impact of adding a non-deterministic factor was examined.

### Problem

Cellular Automata algorithm can generate infinite number of unique patterns depending on the rule and size of the neighborhood. Some of them are finite, while the others will keep spreading as long as the script is running. Noone can deny, that some of them are really interesting and eye-catching. But what is the secret of the beauty? Is it the mathematical and strictly deterministic rule of evolving new cells?

Problem of this work is to analyse the impact of adding some "evolutionary noise" on generated by the algorithm images and patterns. Comparing it to the-same-rule, but lacking probability algorithm can give answers, whether it will ruin patterns, develop images, remain unnoticed, or all.

### Method

In this project custom cellular automata algorithm was implemented using extended bit-space size of neighbours and with several new states for cells – not only whether it is dead or alive,

but also what shade of blue it is. The most important part of this script is using a single non-deterministic factor just to create some 'noise' on the output (and definitely not to kill the amazing idea of cellular automata itself). Once it was implemented, the results were compared to the outcome of similar algorithm with constant factor instead of the randomizing one.

All work was based on cellular automata implementation in The Evolife simulation software.

## Rule

Work was based on existing implementation, that used neighbourhood of size 3 to decide if the newly created cell will be dead or alive. Simple idea of what it was is represented in figure 1.

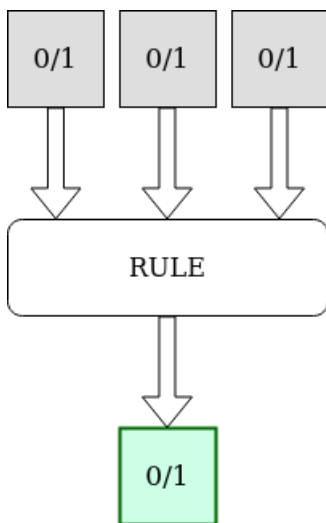


Figure 1: Default idea of Evolife cellular automata implementation

Three-bit value is evolving into dead or alive cell according to the rule: 8 possible input combinations, 2 possible outputs.

In this work's new approach, fourth input bit was added and placed on the least significant bit – this is the bit, that is designed to generate some signal noise. Placing it on the least significant bit will make it more intuitive, since it can change the human-known rule number only by 1. The value on this input is a copy of one of the original three neighbours, randomly chosen for each new cell.

For example, if the neighborhood is a bit sequence [011], then for each cell there is 33.(3)% chance of using [0110] input, and 66.(6)% chance of using [0111] input. Whole idea is visualized in figure 2. Using 4 input bits, 16 different combinations can be generated. In this case, two of them are cannot be reached ([0001] and [1110] are forbidden, since the fourth value is to be copied from the others), so the idea ends up with 14 possible inputs.

## Colour

Firstly, according to the rule, new cell is evolving into dead one or alive one. Once it is alive, colour is generated for the cell according to another rule. The rule is simple and uses 3 default input bits (the non-copied ones). Easy to count, there are 8 possible values of 3-bit number, so that corresponds to the shade of blue of the cell. For [000] it is the darkest blue, for [111] the lightest blue, all defined by the Evolife default colours.



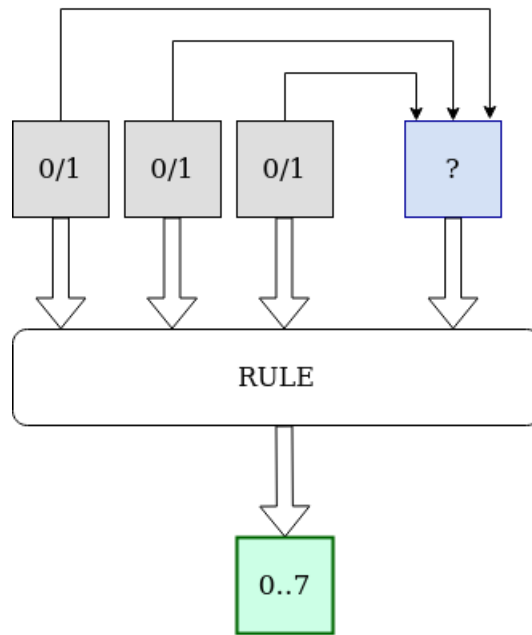


Figure 2: Custom idea of Evolife cellular automata implementation

## Results

To compare with deterministic output, the non-deterministic factor was changed to constant value of 0. Then the same *'rule'* was used to generate image with both algorithms and their patterns were compared and analyzed. For this paper, let's call them *'static 4-bit'* and *'random 4-bit'*.

As it was expected at the start, use of non-deterministic factor generated mostly either blank or fully filled images, just like the static 4-bit or default evolife implementation.

During the experiments, rule 420 was noticed as a very interesting one. For this rule, the static 4-bit automata generated very regular pattern, that many might find beautiful. When the same rule was run on the random 4-bit, all the magic has collapsed. It can be easily seen, that irregularities at the top of the pattern leads to complete lack of regularity at further steps – like a snowball effect. Still, the fact that the development of all three starting points differs is very interesting. Results of the rule 420 might be seen in figure 3.

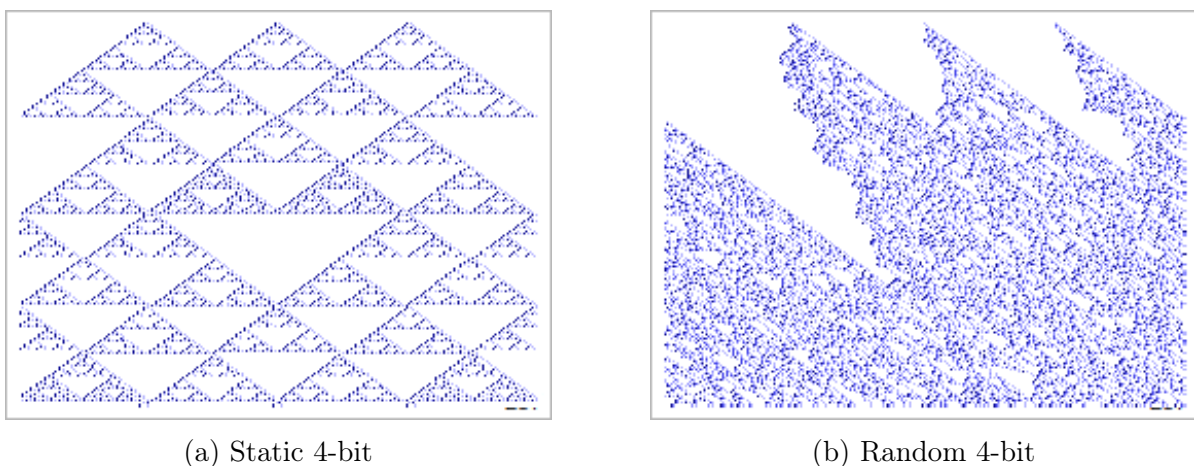


Figure 3: Rule 420

Later on, rule 502 was found. Unexpectedly it gave completely different results than the previous example. This time, the static 4-bit generates boring and bland image, mostly filled with same colour alive cells mostly non-regular inside. Meanwhile, the same rule used with random 4-bit gives us way more interesting image. The outcome is mostly regular, but with some easy to see irregularities. The regular part of the image is way more regular than the static 4-bit itself. The outcome of random 4-bit is somehow similar to some mountain landscapes, with irregular summits and rivers. This is very interesting and unexpected effect of the implemented algorithm. The effects may be seen in the figure 4

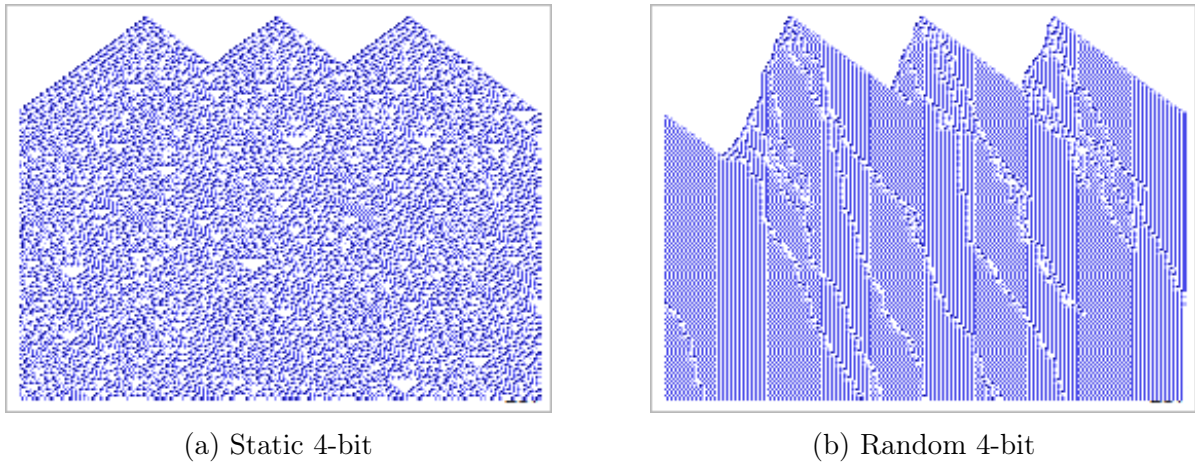


Figure 4: Rule 502

## Discussion

The expectation were that adding a non-deterministic factor may mostly destroy the beauty of this algorithm, and that it will not worsen the image at its best.

Mostly – it is true. For many rules, outcomes of the non-deterministic algorithm give us worse, or just as bad images, as did default algorithm implementation. But unexpectedly, during the experiments some of interesting rules were found – rules like the one specified in previous section, rule 502 (4). Apparently, there are environments, where using strict math rules is boring and using some of unexpected randomness may bring up the beauty. It is an example, where deterministic approach of cellular automata generates bald image and using non-deterministic factor makes it more interesting, more fascinating and more regular for a human eye.



That being said, this idea of using just a little randomness in this algorithm should be widely examined. There are infinite possibilities of inserting random factor into the image and the most beautiful images are still out there. There seem to be no limits for the exploration.

## Bibliography

<https://evolife.telecom-paris.fr/>

<https://www.mathpages.com/home/kmath439/kmath439.htm>

[https://www.whitman.edu/documents/Academics/Mathematics/SeniorProject\\_IanColeman.pdf](https://www.whitman.edu/documents/Academics/Mathematics/SeniorProject_IanColeman.pdf)

 	<p style="text-align: right;">December, 2023</p> <p style="text-align: center;"><b>Emergence in Complex Systems</b> Micro-study</p> <p style="text-align: right;"><a href="https://teaching.dessalles.fr/ECS">teaching.dessalles.fr/ECS</a></p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Name: **Cardot Aymeric**

---

## **How does humanity signal against a global emergency?**

### ***Abstract***

My study was about the emergence of signaling when a population is facing a global emergency such as climate changes. This study will be confronted with an extrapolation of what is going on now in the world.

### ***Problem***

Climate changes are today, one of the biggest emergencies that humans must face. To do that, we have, all together, to try to change our life habits. So, we need some people to signal themselves to show to other people it is possible to do it. But, and the main topic is here, the time is very limited and for everyone.

### ***Method***

My idea was to implement three big methods of signaling that we can observe now. For like famous people or institutions, asymmetrical bound method. For environmental associations, time sharing method. And for militant, “shame on you” method. The goal was to see what kind of signaling method would emerge in the population and which would be the better one (if better has a definition). I had many tests to try, for example:

- If all the population, try to signal, what would be the result? Or just a few of it?
- If signaling people would be in concurrence? Or not?
- If there were like “bad” signaling people who tried to signal against the other?
- What if some people would have been upset?

The main behavior of agents is to act like gregarious swallows to make the model explode if there are enough “convinced people”.

## **Results**

I don't have any result of this because it was more time consuming than what I have expected. I had many issues with the simulation of signaling and the interaction between each other. In my algorithm, nothing really happened because my agent seems to never interact. I don't use Evolife because I found difficult to implement more than two classes for my agent but maybe I have missed something.

## **Discussion**

There is a lot of limits in this simulation, especially the idea that it is not a simulation of real life. We can add many other mechanics like vote, clustering, etc.

## **Bibliography**

[https://www.researchgate.net/publication/348667760\\_Global\\_Lessons\\_from\\_Climate\\_Change\\_Legislation\\_and\\_Litigation](https://www.researchgate.net/publication/348667760_Global_Lessons_from_Climate_Change_Legislation_and_Litigation)

Jorge M. Pacheco, Vítor V. Vasconcelos, Francisco C. Santos,  
Climate change governance, cooperation, and self-organization,  
Physics of Life Reviews,  
Volume 11, Issue 4,  
2014,  
Pages 573-586,  
ISSN 1571-0645,  
<https://doi.org/10.1016/j.plrev.2014.02.003>.



November, 2023

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](https://teaching.dessalles.fr/ECS)

Name: **Corentin Caris**

---

## Analysis of Schelling's Segregationism using phase transition in the Ising model

### **Abstract**

I showed that Schelling's model of segregationism is subject to continuous phase transition when adding a dynamism parameter that urges individuals to move out even when satisfied of their neighbours.

### **Problem**

I was inspired by the evident similarity between Schelling's model and the Ising model for ferromagnetism I encountered in a Statistical Physics class. The temperature increases the probability of a particle to change spin without considering the average spin of its neighbors.

At low temperatures the model predicts a distribution of spins that reduces the energy and thus harmonizes the spins. At high temperatures the model predicts a random distribution of spins. When increasing temperature, the average absolute magnetization transitions from 1 to 0 and changes phase.

### **Method**

I added a Dynamism parameter and tried to observe the same transition phase from segregated neighbourhoods to a random distribution.

I then tried to implement the Ising model directly in the file but the current definition of the population and the groups through the modules of Evolife made it difficult, my tentatives are in the code (Ising.py). The last difficulty I encountered was that I could change colors of individuals but not change their group in the population, so the fonctions already implemented did not work.

## **Results**



I found out that it is possible to observe such a phase transition and the critical state is found for a dynamism value of around 15% with a tolerance at 40%.

## **Discussion**

In conclusion, while we can observe segregation even if the tolerance is at 40%, having a relatively small proportion of individuals that move out randomly and ignore the neighbourhood can help reduce segregation.

## **Bibliography**

- <https://www.bdhammel.com/ising-model/>
- <https://rf.mokslasplius.lt/ising-model/>

 	<p style="text-align: right;">December, 2023</p> <p style="text-align: center;"><b>Emergence in Complex Systems</b> Micro-study</p> <p style="text-align: right;"><a href="https://teaching.dessalles.fr/ECS">teaching.dessalles.fr/ECS</a></p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Name: **Diogo Castelo Cardoso**

---

## Segregationism: real life adaptations

### ***Abstract***

For my project, I made some alterations to the initial model of segregationism. I made it so each population would have a random density, each agent would have variability in his tolerance, and agents could move for reasons unrelated to tolerance.

### ***Problem***

The Thomas Schelling's model has a very negative result when it comes to representing the issue of segregationism which showcases how different individual decisions on where to live produce neighborhoods segregated by race.

I wanted to see if adding some parameters to the original model, making it more realistic, would cause a more positive outlook or if the results were maintained.

### ***Method***

To reach my objective, I first changed the initialization of the class Population in the file Population.py so that the initial state of the problem would have random densities of different groups.

Secondly, I added the constant TOLERANCE\_VARIABILITY so that when an agent is tested on whether to move or not, he will be more or less tolerable to  $n$  neighbors (in the delivered file, it is considered a radius of 1 and an  $n$  of 1). The formula is  $(n/nb)*100$  where  $n$  is the number of neighbors intended to be more or less tolerable towards and  $nb$  the number of neighbors considered for a given radius. This is used in the satisfaction function.

Lastly, I added the constant MOVE\_FOR\_OTHER\_REASON, whose value decides the percentage of agents that will move for reasons unrelated to tolerance. It is considered that

these agents will move to a random location and stay there until the end of the simulation. For this, I created the global variable `N_moved` and added the parameter `specialReason` to the class `Individual` as auxiliaries.

## ***Results***

With the implemented changes, there were some differences in outcomes. For higher tolerance variability, the resulting grid was less segregated. The same is applicable for a higher percentage of agents moving for reasons unrelated to tolerance.



These positive results are especially visible when the densities of each population are more evenly distributed.

## ***Discussion***

With the changes made, I expected that they would have a big impact, ending in much less segregated grids, especially for high `MOVE_FOR_OTHER_REASON` values, and although I expected a bigger impact than the results, you can clearly see some huge differences when compared to the results of the original model.

For example, when considering a tolerance level of 40, you can start to see some relevant changes around a `MOVE_FOR_OTHER_REASON` of 30, where there are a lot more neighborhoods of each population, and you can see some outliers inside each segregated neighborhood. If you increase `MOVE_FOR_OTHER_REASON` to 50, there is almost no visible segregationism.



 	<p style="text-align: right;">November, 2023</p> <p style="text-align: center;"><b>Emergence in Complex Systems</b> Micro-study</p> <p style="text-align: right;"><a href="https://teaching.dessalles.fr/ECS">teaching.dessalles.fr/ECS</a></p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Name: **Katia Chardon**

---

---

## A more realistic ant simulation

### ***Abstract***

During the course, we have learned about emerging behaviors and how collective behaviors emerge from the relationship between the individuals. The collective behavior can't be predicted by looking at each individual's behavior but only by looking at the whole.

### ***Problem***

To dig further, I chose the ant simulation present on Evolife. I wanted to add more realistic functions to it to see if the behavior of the ants would be different. In the simulation, each ant follows irregular paths to find food and generally the collective discover optimal paths and go to the closest food first. I decided to add renewable food, variable quantity of available food at food sources, evaporation of pheromones proportionally to the local quantity and diffusion to neighboring cells (proportionally to the local quantity). The question was "What will be the collective behavior with these adds ?"

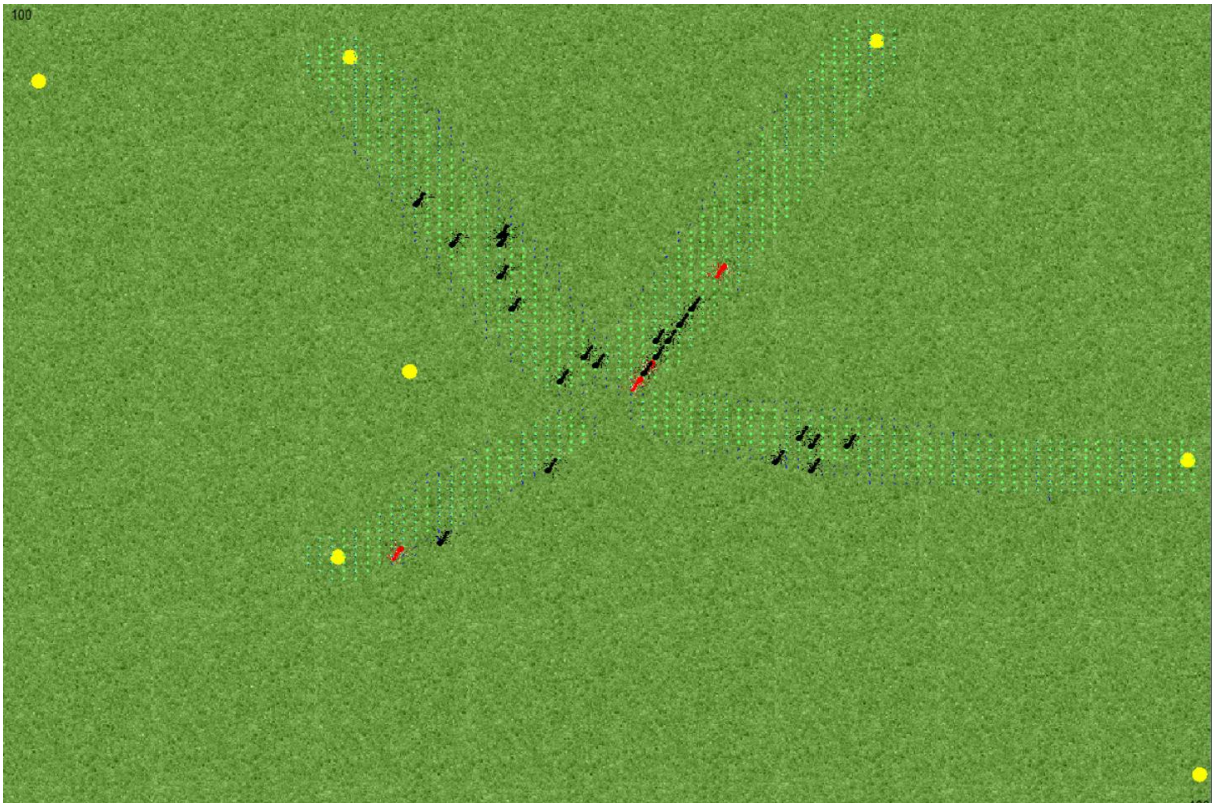
### ***Method***

To do that, I modified the 'Ants.py' file. For the variable quantity of available food, I modified the line to fill the food sources with a random quantity between 10 and the global parameter FoodQuantity. For the renewable food, I added a counter in the FoodSource class that count the number of moves made from the moment the food source reached 0. After 10000 moves, the food source is refilled with a random amount of food between 10 and FoodQuantity. For the evaporation of pheromones proportionally to the local quantity, I modified the 'evaporate' function in the LandCell class. The global parameter Evaporation is the percentage of quantity of pheromones that evaporates at each move. When the quantity of pheromones is lower or equal to 10, I put it to 0. For the diffusion of pheromones to neighboring cells, I modified the functions 'npheromone' and 'ppheromone' in the Landscape

class. When an ant drops pheromones on a cell, the pheromones diffuse to the neighbor cells in a 2\*2 square proportionally to the distance.

## **Results**

I observed that with the diffusion of pheromones, the paths to the food sources became bigger and so the ants don't go straight to the food source but rather moves randomly on the path; sometimes going straight forward to the food, sometimes going back, sometimes going to the left or the right.



I also saw that with the renewable food, the ants first eat all the food from the closest source and then go to farthest sources. My intuition was that, since some of the ants explore while the other bring food home, they would see when the closest source is refilled and go to it. Sometimes, they did this. However, with the bigger paths and the negative pheromones around the paths, the ants are sometimes stuck inside the paths. Since they are stuck, no ants explore and thus they don't go to closest food source. They first finish the farthest food sources and then go to the closest sources.

## **Discussion**

The results were a bit different than what I expected and it could maybe be resolved by changing my functions. One modification that I could make is with the diffusion of pheromones. I could make each cell containing some pheromones diffuse to the ones around. This could create a new emerging phenomena and maybe solve the problem of the ants being stuck.

November, 2023



ATHENS course : TPT-09

# Social Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/SECS](http://teaching.dessalles.fr/SECS)

Author: Michał Ciasnocha

## Rock-Paper-Scissors Cellular Automata

November 27, 2023

### Abstract

Everyone knows about Rock-Paper-Scissors game from their childhood. You can play it mindlessly, guess what your opponent might do, or actually think about a strategy. What if we create a strategy - a set of rules and build a 2D cellular automata around it?

### Problem

This micro-study is a modification of a well known Conway's Game Of Life [1]. We want to expand the set of possibilities a cell can have, from binary states  $\{0, 1\}$ , to any number of states  $\{0, 1, \dots, n - 1\}$ . Because of this, we should widen the rule set, defining additional interactions between cells of different states. And here comes the rock-paper-scissors idea: if a cell is surrounded by a group of cells that 'beat' it, we will change the state of this cell's state to the winners' state. The full interaction set is roughly as follows: In a game with  $n$  states, cell of state  $i$  beats cells of states  $\{(i + 1)\%n, (i + 2)\%n, \dots, (i + \frac{n}{2})\%n\}$  and gets beaten by the rest of them, except for itself. If  $n == 3$  we arrive at a regular rock-paper-scissors game. This will hopefully create interesting patterns with many oscillations, as cells with cyclically swap states with each others.

### Method

I implemented a solution both in *C* and *Python*. The *C* solution is without a doubt faster and allows for much bigger simulations which are significantly more attractive. Here I provide all the

important code with comments that should be easy to follow:

```
// mnca - present states for each state
// counts - contains counts of each possible state, for every cell's neighbourhood
int compute(int* mnca, int** counts, int width, int height){
    const int radius = RADIUS;
    // iterate over each cell
    for(int i = 0; i < height; i++){
        for(int j = 0; j < width; j++){
            // delete previous state information
            for(int s = 0; s < STATES; s++){
                counts[s][i * width + j] = 0;
            }
            // iterate over each neighbour
            for(int ii = -radius; ii <= radius; ii++){
                for(int jj = -radius; jj <= radius; jj++){
                    const int i_index = (i + ii + height) % height;
                    const int j_index = (j + jj + width) % width;
                    // count neighbours with each state
                    counts[mnca[i_index * width + j_index]][i * width + j]++;
                }
            }
        }
    }
    for(int i = 0; i < height; i++){
        for(int j = 0; j < width; j++){
            float ps[STATES];
            for(int s = 0; s < STATES; s++){
                // calculate occupancy by every state
                ps[s] = (float)counts[s][i * width + j] / ((2 * radius + 1) * (2 * radius + 1));
            }
            for(int s = 1; s <= STATES / 2; s++){
                // change state if the condition is met
                if(ps[(mnca[i * width + j] + s) % STATES] > THRESHOLD){
                    mnca[i * width + j] = (mnca[i * width + j] + s) % STATES;
                    break;
                }
            }
        }
    }
}
```

Figure 1: Compute function in  $C$  implementation

## Results

We can observe very interesting behaviour:

1. In the beginning we can see many rotating spirals form. These are triplets which resemble our rock-paper-scissors idea.
2. At some point various oscillating structures form, which are slowly 'eating' everything around.
3. When all the spirals disappear, we find ourselves at 'war' between remaining oscillating structures.
4. The equilibrium is only achieved when just a single structure wins all the terrain (with some exceptions.)





Figure 2: Many spirals forming.

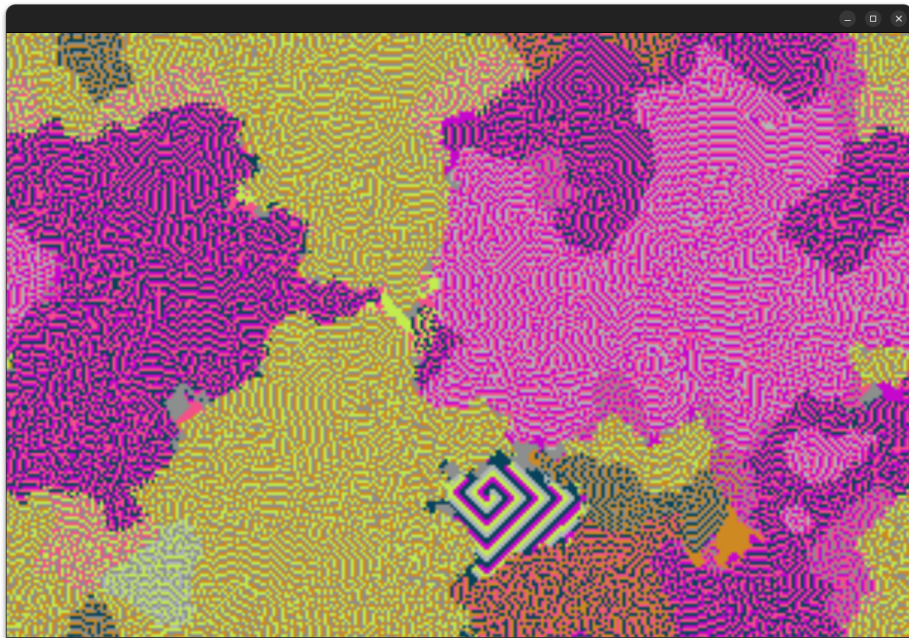


Figure 3: Oscillating structures emerge and 'eat' nearby spirals.



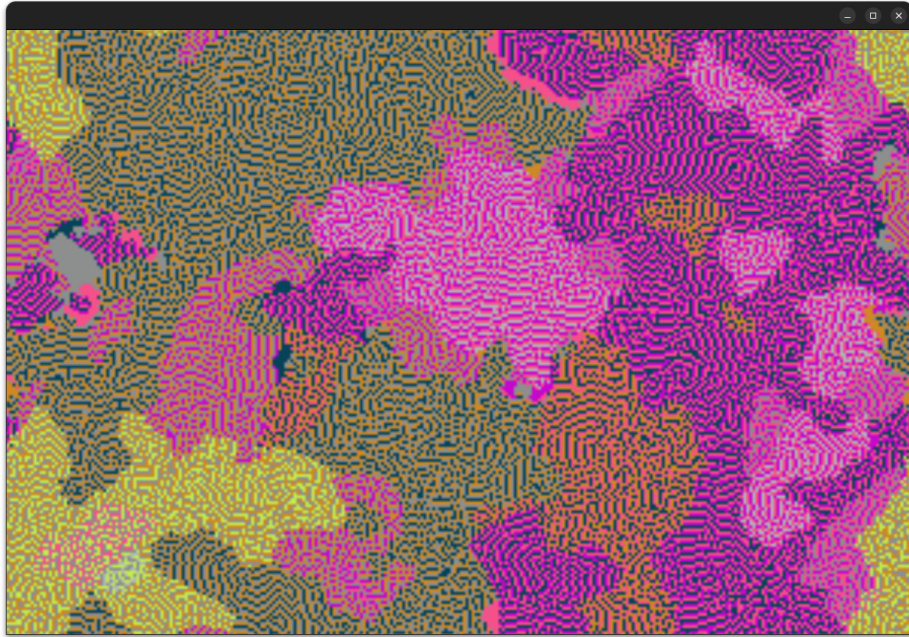


Figure 4: The 'war' for supremacy.

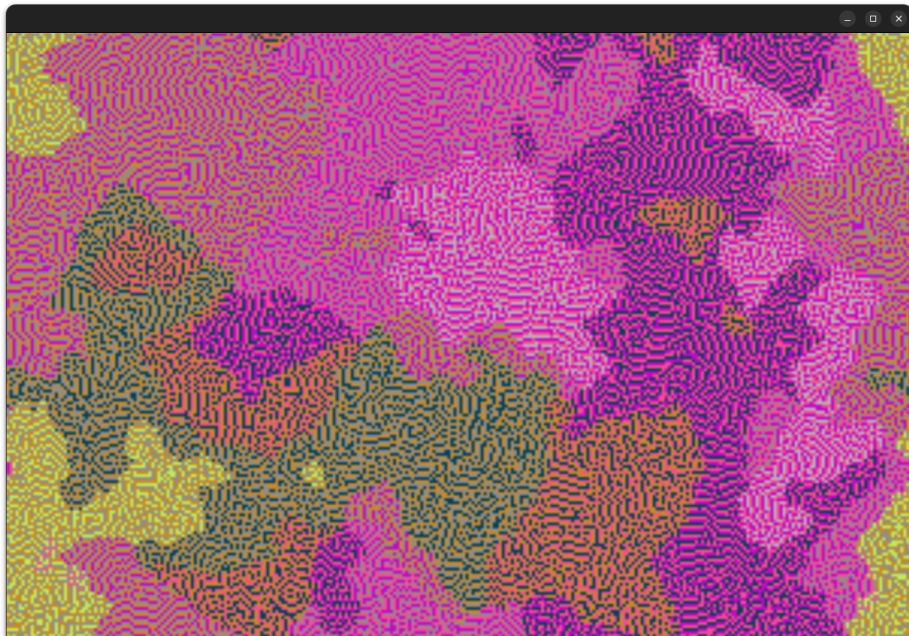


Figure 5: An equilibrium is slowly being approached.

An also interesting behaviour appears as we increase the neighbourhood radius. Here we experience spirals again, this time, more circular. They remain intact, but now it is a fight for 3 best colors. It appears as only 3 colors reach an equilibrium in every scenario.

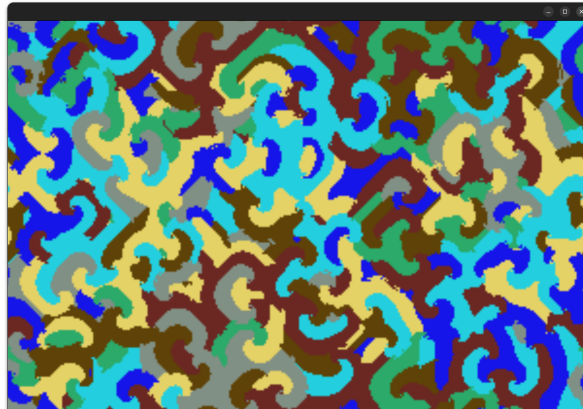


Figure 6: Initial spirals are formed.

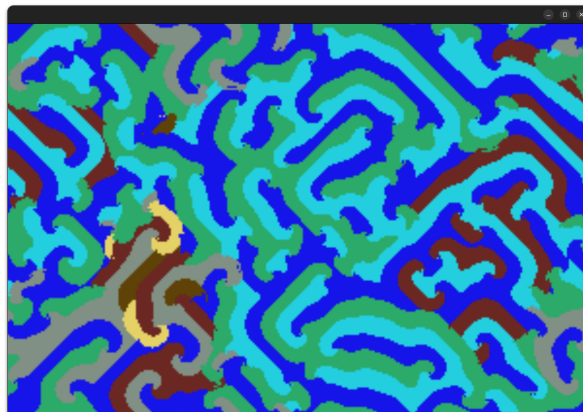


Figure 7: The 'weakest' colors are being pushed out.

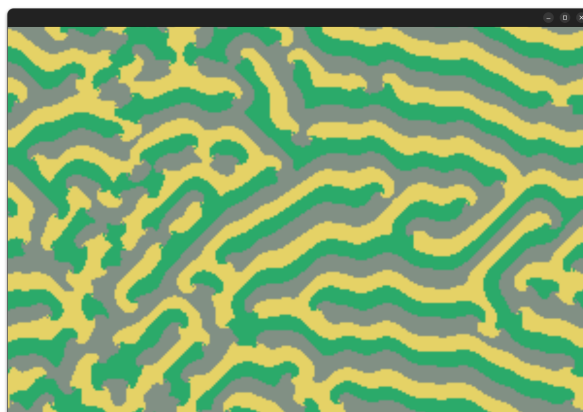


Figure 8: Only 3 colors remain in the end.

## Discussion

The results were not disappointing: The emergence of spirals, oscillators and a fight between them was not something I expected from such a simple rule set. It would be very interesting to see what similar variations of this cellular automata might offer. Third dimension, continuous instead of discrete states or maybe different shapes of neighbourhoods, these are interesting ideas to explore.

## References

- [1] Mathematical Games. The fantastic combinations of john conway's new solitaire game "life" by martin gardner. *Scientific American*, 223:120–123, 1970.



November, 2023



ATHENS course : TPT-09

# Social Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/SECS](http://teaching.dessalles.fr/SECS)

Name: Florine LEFER, Mathieu CLAVE

---

## Multiple Choice Hawk-Dove Problem

November 28, 2023

### Abstract

#### Abstract

The Hawk-Dove problem is a fairly simple simulation, where individuals have two choices only, sharing or stealing. Our work implements a denser representation of reality, where multiple actions are possible, and our goal is to see if some sort of global peace can still emerge.

### Problem

The Hawk-Dove problem is very limited, and often leads to an all-out civil war, when the rules are not chosen to heavily favor the doves. The only way to have peace is for the game master to enforce it. Moreover, as it lacks in complexity, it fails to capture depth in the social interactions, in particular strategies and mind games. In playing the standard game, there are two cases to consider. If I think that my partner/opponent is a naive optimistic pacifist, I can take advantage of him by forgetting peace and trying to bully. In that case, we see how knowing my opponent leads me to victory. But in the other hand, if I know that my partner/opponent is a nervous hot headed maniac, and that he will most probably try to steal, what should I do ? Truth is in this standard game there is justice, no law and order, and no way to protect yourself. These two ideas seemed to us very far from the reality, where knowing what my opponent will play always lead to flawless victory, and stealing does not feel so dangerous. This is the reason for our work, where we tried to implement an extended version of the game, following these principles.

# Method

The first work was to define the different strategies. We wanted a somewhat balanced game where anticipating is rewarded, and being a good citizen is globally beneficial individually and for the society. We decided to keep the stealing (hawk) and sharing (dove) as they were, and adding complexity. We came up with three additional actions that would be best understood using the boxing metaphor :

- **Focus** : not throwing the first punch, but watching closely the opponent. Designed to punish stealing, it is overall weaker than sharing but is the best choice when encountering a fearsome individual.

- **Feint** : moving the shoulders to fake throwing a punch in order to create a reaction, and trying to capitalize on it. Designed as a balancing tool for the Focus strategy, it tries to prevent exploitation of other, but still keeping an open mind about people. - **Leave** : fleeing the fight and hoping for future compensation. Equivalent to Poker folding.

This leads to the following table that defines the rules. It should be read as such :

	my opponents' action
my action	the reward I get

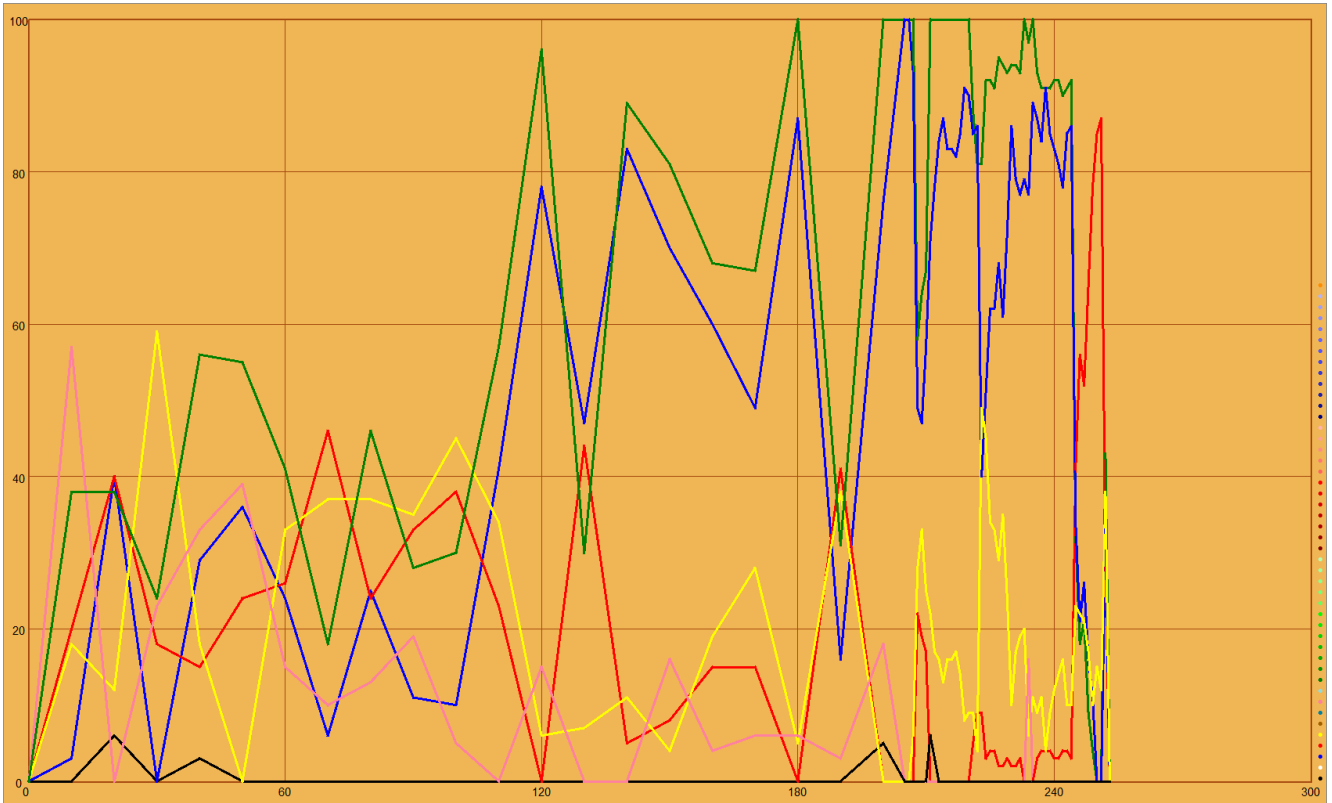
$V$  is the pie to share, the total reward, and  $C$  is the cost for battle.

	Steal	Share	Focus	Feint	Leave
Steal	$\frac{V-C}{2}$	$V$	$0$	$V$	$V$
Share	$0$	$\frac{V}{2}$	$\frac{V}{2}$	$\frac{V}{2}$	$\frac{V}{2}$
Focus	$V$	$\frac{V}{2}$	$0$	$0$	$\frac{V}{2}$
Feint	$0$	$\frac{V}{2}$	$V$	$0$	$\frac{V}{2}$
Leave	$0$	$0$	$0$	$0$	$0$

To implement the game, we used a dictionary to store the table, and we imagined a population of five deterministic species. Every individual always tries the same action, but the number of individual in each species vary according to the rewards of the individuals. We also wanted mutations to happen, individuals to change species. We then had to make sure the species were encoded in a way that made every distance the same. That is why we used Grey encoding, the same way than with the Heteroclyne Cycle. All that was left was to count the peaceful encounters, which we decided would be all the encounters, except for when someones tries to steal or there is a feint/focus conflict.

# Results

We can see that with  $V = 10$  and  $C = 8$ , we do have general peace!



### Legend

Curves:

- green:** proportion of peaceful encounters
- red:** Proportion of steal tries
- blue:** Proportion of share tries
- yellow:** Proportion of focus tries
- pink:** Proportion of feint tries
- black:** Proportion of leave tries





# FINAL PROJECT

## EMERGENCY IN COMPLEX SYSTEMS

---

### Project 1 – Mandelbrot and Julia Fractals Sets

---

**Autores:**

Larissa Da Silva Montefusco  
João Melga

[larissa.montefusco7@gmail.com](mailto:larissa.montefusco7@gmail.com)  
[joaolucasfm@gmail.com](mailto:joaolucasfm@gmail.com)

**Emergence of fractals**

2023/2024 – 1st Semester

# 1 Abstract

A fractal is a mathematical shape that exhibits self-similarity on all scales. This means that if you zoom in on a fractal, you will see the same pattern repeated over and over again, at smaller and smaller scales. Fractals are often very intricate and detailed, and they can be found in nature as well as in mathematics. Therefore, our chosen final project for the "Emergency in Complex Systems" course centers on the implementation of two distinct fractals, Mandelbrot and Julia sets, using the Evolife library. This report will provide comprehensive explanations of both the mathematical principles and the developed algorithms that underlie the chosen fractals.

## 2 Problem

### 2.1 Mandelbrot Set

The Mandelbrot set is a collection of complex numbers that exhibits fascinating fractal properties. It is generated through the repeated application of the formula

$$z_{n+1} = z_n^2 + C, \quad z_0 = 0 \text{ and } z, C \in \mathbb{C} \quad (1)$$

to each complex number. If this sequence converges, then the complex number 'C' is in the Mandelbrot set.

This complex set possesses several interesting properties, such as:

- **Self-Similarity:** The Mandelbrot Set exhibits self-similarity at different scales. Zooming into any part of the set reveals structures similar to the overall set, a characteristic common in fractals.
- **Infinite Complexity:** The boundary of the Mandelbrot Set is infinitely complex. No matter how much you zoom in, there are always intricate patterns and details to explore, making it a fractal with infinite complexity.
- **Invariant under Conformal Mapping:** The Mandelbrot Set is invariant under conformal mappings, meaning that certain types of transformations won't change its overall structure. This is a significant property in complex analysis.

### 2.2 Julia Sets

Julia sets constitute a class of mathematical sets associated with the iteration of complex functions. Named after the French mathematician Gaston Julia, who extensively studied and popularized their analysis in the early 20th century, Julia sets are renowned for their visual beauty and structural complexity. Defined through the iteration of a simple complex function on points in the complex plane, Julia sets often exhibit intricate patterns and fascinating details. These sets are directly related to the Mandelbrot set, since both are expressed by the same mathematical formula.

In the context of Julia sets, the terms *connected* and *unconnected* refer to the geometric properties of the sets and their relationship with the complex parameter 'C' in the defining

function  $z_{n+1} = z_n^2 + C$ . A Julia set is considered "connected" if, for a particular 'C' value, all points in the set are part of a single, connected component. In other words, there are no isolated or disjoint components within the set. The set forms a continuous, unbroken structure. On the other hand, an unconnected Julia set has multiple disconnected components. This means that there are isolated regions or islands within the set that are not connected to the main body of the set. The presence of disconnected components in a Julia set is often linked to the instability of the iterative process. If points in the complex plane diverge to infinity or exhibit chaotic behavior, it can result in the formation of unconnected components.

## 3 Method

### 3.1 Mandelbrot Set

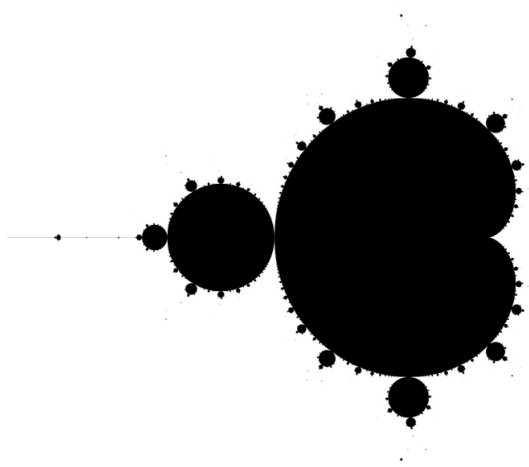
The first objective of this project was to create an algorithm using the Evolife library for visualizing and plotting every element within the Mandelbrot Set. To accomplish this, the *Mandelbrot.py* code was developed. In this code, a random complex number C within the range of -1 to 1 is selected, and it undergoes a verification process to determine if the sequence converges based on the function `is_stable()`. If the sequence converges, indicating that the selected number belongs to the Mandelbrot set, it is then plotted.

### 3.2 Julia Sets

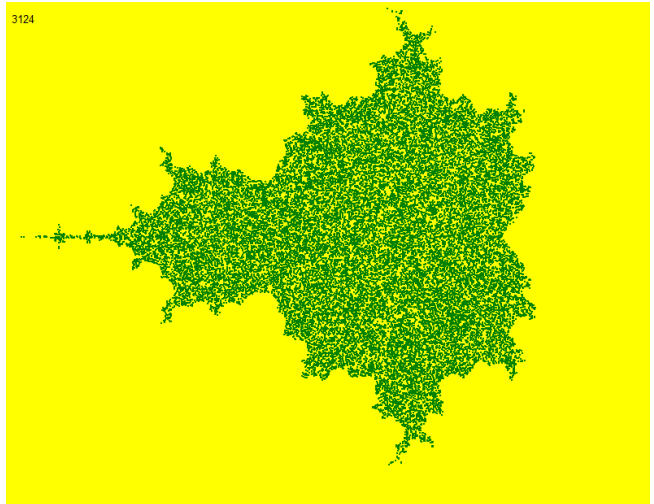
We have developed an algorithm utilizing the Evolife library to visualize Julia sets *Julia.py*. In essence, the algorithm randomly selects a complex number 'C' from the Mandelbrot set, confined within the range of -1 to 1. It then tests a series of random complex numbers 'z' to determine if the iterative sequence converges (`is_c_stable()` and `is_Z_stable()` methods). If convergence is achieved, the corresponding 'z' value is deemed to belong to the Julia set defined by the complex number 'C' so then its value is plotted by the `addDot()` method.

## 4 Results

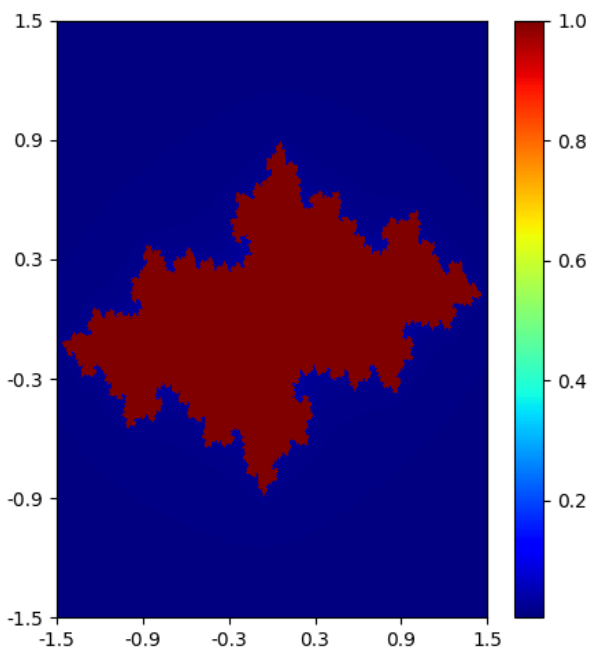
The results for both *Mandelbrot.py* and *Julia.py* can be seen in the images below.



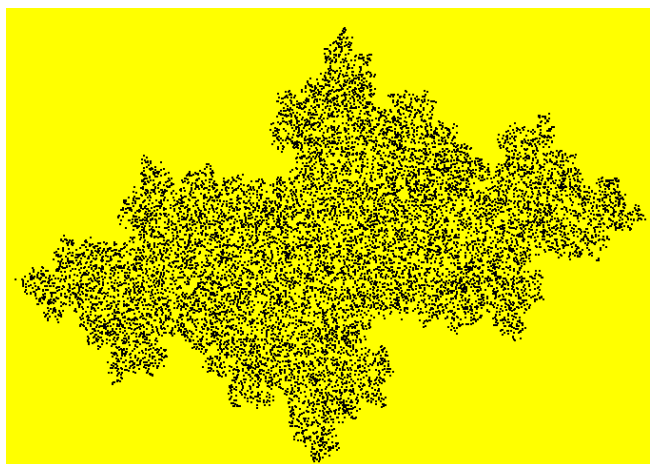
(a) Mandelbrot Set



(b) Image plotted by Mandelbrot.py

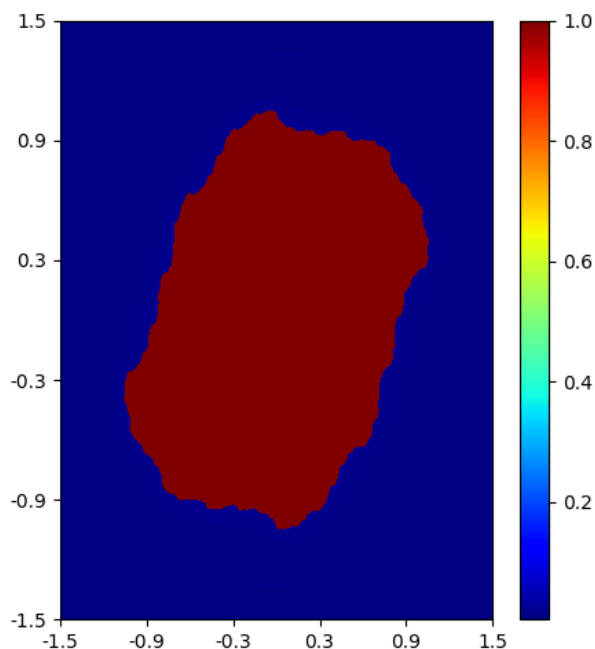
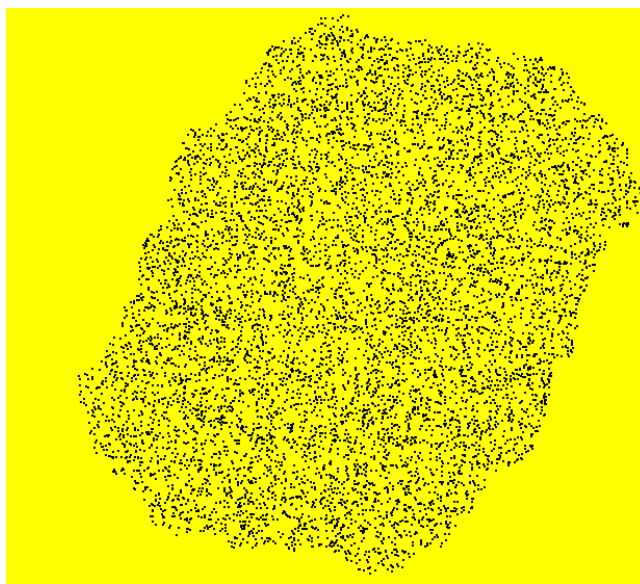


(a) Julia set for a complex number  $C = -0.673 - 0.229j$



(b) Image plotted by Julia.py for a complex number  $C = -0.673 - 0.229j$



(a) Julia set for a complex number  $C = 0.042 - 0.27j$ 

(b) Image plotted by Julia.py for a complex number

## 5 Conclusion

In conclusion, the performance of both programs within the Evolife library has met our expectations, delivering satisfactory results. Some future improvements could be done, such as providing users with the ability to manually select the parameter 'C' through Evolife's parameter configuration editor would offer a valuable customization option. This empowers users to tailor the program to their specific needs, fostering a more user-centric experience. Additionally, in the context of the Julia set plot, incorporating a broader spectrum of colors and expanding the color palette used in the plot can enhance visual representation would provide a more vivid and informative display of the results. This addition not only serves an aesthetic purpose but also contributes to a more comprehensive interpretation of the Julia set.

## Referências

- [Bar88] Michael F. Barnsley. *Fractals Everywhere*. 1988.
- [Num19] Numberphile. *What's so special about the Mandelbrot Set?* 2019.
- [gui20] The mathemagicians guide. *Python reference manual*. 2020.
- [Zac22] Bartosz Zaczyński. "Draw the Mandelbrot Set in Python". Em: (2022).





November, 2023

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](https://teaching.dessalles.fr/ECS)

Name: **DALMARD Alban**

---

## **Robustness of the Efficient Market Hypothesis to constraints relaxation**

### ***Abstract***

I use a multi-agents simulation with the EMH hypothesis to see the emergence of the emergence of the maximisation of the utility. We then relax hypothesis in our model to see how it modifies the emerging properties

### ***Problem***

The EMH is a recurrent hypothesis in policy recommendation,, especially to deregulate markets. Though it is based on obviously false hypotheses such as the rationality of the agents. The question of the application domain on which the EMH still applies is very important to know the way we should organise a market.

### ***Method***

We use a multi agents simulation to simulate a two market economy following the hypothesis of the EMH. There is one market of labour and one of products. The production is proportional to the labour and one of the products. The agents try to maximise their utility based on their preferences by working and buying products. Their marginal utilities are decreasing and the utility of labour is negative. We obtain a curve of the global utility over the time at each iteration of the model. We then try to alleviate constraints by adding white noise to the decisions of agents. We then draw the new curve which we can compare to the first one, depending on the size of the noise.

### ***Results***

I didn't manage to complete the simulation

## ***Discussion***

Comparison with expectations, limits, perspectives.

## ***Bibliography***

William F. Sharpe. "Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk." *Journal of Finance*, 19 (September, 1964), 425-42.



# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)

Name: **Rémy Kristen de Thomassin de Montbel**

---

## Desire Paths

### Abstract

Desire paths are a well-known phenomenon by urbanists, they are unplanned trails created as a consequence of erosion caused by human or animal traffic. They typically appear where infrastructure is incorrectly planned or non-existent. Just like ants, humans are able to find a path to reach their destination more optimally. The human reality is a little more complicated as it is also a social matter since not every human is going to walk on the grass if it is still green because of their civic behaviour. Most of us walk on desire paths with no regrets as there is nothing to further destroy. In short, when there is still green grass it is less likely that someone will explore that path possibility.



Figure 1: Desire path at the Ohio State University

# Problem

What we are looking for in this study is to obtain an approximate model that describes the creation of desire paths. How to include the social aspect of it in the model? What emergent phenomena are linked with desire paths?

# Method

To do this study I chose to use the "Path Finder" app on Evolife as a base for my simulation. This app looks a lot like the "Ant" app since both use the so-called pheromones. But the Path finder seemed more suitable for the simulation I wanted to do.

In short, what I did was to make an analogy between several parameters on the "Path Finder" app and the simulation I wanted to make. The analogies are the following:

Pheromones ↔ Grass erosion  
Altitude ↔ Grass presence  
Slope aversion ↔ Civicism

What this means is that agents will prefer to stay in a grassless path (low altitude path) and will only change to a grassed path (higher altitude path) according to their civicism (slope aversion). The erosion is represented by pheromones. When an agent decides to walk on a grassed surface, it will leave pheromones behind. Why pheromones? As mentioned in the abstract, people will tend to avoid stepping on grass but will probably not hesitate to use a desire path. It is therefore needed to make these paths somewhat attractive to counteract the negative impact that has the altitude on the agent's decision-making.

The changes in the program are simple:

- pheromones are now only emitted if the agents are above a minimal altitude (which is equal to the landscape's minimal altitude added to a constant set in the parameters called **grass threshold**).
- agent will only move from the "spawning" point to their destination and not going back (they will now be teleported) therefore there is no longer a pheromone mechanic in which the fastest path has the most pheromones
- to make agents more evenly spaced between them (to make a more realistic simulation as humans do not walk close to strangers), agents will "spawn" with a random delay (the max delay possible can be set in the parameters).
- to delve further into the study it was also implemented to have a variable civicism (which is more close to reality as not everyone will react the same way upon encountering a grassed surface). The **civicism** parameter and **sigma** parameter will respectively determine the mean civicism among the population and the standard deviation of the Gaussian curve that gives the probability law followed by the individual's civicism value.

# Results

The results were, to a certain point, as expected. The behaviour generated by the civicism parameter is a central part of the study. A very high civicism value gives a population that will always follow the grassless path. There is no emergence of desire paths (see Fig.2). A very low value of civicism will give a population that will only cross the grassed surface to reach

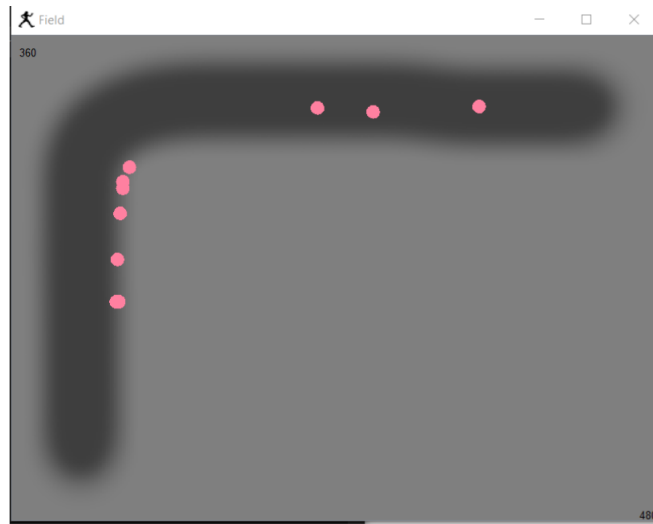


Figure 2: DesirePaths app simulation using  $\text{civicism} = 80$

their destination (see Fig.3). Note that this behaviour is instantaneous, from the beginning to the every end, each and every agent will use this path. Even if the result in Figure 3 seem close to reality, the chronological aspect of the simulation is implausible. When the  $\text{civicism}$



Figure 3: DesirePaths app simulation using  $\text{civicism} = 10$

parameter is in a certain interval, high but not very high, the simulation gives strange desire paths (see Fig.4).

This is one of the motivations to have a variable  $\text{civicism}$ : it would be great to be able to see the desire path formation progressively but if the  $\text{civicism}$  is too low it is immediate and if it is too high it is strange or non-existent. The following results are using a variable  $\text{civicism}$  so a  $\sigma$  value different from 0. With this new configuration we can have a high  $\text{civicism}$  (mean value) and yet have desire paths emerging.

I would also like to add the more friendly interface I made in the scenario `RealPaths.evo` in which the displayed image is a real park with desire paths and the information of altitude is in a separate black and white image (see Fig.5).



Figure 4: DesirePaths app simulation using  $\text{civicism} = 65$



Figure 5: More visually friendly simulation

## Discussion

The first configuration (with constant  $\text{civicism}$ ) gave results that were expected and that helped me realize that another model was needed. With the variable  $\text{civicism}$  i never fully achieved to have a progressive formation of desire paths. I lacked time to perfect this model.

I finally realized that a final study showing another complex phenomenon appearing could have been made. This study would consist in decreasing the  $\text{civicism}$  value (mean value in case the variable setting is chosen) of all agents lineally and eventually observe a sudden growth of agents taking the desire path instead of staying on the grassless path (just like in the birds simulation in which increasing lineally the probability of landing on the cable generates a sudden growth in the amount of landed birds).

## Bibliography

Desire path, [wikipedia.org](https://en.wikipedia.org) (last visited Nov. 28, 2023).

The illicit trails that defy urban planners, [theguardian.com](https://www.theguardian.com) (Oct. 5, 2018).

Desire Paths, Urban Planning, and their Impacts on UI Design, [jjbrowndesign.com](https://www.jjbrowndesign.com) (Jan. 22, 2022).



ATHENS course : **TP-09**

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)Name: **Donna Hooft and Sumukha Shridhar**

## Modeling the Wolf-Sheep population dynamics in Evolife

### Abstract

The wolf-sheep predator-prey model explores the cyclical interactions between wolf and sheep populations. This project implemented a simplified version of the model in the Evolife software to visualize these dynamics. The model omits the grass/resource factor and focuses on the relationship between wolves (predators) and sheep (prey). Results demonstrate the characteristic oscillatory pattern of predator and prey populations seen in ecological models. The Evolife model qualitatively reproduces the dynamics of classic predator-prey models.

### Problem

The wolf-sheep-grass model is a classic ecological simulation representing the dynamic interaction between predators (wolves), prey (sheep), and resources (grass). It explores how changes in one population influence the others over time, creating cyclic patterns of rise and fall (Wilensky and Reisman, 2006).

Within the model, the Wolf, Sheep, and Grass act as agents. Wolves negatively influence the amount of sheep, and the amount of sheep positively influences the amount of Wolves, as does Grass for Sheep. If there is too little Grass, the population of Sheep decreases, which ultimately leads to the population of wolves decreasing. The effects of the different populations on each other have a delay, often resulting in the extinction of one Agent.

In Evolife (Dessalles, n.d.), several population dynamics Scenario's are included, however, a similar model such as the Wolf-Sheep-Grass population dynamics is yet to be implemented. In this project, we propose the implementation of the wolf-sheep model as a scenario in

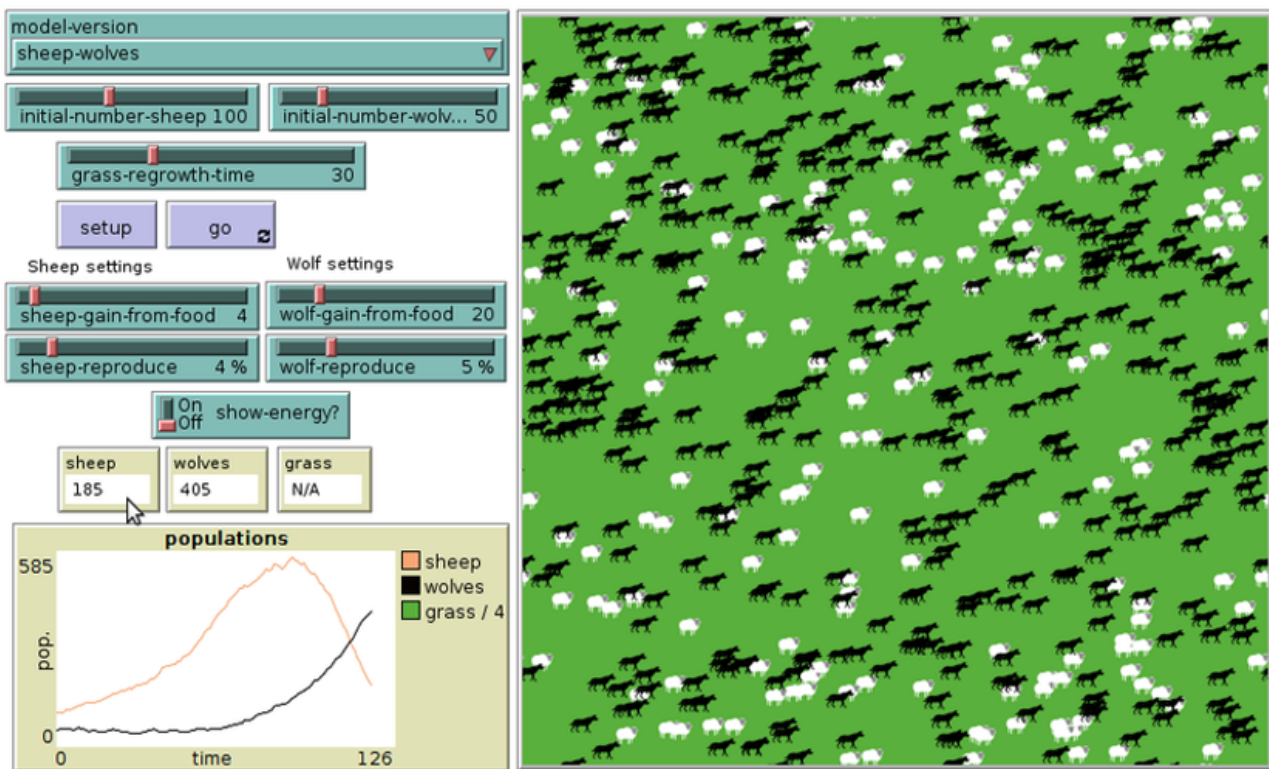


Figure 1: Image of NetLogo WolfSheep Predation model, image from “NetLogo Wolf Sheep Predation model - EduTech Wiki”, n.d.

the Evolife software. Our focus is on capturing the essential dynamics of the wolf-sheep interaction within a simplified ecosystem, omitting the grass factor. The goal is to visualize the cyclical interactions between the wolf and sheep populations.

## Method

In previous population modeling classes, Sumukha, has already worked on Predator-Prey-Resource models. Therefore, we adopted an existing model and adjusted it to our specific Wolf-Sheep-grass dynamics found in NetLogo Models Library (Salecker et al., 2019). The resulting code can be found in Appendix A, *Wolf-Sheep-Grass.ipynb*, a Jupyter Notebook containing a fully functioning Wolf-Sheep-Grass model.

### Wolf-Sheep-Grass Model in Python

The model is implemented in Python using the package Mesa (Kazil et al., 2020). Mesa is a Python library for agent-based modeling for simulating complex systems and emergent behaviors. In Mesa, we have *MultiGrid* for modeling space and *RandomActivation* for time. The agent and Model are defined by Mesa in classes *Agent* and *Model*.

In the code, we have a class *RandomWalker* as the base class for the agents *Wolf* and *Sheep*. In this class, we define the coordinates for the movement of the agents in space. The movement of the agents is *random*. There are three agents in the model, namely *Wolf*, *Sheep*, and *GrassPatch*. The agents *Wolf* and *Sheep* inherit from the class *RandomWalker*. The *Sheep* eats the *GrassPatch* and the *Wolf* eats the *Sheep*.

The *WolfSheepGrassModel* inherits from the *Models* class from Mesa. Here, we define the initial parameters like the number of *Wolf*, *Sheep* and the dimensions of the space. Further,

the *step* method has a container for collecting the number of Wolf and Sheep at each *timestep*. Finally in the *run\_model* method we define the number of *timestep* to execute and the *step* method is called at each *timestep*. The *WolfSheepGrassModel* is initiated in the class *Scenario*, to be consistent with the EvoLife implementation.

## Wolf-Sheep Implementation in Evolife

After verifying the model described in the previous section, we sought to implement this as a Scenario in the Evolife software. The code for this Scenario can be found in Appendix B, *S\_Wolfsheep.py*. For simplicity, we omitted the effect of the grass and focused on the Wolf-Sheep dynamics. Similarly, as in the Wolf-Grass-Sheep model, the agents are modeled as random walkers, and when running into each other a wolf will eat a sheep with a certain probability. The sheep reproduction rate affects the time it takes for the sheep population to recover. When running the Wolfsheep scenario in Evolife, the population numbers of wolves will be plotted in black, and the population numbers of sheep will be plotted in white.

## Results

In the section below first, the results with different initial parameters of the *Wolf-Sheep-Grass.ipynb* model will be included, subsequently, the *WolfSheep.py* scenario implementation in Evolife.

### Wolf-Sheep-Grass Model in Python

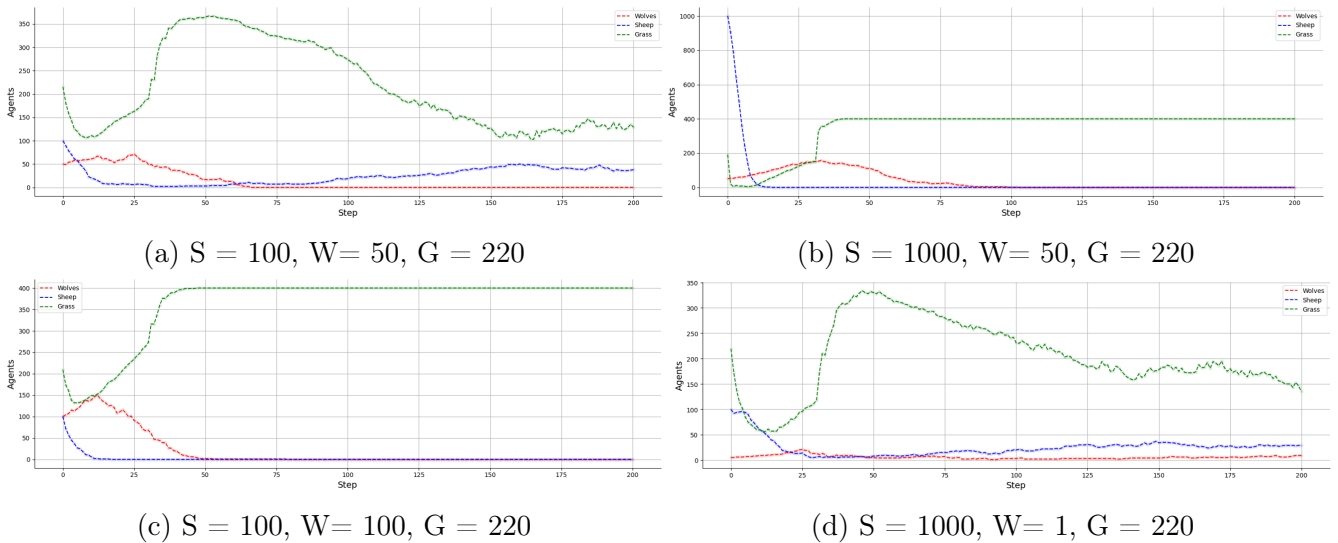


Figure 2: Plots of Wolf-Sheep-Grass Models with varying initial conditions for the number of Wolves ( $W$ ), number of Sheep ( $S$ ) and amount of Grass ( $G$ ).

We tested a few different initial values to verify the model and to see the effects of the population dynamics in the long term. In the figure above the different effects of varying initial conditions for the amount of Wolf, Sheep and Grass agents can be seen.

In Figure 2a, there are initially a lot of sheep, these eat a lot of grass, the wolf population also increases because of this. Therefore, the population of sheep decreases, to a critical value that the wolves can no longer sustain themselves, leading to the extinction of the wolves. The small population of sheep rises in the end again, resulting in a Sheep-Grass alternating equilibrium.

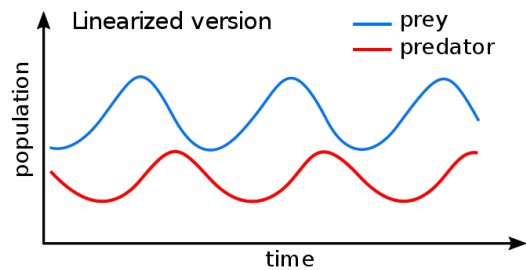
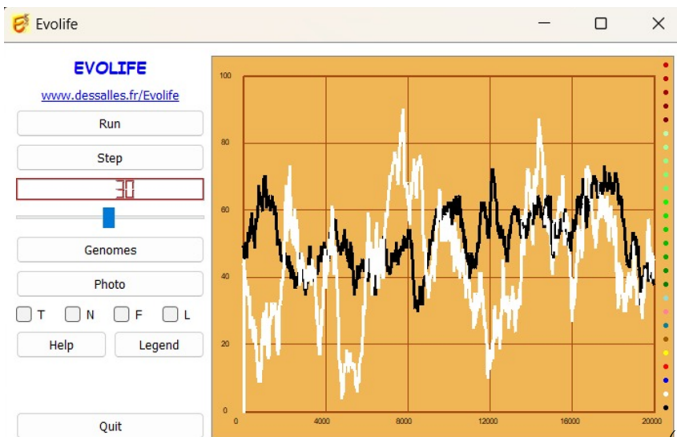
In Figure 2b, there is a huge population of sheep in the beginning, leading to a quick rise in wolves and grass depletion, ultimately leading to the death of all sheep. This ultimately leads to the extinction of wolves and grass domination. In this case, both the low presence of grass, and the relatively high prevalence of wolves lead to the extinction of sheep.

In Figure 2c, there is initially an equal population of sheep and wolves. The population of wolves rises so rapidly that the sheep will go extinct, after this, the wolves go extinct. Ultimately, only grass will be left. Differently, as in the previous case, solely the wolves were the limiting factor for the sheep population

In Figure 2d, we can see a particular outcome of the Wolf-Sheep-Grass experiment. Initially, the population of sheep decreases and the population of wolves slightly increases, until these both reach a plateau. After the recovery of the grass factor, the sheep population slightly increased just as the wolf population. In the end, the model seems to reach a Wolf-Sheep-Grass equilibrium, but it also seems as if the simulation is not quite stable yet. (Note: As the wolves reproduce asexually, therefore, one wolf can start a population)

From these analyses, we verified the accuracy of the model on the effects of the different agents (Wolf, Sheep, and Grass) on each other and their interactions.

### Wolf-Sheep Implementation in Evolife



(a) Population Wolves (Black) and Sheep (White) (b) Lotka-Volterra Predator-Prey model, image from “Lotka–Volterra equations”, 2023

Figure 3: Comparison Wolf-Sheep Model in Evolife vs Lotka-Volterra Model

The Evolife implementation shows the alternating population sizes of the wolf and sheep agents. As is evident in Figure 3a a rise in Wolf agents, precedes a decline in Sheep agents. This decline reach a critical value, at this point, the Wolf population decreases due to the lack of sheep, the sheep population recovers, resulting again in a rise of the wolf population. In this trial, there is a cyclical alternation between the population sizes of the two agents, similar to as in the classical Lotka-Volterra model which is included in Figure 3b

In Appendix C, images of multiple runs are included to indicate the repeatability of this cyclical Wolf-Sheep model. Appendix D includes a video of 1 single run of the WolfSheep Scenario in Evolife.

## Discussion

The Evolife implementation captures the essence of wolf-sheep cyclical dynamics. The characteristic rise and fall of predator and prey populations is evident. The proportions of wolf and sheep populations do not precisely match external models, likely due to differences in effect

parameters and initialization. Noise in the Evolife model results from inherent stochasticity. While the external Python model functions quite properly, there are several improvements that could still be made in the Scenario implementation in Evolife.

Key aspects for improvement include adjusting parameters to better reproduce external model dynamics, adding initial condition settings, and incorporating the grass/resource factor. The noise could be reduced by tuning the stochastic algorithms. Further testing across a range of initial populations and model variations could improve robustness. Overall this project demonstrates a proof of concept for modeling ecological interactions in Evolife. The wolf-sheep system provides a foundation for expanding to more complex and realistic food web simulations.

## References

- Dessalles, J.-L. D. (n.d.). JL Dessalles - Evolife. Retrieved November 24, 2023, from <https://evolife.telecom-paris.fr/>
- Kazil, J., Masad, D., & Crooks, A. (2020). Utilizing python for agent-based modeling: The mesa framework (R. Thomson, H. Bisgin, C. Dancy, A. Hyder, & M. Hussain, Eds.), 308–317.
- Lotka–Volterra equations [Page Version ID: 1184078449]. (2023, November). Retrieved November 24, 2023, from [https://en.wikipedia.org/w/index.php?title=Lotka%E2%80%93Volterra\\_equations&oldid=1184078449](https://en.wikipedia.org/w/index.php?title=Lotka%E2%80%93Volterra_equations&oldid=1184078449)
- NetLogo Wolf Sheep Predation model - EduTech Wiki. (n.d.). Retrieved November 24, 2023, from [https://edutechwiki.unige.ch/en/NetLogo\\_Wolf\\_Sheep\\_Predation\\_model](https://edutechwiki.unige.ch/en/NetLogo_Wolf_Sheep_Predation_model)
- Salecker, J., Sciaini, M., Meyer, K. M., & Wiegand, K. (2019). The nlr<sub>x</sub> r package: A next-generation framework for reproducible netlogo model analyses. *Methods in Ecology and Evolution*, 10(11), 1854–1863.
- Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—an embodied modeling approach. *Cognition and instruction*, 24(2), 171–209.

# Appendix C

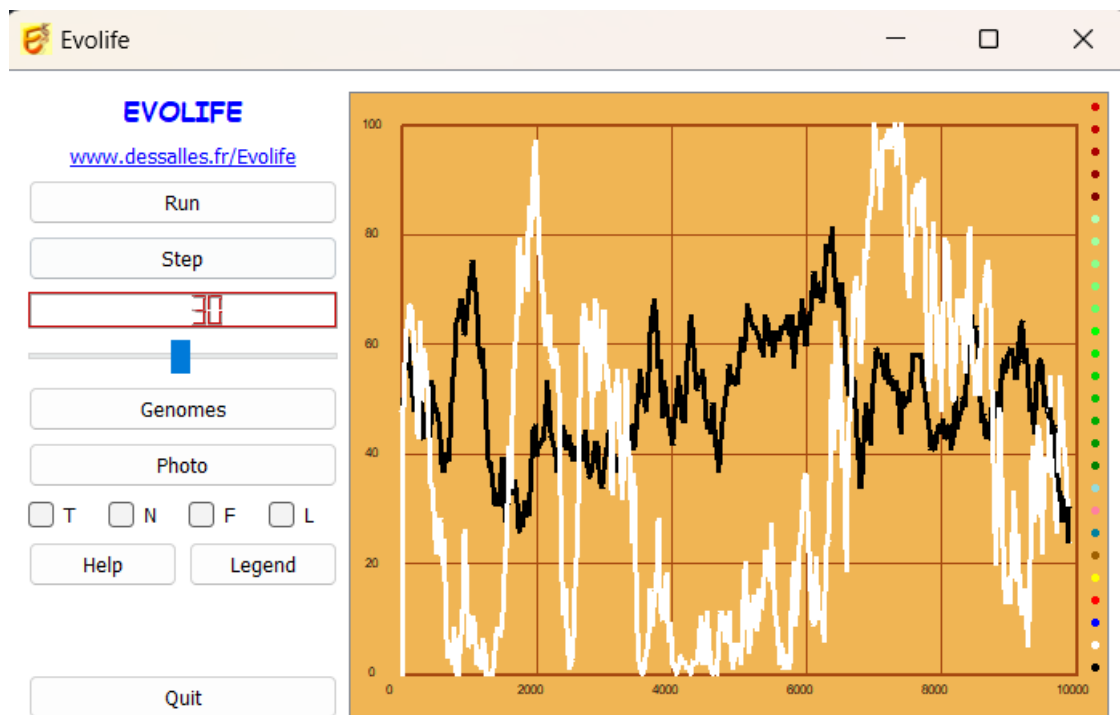


Figure 4: Run 1

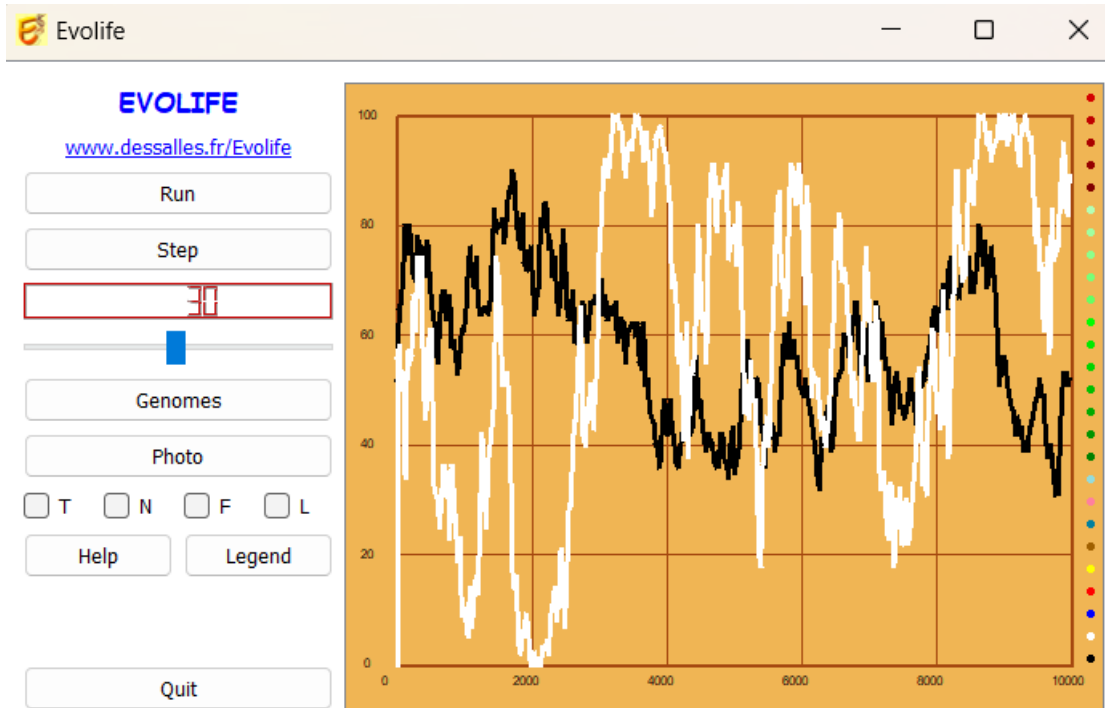


Figure 5: Run 2

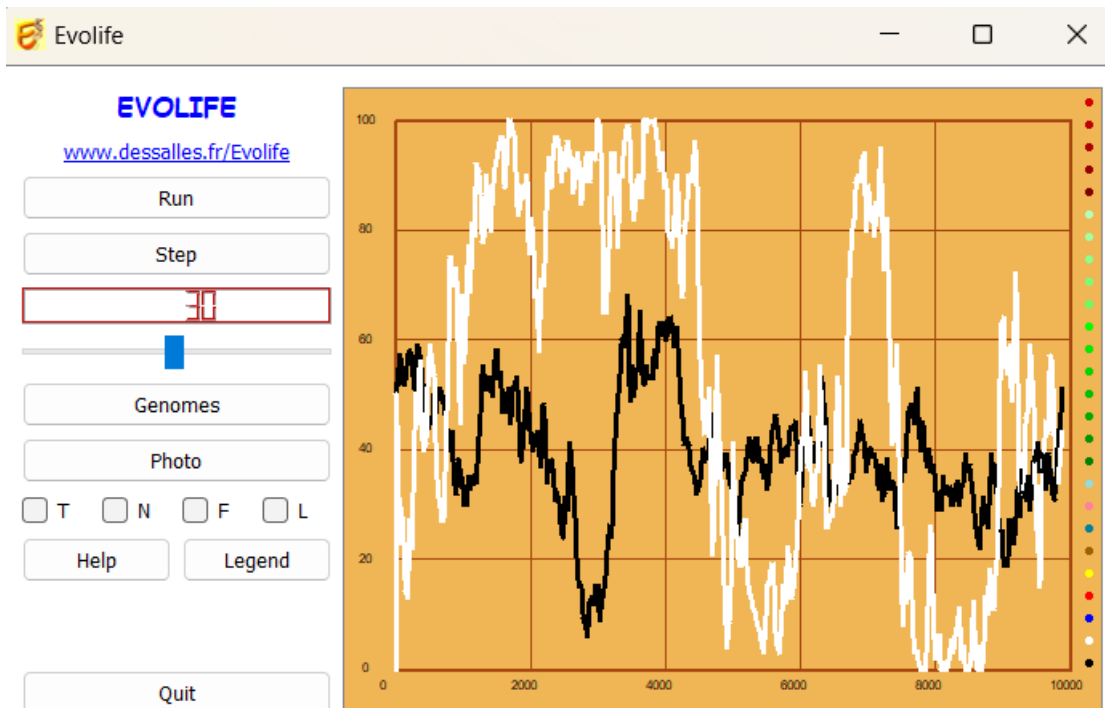


Figure 6: Run 3

## Appendix D

Link to a video of 1 run:

[CLICK HERE!](#)

or here:

<https://www.loom.com/share/849e3f6763d1403b9bb33ef1524174e9?sid=6432671b-c5e2-434d-9390-4c581ab44>





ATHENS course : **TP-09**

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)

Name: **Jan Kuc**

---

## Study of the impact of adding gender property to Emergence of Segregationism

### Abstract

The study focused on the impact of a *gender* parameter within the segregation algorithm based on Schelling's model. The investigation examined scenarios where not all individuals, but a majority, might exhibit gender attraction. Under these conditions, individuals were considered satisfied if they had at least one opposite-gender individual in their neighbourhood.

### Problem

The project addresses the Schelling model, which simulates agent behavior based on satisfaction derived solely from the presence of similar individuals in neighbourhood. Introducing the *gender attractiveness* parameter explores a phenomenon where agents, apart from preferring closeness to similar units, also look for the presence of individuals of the opposite gender. It's important to note that this modification and the test scenario do not directly represent the desire for relationships or partnerships, but rather the desire for the presence of at least one individual of the opposite gender in their surroundings. This modification seeks to explore how such preferences might affect the segregation dynamics in agent-based simulations, exploring an interesting part of social behavior that goes beyond just looking for similar company.

Expected results and insights from study of a such problem are:

- changes in segregation dynamics due to the *gender attractiveness* parameter,
- fluctuations in agent satisfaction levels concerning both similarity and the desire for the opposite gender's presence,
- exploration of the balance between these preferences and their impact on agent behavior,
- identification of emergent patterns or unexpected formations.

## Method

In the standard version of the Emergency of Segregationism implemented in *Evolife* individuals satisfaction is determined only basing on presence of the units of the same colour in the neighbourhood of the individual. So an individual can be satisfied (it means that it do not need to move and look for other place in grid) if the ratio of neighbours of different and same color is above *Toleration* parameter. This is described with given formula:

$$\frac{\text{different}}{\text{same} + \text{different}} > \text{Tolerance}$$

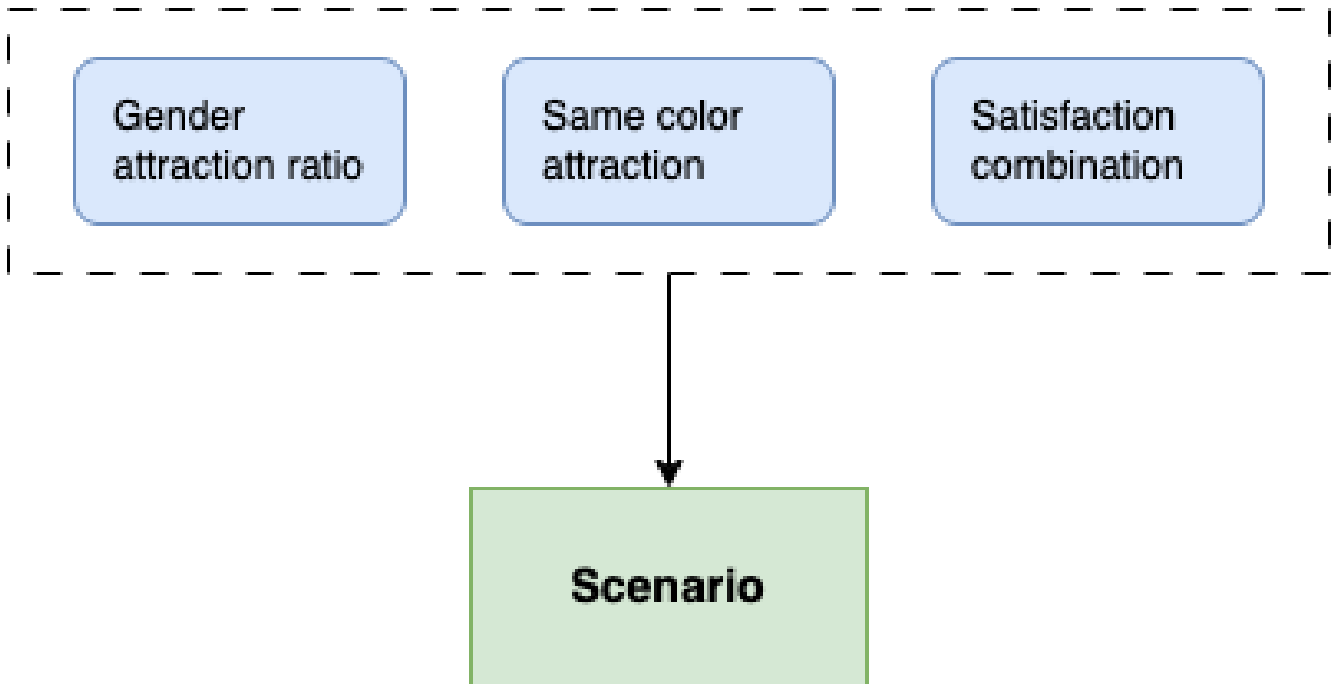
In order to add the influence of the 'gender' parameter into the segregation algorithm, I opted to utilize the colour groups already present in the existing program version. To introduce the effect of opposite-gender interest on agent behavior, the following attributes were added to the *Individual* class:

- *gender*,
- *if gender attracted* – property indicating an individual's attraction to the opposite gender,
- *gender satisfied* – property determining an individual's satisfaction regarding the presence of the opposite gender

Considering the objectives of this study and the multifaceted nature of individual interest and satisfaction concerning the opposite gender, the following customizable scenario parameters were defined and added to *Scenario* class:

- *gender attraction ratio* – probability of an individual being attracted to the opposite gender,
- *same color attraction* – specifies whether individuals are satisfied with the presence of the opposite gender from another color or only with units of the same color,
- *satisfaction combination* – determines whether a gender-attracted individual needs satisfaction from both default and gender-specific rules or is satisfied with one of those rule separately.

### ***Additional parameters***



Those modifiable parameters of scenarios allow to test various cases, for instance: gender attracted individual could be satisfied with presence of opposite gender unit regardless of the color its color and to consider an individual as satisfied both rules of satisfaction need to be fulfilled.

Introduction of the *gender* property and its influence on segregation also required modifications to the satisfaction function. The previous version considered individuals of the *Same* type only based on their color equality to the given individual. Now, individuals of the *Same* type include those from the same color family, irrespective of gender. Similarly, individuals of the *Different* type are now all individuals from a different color family. For a clearer illustration, the dictionary `self.colour_groups` defines color families as follows:

```
{'red' : ['red0', 'red5'], 'blue' : ['blue0', 'blue5']}
```

Next, individuals of the opposite gender are counted, determining the boolean value of the *gender\_satisfied* attribute. The enclosed code snippet exemplifies this process:

```
if self.ifGenderAttr:
    if self.Scenario.sameColorOnly:
        Opposite = sum(
            [
                Statistics[C]
                for C in self.Scenario.Colours
                if C in self.opposite_gender and C in self.species_colours_lst
            ]
        )
    else:
        Opposite = sum(
            [
                Statistics[C]
                for C in self.Scenario.Colours
                if C in self.opposite_gender
            ]
        )
    self.gender_satisfied = False if Opposite == 0 else True
```

Here's an exemple of dictionary defining the correspondence between gender and colors:

```
{'red0' : 0, 'red5' : 1, 'blue0' : 0, 'blue5' : 1}
```

where gender is represented by the values 0 or 1.

Finally, based on the selected scenario settings, determining an individual's satisfaction follows attached part of code:

```
if self.ifGenderAttr:
    if self.Scenario.satisfyComb:
        return self.satisfied or self.gender_satisfied
    else:
        return self.satisfied and self.gender_satisfied
else:
    return self.satisfied
```

# Results

Several test scenarios were designed, and below, their outcomes will be presented.

## Test #1

### Scenario parameters:

- gender attraction ratio: 0.65
- same color attraction: True - attraction only to opposite gender of the same color
- satisfaction combination: 'and' - both satisfaction rules must be fulfilled

### Simulation parameters

- population size: 1100
- neighbourhood radius: 1
- tolerance: 60

```
Curves:
red0: average satisfaction of red0 individuals (gender: 0)
red5: average satisfaction of red5 individuals (gender: 1)
blue0: average satisfaction of blue0 individuals (gender: 0)
blue5: average satisfaction of blue5 individuals (gender: 1)
#FFFFFF: average global gender satisfaction
=====
( [Esc] to close )
```

Figure 1: Legend for generated charts

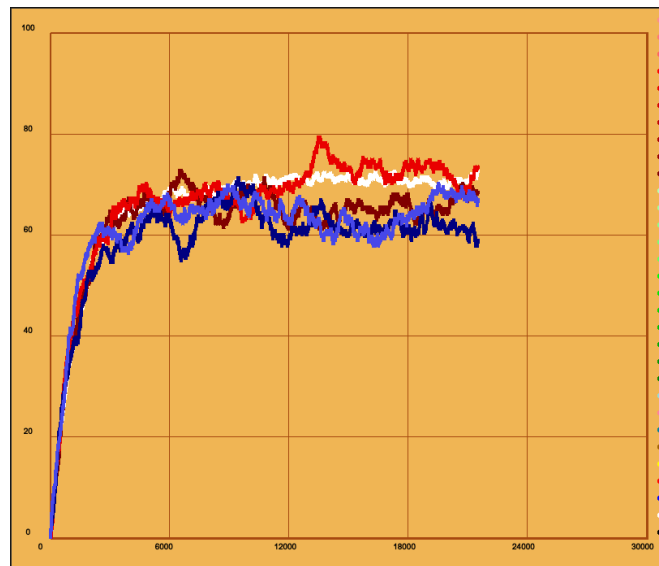


Figure 2: Satisfaction curves for Test #1

The achieved level of satisfaction appears lower when compared to the version without gender attraction. Individuals face increased challenges in identifying suitable neighborhoods. Here, individuals are required to fulfill two satisfaction criteria to decide on remaining in their current location without moving. Consequently, the process becomes more complex and demanding to find a satisfactory place.

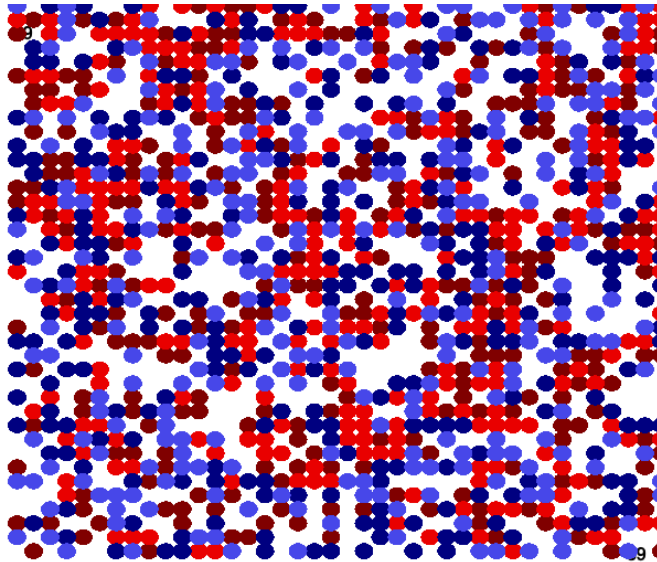


Figure 3: Generated field for Test #1

## Test #2

### Scenario parameters:

- gender attraction ratio: 0.65
- same color attraction: True - attraction only to opposite gender of the same color
- satisfaction combination: 'and' - both satisfaction rules must be fulfilled

### Simulation parameters

- population size: 1100
- neighbourhood radius: 2
- tolerance: 60

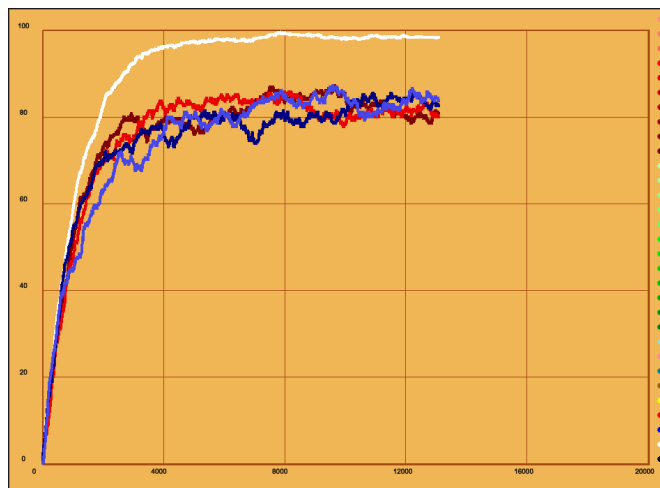


Figure 4: Satisfaction curves for Test #2

In this case, radius of neighbourhood was increased, so it was easier for units to find satisfactory place, but still segmentation is not perfect. Global average gender satisfaction (white curve on Fig. 4) is on very high level, that means individuals properly seek opposite-gender connections within their neighborhoods.

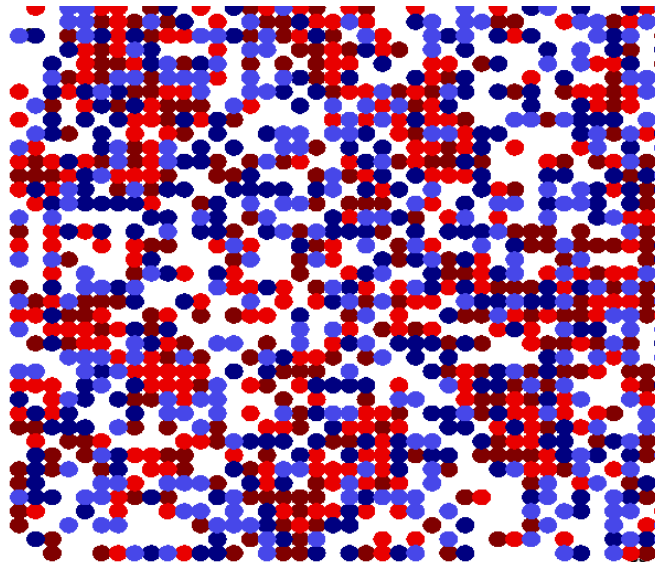


Figure 5: Generated field for Test #2

### Test #3

#### Scenario parameters:

- gender attraction ratio: 0.65
- same color attraction: False - attraction to opposite gender, regardless of color
- satisfaction combination: 'and' - both satisfaction rules must be fulfilled

#### Simulation parameters

- population size: 1100
- neighbourhood radius: 1
- tolerance: 60

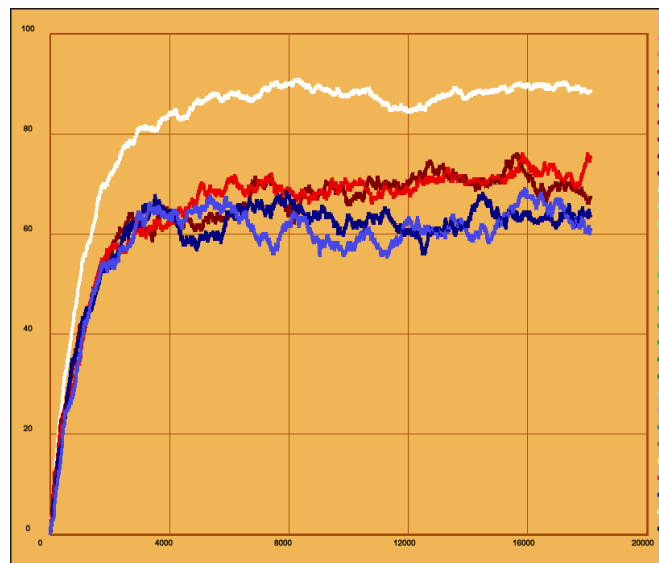


Figure 6: Satisfaction curves for Test #3

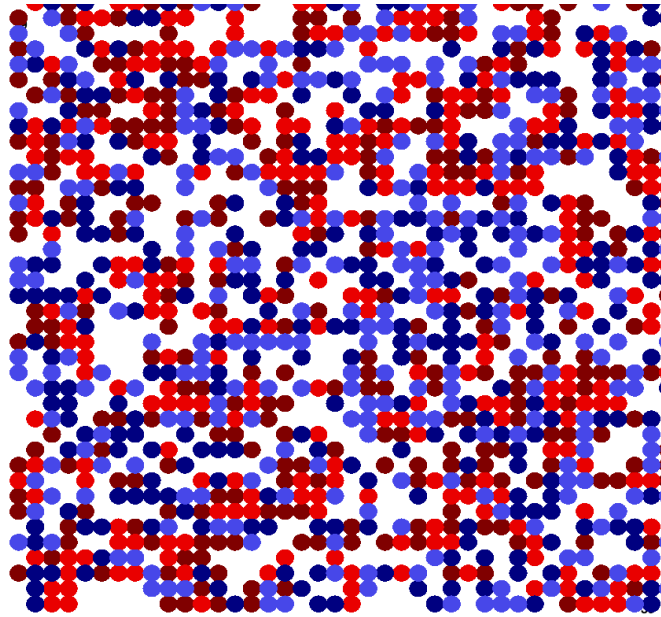


Figure 7: Generated field for Test #3

In this scenario, individuals could be satisfied with presence of opposite-gender unit even if unit was not from their color family. That resulted in increased global gender satisfaction, but each group satisfactions decreased. This outcome might not be surprising, as the satisfaction rules now have the potential to conflict with each other, consequently contributing to lower segmentation.

## Test #4

### Scenario parameters:

- gender attraction ratio: 0.6
- same color attraction: False - attraction to opposite gender, regardless of color
- satisfaction combination: 'or' - one of satisfaction rules needs to be fulfilled

### Simulation parameters

- population size: 1100
- neighbourhood radius: 1
- tolerance: 60



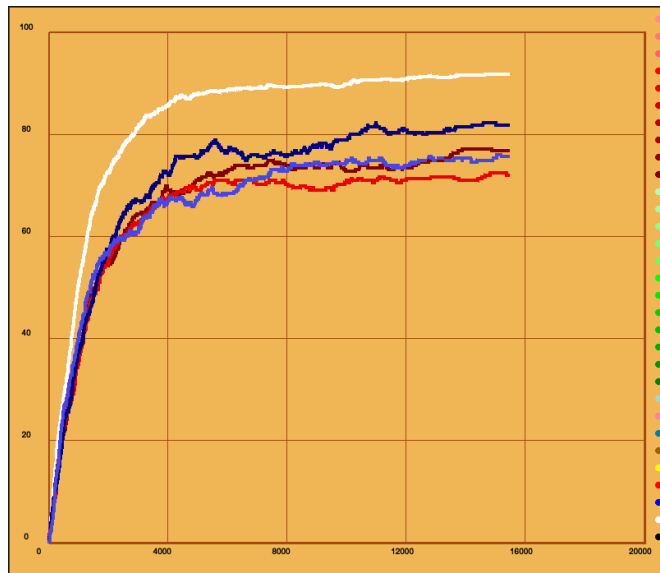


Figure 8: Satisfaction curves for Test #4

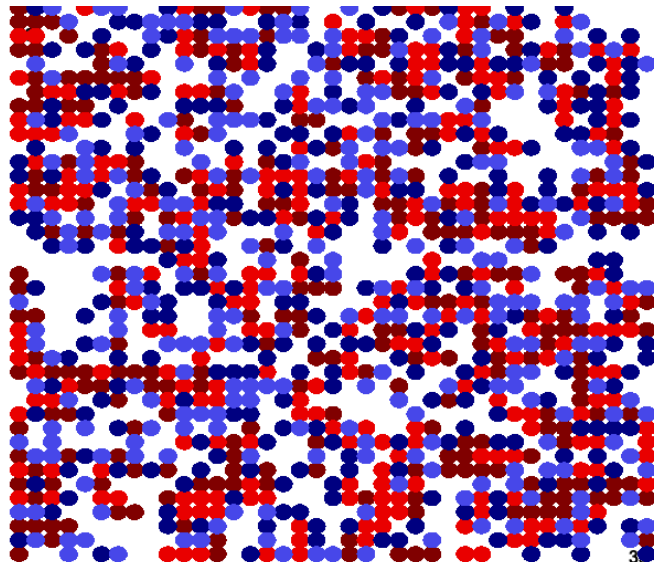


Figure 9: Generated field for Test #4

In the last scenario, satisfaction conflict which occurred in the previous case should be eliminated, because now each gender attracted individual can be satisfied by one of the two satisfaction rules. This leads to slightly better segmentation with the global average gender satisfaction level still higher than group satisfaction levels.

## Discussion

The experiments conducted to explore the impact of gender attraction on segregation dynamics in the simulation provided interesting observations. The introduction of gender attraction significantly influenced segregation patterns, demonstrating that adopting two satisfaction rules for individuals increased the challenge of finding suitable neighborhoods. Despite an increase in the global level of gender-related satisfaction, there was a noticeable decrease in satisfaction within individual groups.

Comparing the results with the expectations highlighted the complexities due to the dual satisfaction rules, leading to decreased satisfaction for each group but an overall rise in global gender satisfaction. The two rules satisfaction conflict was resolved to a certain extent, by allowing individuals to choose between the two rules. It showed promising improvements in segmentation, but still the global gender satisfaction level remained higher than satisfaction per group.

Future research could focus on improving the satisfaction criteria or studying how gender attraction affects segregation in different population sizes, areas or when populations of each gender have different sizes. Exploring how various factors influence satisfaction rules might help us understand social segregation dynamics better. Another study case could be defining gender attraction as the desire of individual to have a partner and be in a relationship.

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](https://teaching.dessalles.fr/ECS)

Name: **Thibault Lambert**

---

## Cocktail Party: An Implementation of Agent Movement

### ***Abstract***

This study aims to enhance the cocktail party problem by incorporating agent movement into algorithms. Agents will attempt to join discussions or flee them if they cannot tolerate the sound volume.

### ***Problem***

In the cocktail problem, agents can emit sound and hear other agents. By making agents speak louder when there is noise around them, the global sound level will significantly increase, leading to emergent phenomena.

The starting point of my thought was that a substantial number of agents were needed to observe emergence. Thus, my idea was to enable them to group up to expedite the process. This also allows us to observe group emergence.

### ***Method***

I began by studying the cocktail algorithm, focusing on the Landscape, Agent, and AgentPopulation classes. The program randomly selects an agent to check if it needs to raise its voice. Therefore, I had to implement agent movement using a function and incorporate it into this part of the code. Since the code does not use Numpy, I had to search for sound level maxima around the agent.

An issue arose because a function that returns sound levels was implemented, but agents were influenced by their own emissions, causing them to move toward the top-left corner if they were alone.

## **Results**

Despite the program's relatively slow performance, we observe both the emergence of groups and changes in sound levels. Emergence is, on average, three times quicker than the previous model. The most interesting outcome is the appearance of small groups (4 or 5 people) that start moving and repelling other agents.

## **Discussion**

The simulation matches the expected results. A quicker version using *Numpy* could be a very good addition from this point. As the simulation has been made in a thoric world, it could also be interesting to observe edge effects.



December, 2023

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](https://teaching.dessalles.fr/ECS)

Name: Missôra MAMILO

---

## A quest for Enquist Hawk and Dove dilemma implementation

### **Abstract**

Enquist proposed a version of the Hawk and Dove dilemma where each animal could signal what type they were and even lie about it. This study is an attempt to implement part of his proposal and discover which combination of behavior is the most favorable.

### **Problem**

How important can the immediate environment surrounding a combination of gene ( a behavior) be to determine its quality?

### **Method**

To implement the lying factor, another gene was added to the genome where 0 meant the bearer was truthful and 1 meant the bearer was a liar. Individuals were randomly assigned an allele of the gene. Then, depending on the two genes, the type of animal and the truthfulness, the interactions between them were determined based on a imagining of the possible outcomes and the list is available as an annex. The outcomes in terms of value where adjusted to balance the study as much as possible. Depending on the interactions, lying costs or rewards the individual signaling, this factor is adjustable in the parameters of the application. Damages were measured with BattleCost and adjusted to fit the situation.

With the population randomized, each interaction is as likely as the other and the average score of an animal is the quarter of the sum of each interaction with a different combination.

TD= 3V-2LC

LD= 6V-12C-18LC

TH= 17V-3C

LH= 19V-7C-2LC

## Results

The results show that depending on the run all combination of alleles can be possible. The effect of the value of the lying factor as well as the effect of the composition of the population at year 0 were studied. The results are obtained from 15 consecutive runs.

The composition of the population in the beginning of the simulation seems to have little effect. Indeed, the composition at the start and end rarely changes with a 0.5% (50/1000) even if switching might seem favorable for the individual.

	Lying=0	Lying=10	Lying=1000
V=10 C=40	TH(7)	TD/LH (5)	LD/TD (5)
V=C=1	TD(8)	LD(7)	LD(7)
V=40 C=10	TD(5) LH/TH(4)	LD(7)	LD(9)

LD wins in 5 cases out of 9 and TD in 4 out of 9. LH and TH win one and are close second in another case.

It seems Truthful animals do better when Lying is null and Liars do better when Lying is a bigger number.

## Discussion

Before running the simulation, lying hawks were expected to be successful as they win against TH, LD, TD and draw with their own kind. However it seems Doves are actually more successful (7/9 cases won). Moreover, Lying animals were expected to win more when Lying was null since they would keep their benefit without suffering the cost if caught for LD or fighting another LH for LH. However, the coding does not reflect this as Lying=0 causes Liars to lose a lot of their benefits, this should be patched in the future to have a more realistic and balanced simulation.

The study shows a inclination towards Doves and especially Lying ones. However, this system can get quite tricky to balance and code cleverly and thus results might be skewed by the point system. Furthermore, since this study is a straightforward addition of lying to the

original Hawk-Dove dilemma, the interactions between species still favor Hawks all the time. Also the number of runs done to come to a conclusion does not seem large enough to accurately represent the situations. Some results might be difficult to interpret based solely on the math as they require to take in account all the different factors and try to make sense of them. For example, in a population of LD where no one is able to eat any food it would seem logical that Hawks would appear as they have the advantage in a population of LD but the population stays LD, maybe because the population dies too quickly to reproduce efficiently or because the mutation rate is too low to shift an entire population. In short, interpretation can easily get tricky.

In terms of perspectives, other than improving the balance of the game, an earlier idea was to expand on Enquist's ideas and add several archetypes or combination of gene. There could be liar, deaf/blind animals (that have trouble perceiving the signaling), loud animals, cowards (who would attack only if they are sure to win and/or retreat if the opponent is louder than them) and distrustful animals (who would never believe the opponent signaling). Implementing the system of repetition of signaling would also be a good idea.

What is to be remembered in this study is that a combination of allele or a behavior is not good in itself, it either is fit to survive in an environment or it is not. When it comes to genes, there is no absolute and sometimes being a lying coward works better than being an honest peacemaker.

## ***Bibliography***

Deterring signals - Magnus Enquist (Anim. Behav. 1985, 33, 1152-1161)





November, 2022

TELECOM  
Paris



ATHENS course : TPT-09

# Social Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/SECS](http://teaching.dessalles.fr/SECS)

Names: Arthur MARTINS BRAGA, Lucas PALMIRO DE FREITAS

## Emergence of segregationism: adding criteria to choose where to move

November 28, 2023

### Abstract

The Thomas Shelling model simulation shows that segregation disappears when tolerance is diminished. To address this unexpected outcome, the project aims to enhance realism by making unsatisfied to choose where to move based on a criterion rather than randomly. The subsequent simulation aims to analyze the results of this improved model.

### Problem

The execution of the implementation of the Thomas Shelling model [1] studied in class shows the emergence of segregationism among individual agents of two types that have preferences over the composition of its neighbourhood. These preferences are expressed by the simulation parameter named **Tolerance**, which is defined as the percentage of individuals that are different ( $= \text{different} / (\text{same} + \text{different})$ ) above which one decides to move. It was seen that a segregated state is achieved by the simulation even with an intermediate tolerance value.

Moreover, it was observed that segregation disappear when individuals have a low tolerance value, more precisely below 25% (with default parameters). However, one can consider this result to be counter-intuitive, since, from empirical experience, it could be expected that with a low tolerance, there would be a high segregation.

That way, it was found that in the implementation provided in the *Evolife* software [2], an individual unsatisfied with its current neighborhood randomly moves to a new empty location. This behavior in the simulation is different from what would be observed in reality,

because, considering the real example of searching for a new residence, an agent is expected to evaluate some housing options and choose the one that best suits their preferences before committing to to change.

Based on this, this project aims to implement a new mechanism for unsatisfied individuals to move in order to improve the model, adding greater realism to its operation. From this, the aim is to study the impact of this addition on the simulation results and verify whether it is possible to observe segregation with low tolerance values.

## Method

So that each individual agent can evaluate some options for where to move instead of moving randomly, a new parameter was added to the simulation, called **SearchProportion**. It is defined as the percentage of empty locations an unsatisfied individual will consider when moving. If it equals zero, the original behavior is performed, that is, the individual chooses a random place in the land.

This was done so that each individual would not have to evaluate all possible empty locations on the grid, which would be very inefficient, leading to a very slow and therefore very long simulation. Furthermore, this behavior would also not reflect reality well (the objective of this project), as it is impossible to consider all housing possibilities when moving.

Thus, the number of options considered by an individual (**nOptions**) is obtained by the following formula, expressed as a function of other simulation parameters:

$$nOptions = \frac{SearchProportion}{100} \cdot (LandSize^2 - PopulationSize)$$

This formula can be understood as a fraction of the number of empty locations, which is given by subtracting the total number of locations in the grid by the population size. In fact, the number of options is given by the maximum between the integer part of the formula result and 1 so that, if the formula provides a value less than 1, at least one location is chosen, which is exactly equivalent to the original behavior of the simulation.

With this, the implemented algorithm randomly selects **nOptions** empty locations and evaluates the neighborhood of all options based on their diversity. If among the options considered, there are empty spaces whose neighborhood has a diversity below or at most equal to the threshold of the **Tolerance** parameter, one of the options that satisfies the condition is chosen randomly. Otherwise, one is randomly chosen from among all the options. Note that in the second case the original behavior is reproduced.

Thus, it can be seen that the proposed algorithm seeks not to introduce any bias when choosing new locations to move to by still taking into account the tolerance threshold. In this way, an individual agent does not choose the option that has the highest proportion of individuals of the same color, but chooses a random option among those that do not exceed his tolerance or, if such options are non-existent, a completely random option.

The code relating to the implementation of this new behavior in the simulation is attached to this report. Modifications were made to only two files in the *Evolife* software:

- Evolife/Apps/Segregationism/Segregationism.py
- Evolife/Apps/Segregationism/SegregationismConfig.xml

Finally, with this implementation, the results obtained in several simulations carried out varying the main simulation parameters, that is, **Tolerance**, **NbColours**, **NeighbourhoodRadius** and the introduced parameter **SearchProportion**, are presented below.

# Results

## Varying search proportion

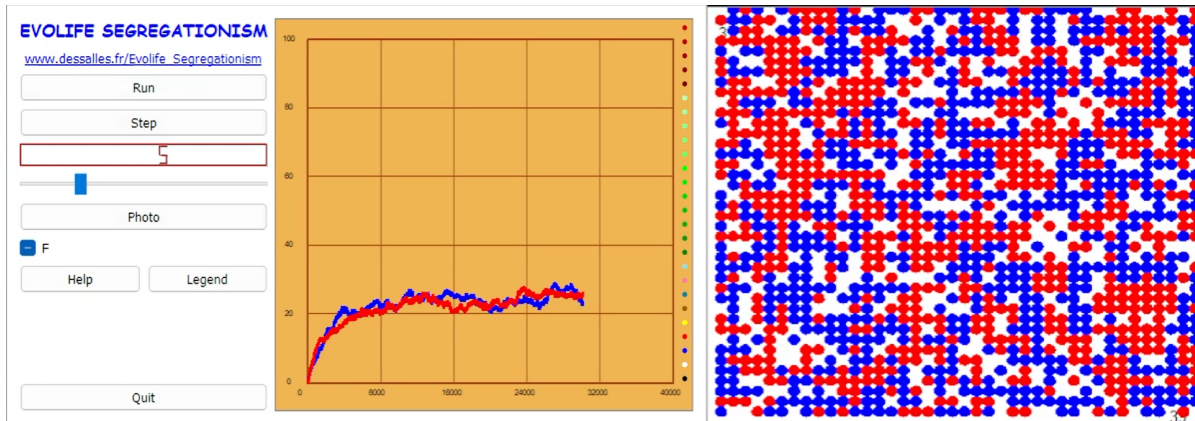


Figure 1: SearchProportion=0% tolerance=10%.

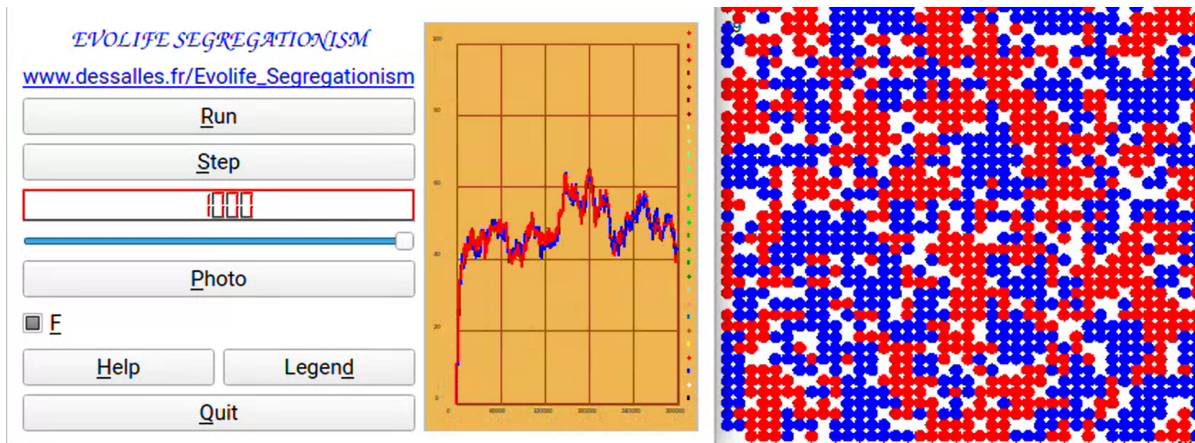


Figure 2: SearchProportion=1% tolerance=0%.

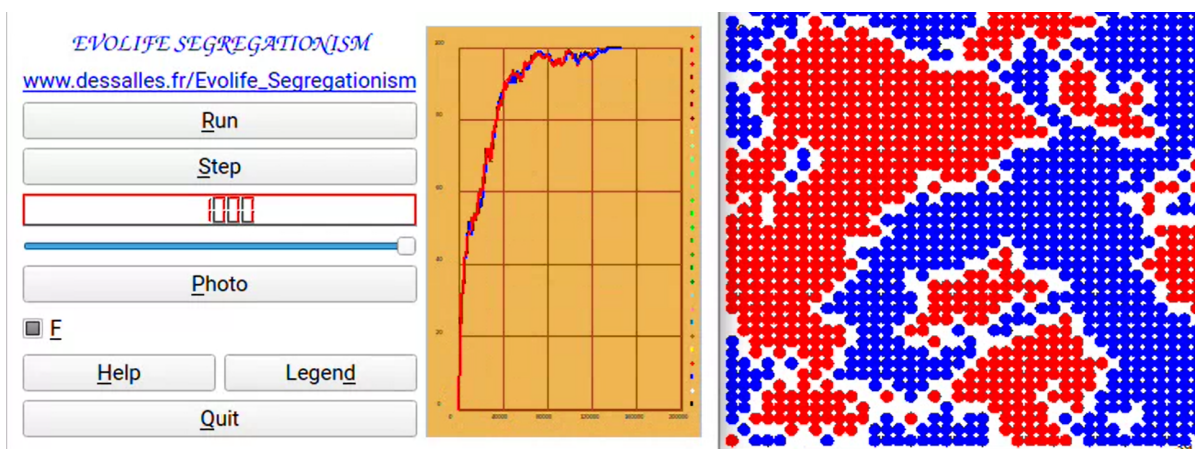


Figure 3: SearchProportion=2% tolerance=0%.

In Figure 1, it is evident that segregation does not manifest even after 3000 iterations, maintaining a consistent satisfaction level of approximately 25% for both colors. Moving on to Figure 2, the segregation phenomenon is notably absent even after 300,000 iterations. However, the satisfaction levels of both colors fluctuate between 60% and 40%. In contrast, Figure 3 reveals that segregation becomes apparent after 320,000 iterations, reaching maximum satisfaction for both colors.

## Varying number of colors

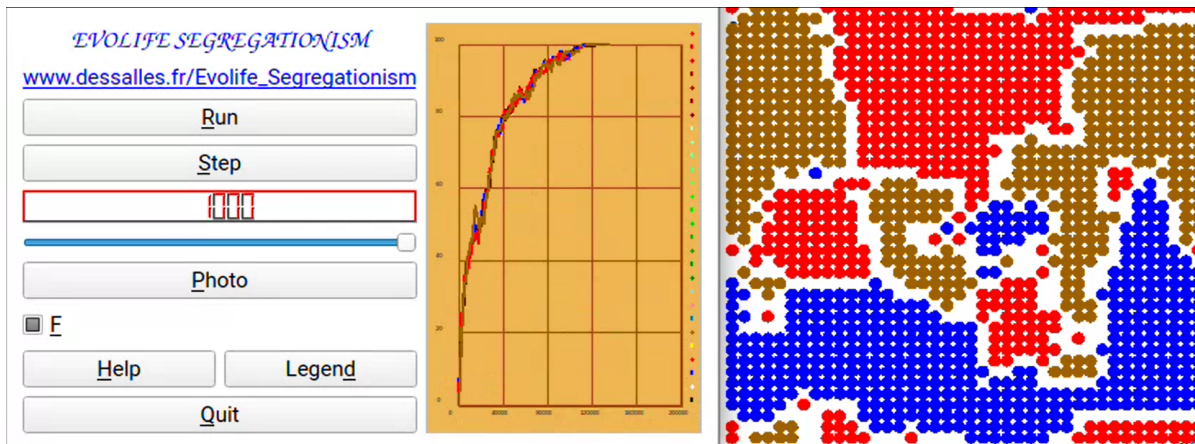


Figure 4: SearchProportion=1% tolerance=25% NbColours=3.

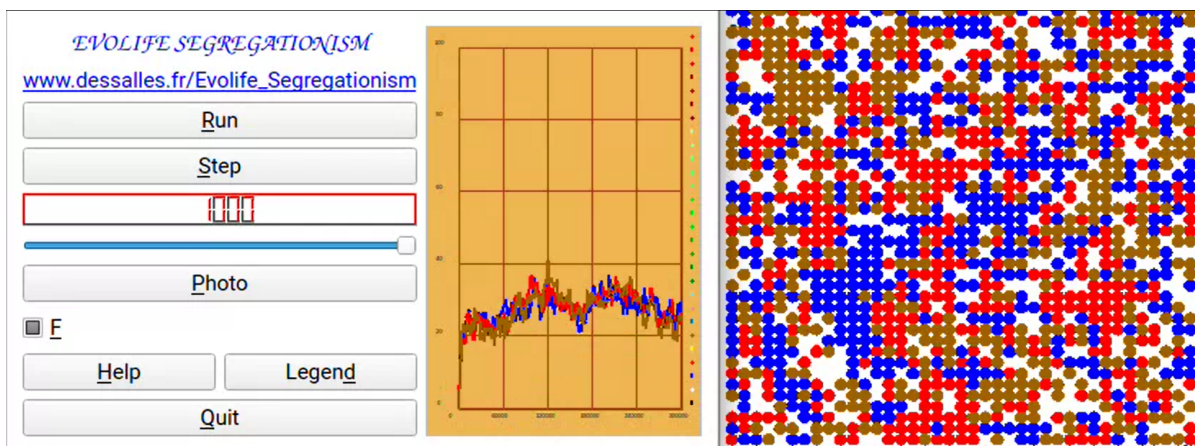


Figure 5: SearchProportion=0% tolerance=25% NbColours=3.

In Figure 4, it is evident that segregation occurs with a low tolerance and three different colors. Notably, each color attains the maximum level of satisfaction.

On the contrary, in Figure 5, segregation is absent, and the satisfaction levels of the three colors fluctuate between 20% and 40%.



## Varying neighbors radius

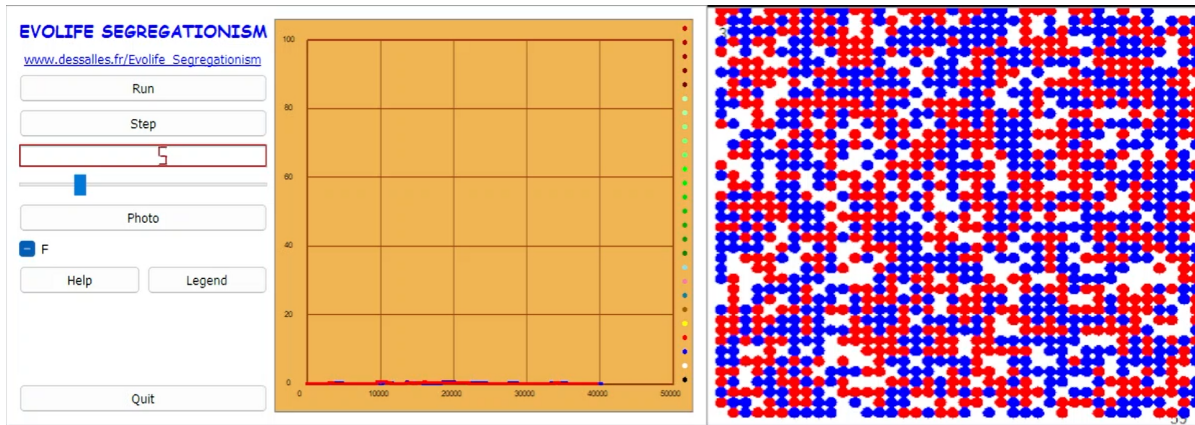


Figure 6: SearchProportion=0% tolerance=30% neighborsRadius=4.

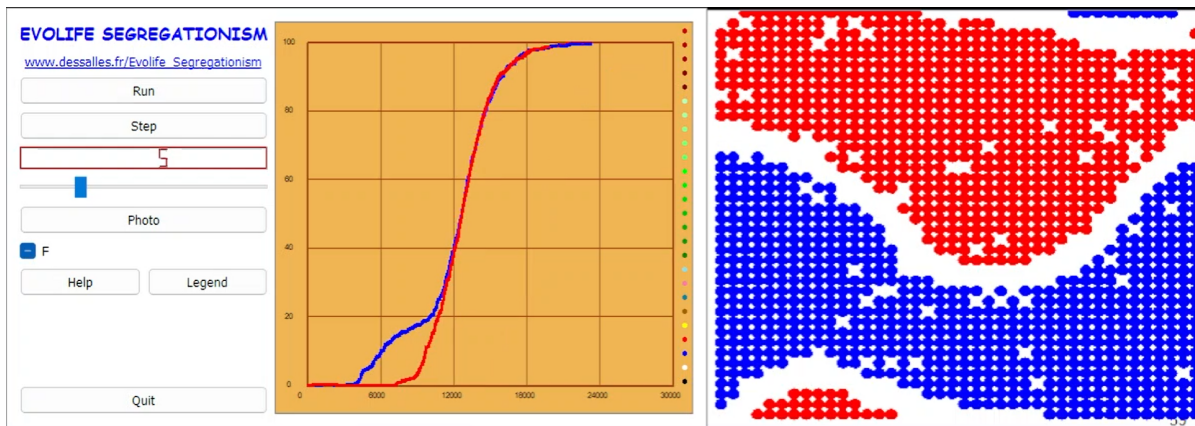


Figure 7: SearchProportion=10% tolerance=30% neighborsRadius=4.

In Figure 6, it is apparent that with a SearchProportion of 0% and a tolerance of 30%, no discernible changes occur even after 40,000 iterations. The satisfaction levels for both colors remain consistently around 0%.

Conversely, in Figure 7, segregation becomes evident after 24,000 iterations with a SearchProportion of 10%. The satisfaction graph initially shows little activity, but a notable transformation occurs as the blue entities start to cluster before the red ones. This clustering contributes to a gradual increase in satisfaction. Subsequently, the red entities also begin to group, leading to a rapid segregation between the two colors. In the end, both colors reach the maximum satisfaction level.

## Discussion

The observed results in the various simulations provide valuable insights into the dynamics of the Thomas Shelling model with the introduced modification of the SearchProportion parameter. The project aimed to address the counterintuitive findings in the original simulation, where low tolerance values did not lead to segregation, and to enhance the realism of agent movement.

Moving to the varying search proportion results, in Figure 2, the segregation phenomenon is notably absent even after 300,000 iterations. This aligns with the expected behavior, as no consideration is given to available options when an unsatisfied individual decides to relocate. However, the satisfaction levels of both colors fluctuate between 60% and 40%. In contrast, Figure 3 reveals that segregation becomes apparent after 320,000 iterations, reaching maximum satisfaction for both colors.

In Figure 4, it is evident that segregation occurs with a low tolerance and three different colors. Notably, each color attains the maximum level of satisfaction. On the contrary, in Figure 5, segregation is absent, and the satisfaction levels of the three colors fluctuate between 20% and 40%.

Moving on to the results with varying neighbors radius, in Figure 6, it is apparent that with a `SearchProportion` of 0% and a tolerance of 30%, no discernible changes occur even after 40,000 iterations. The satisfaction levels for both colors remain consistently around 0%.

On the other hand, with a `SearchProportion` set to 10%, segregation becomes noticeable after 24,000 iterations (Figure 7). What's intriguing is the satisfaction graph reveals a unique pattern not observed in other tests. In this scenario, the blue entities cluster together before the red ones. This is interesting because one might assume that the blues are more intolerant than the reds, suggesting that the segregation is their fault since they start segregating first. However, a closer look at the code implementation dispels this assumption. Another noteworthy aspect is that segregation takes some time to initiate; there are several iterations where nothing happens before the segregation becomes apparent. This behavior sets it apart from other runs.

The results align with the project's goals of introducing a more realistic decision-making process for individuals when choosing a new location. The `SearchProportion` parameter allows agents to consider a percentage of available options, introducing a degree of selectivity that reflects real-world decision-making.

While the modifications show promise in achieving more realistic outcomes, it is essential to acknowledge certain limitations. The introduced mechanism simplifies the decision-making process by considering a subset of available options. This simplification may not capture the full complexity of decision-making in real-world scenarios. Additionally, the impact of the `SearchProportion` parameter may vary with different parameter combinations, warranting further exploration.

In conclusion, the introduced modification enhances the Thomas Shelling model by incorporating a more realistic agent movement mechanism. The observed results demonstrate the model's sensitivity to the `SearchProportion` parameter and its potential to influence segregation dynamics.

## Bibliography

1. Schelling, T. C. (1978). *Micromotives and Macrobehavior*. W. W. Norton & Company
2. Dessalles, J.-L. (n.d.). *Evolife: A Platform for Agent-Based Modeling of Social Evolution*. Telecom Paris. <https://evolife.telecom-paris.fr>



December, 2023

# Social Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/SECS](https://teaching.dessalles.fr/SECS)

Name: Valentin Metz

---

## Predicting group behavior in rational games

### **Abstract**

We performed an in-classroom play of the “guess 2/3rd of the average” game and evaluate the results.

### **Problem**

Multiple individuals in a group submit their (secret) guess for a guessing game with the goal of being as close as possible to two-thirds the average guess of the whole group.

As every player knows that all other players have the same target, the target guess can be lowered by performing iterative reasoning.

### **Method**

All students were given a piece of paper on which they were supposed to write their name and their guess for the two-thirds group average.

### **Results**

In our experimental evaluation the average of all guesses was 28.780, with the two-thirds target being located at **19.187**.

### **Discussion**

Even in a room full of students with basic understanding of game-theory, the result did not converge fully onto the Nash-equilibrium of 0.

## ***Bibliography***

[https://en.wikipedia.org/wiki/Guess\\_2/3\\_of\\_the\\_average](https://en.wikipedia.org/wiki/Guess_2/3_of_the_average)

<https://www.youtube.com/watch?v=MknV3t5QbUc>



 	<p>November, 2023</p> <h1 style="text-align: center;">Emergence in Complex Systems</h1> <p style="text-align: center;">Micro study</p> <p style="text-align: right;"><a href="https://teaching.dessalles.fr/E">teaching.dessalles.fr/E</a></p> <p style="text-align: right;">CS</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Name Pierina MILLE

---

## Simulating the evolution of strength competition and sex ratio in mammals:

### ***Abstract***

The sex ratio in mammals is usually 1:1 in mammals and most other species. In this study, we aim to look at the influence of strength competition in males, allowing them to reproduce, on this ratio. It is found that although this strength reaches high values in the population, the gene that controls the sex ratio stabilizes the ratio to a 1:1 value.

### ***Problem***

Research (Düsing 1930) seems to suggest that most populations will present a 1:1 ratio and not only because of mechanics (diploid species). Indeed it was observed in many haploid species (only one sexual chromosome instead of two), and especially in some Hymenopterae that a 1:1 ratio is preserved. Indeed, having more males or females will not impact the number of children, but it will affect the number of grandchildren. It is more therefore more interesting to produce offspring in the minority sex. This leads to an equilibrium found at a 1:1 ratio. It is worth mentioning however that not all species respect this ratio.

To study the influence of male competitiveness, I added a gene called "Strength". It is a 10-bit long gene that gives a gradation of strength. When it is time to reproduce, I randomly select three fathers that will compete against each other. Those who have lesser strength (computed using `gene_relative_value`) die off without reproducing, and the strongest moves on to reproducing.

We can already estimate the mean maximal distribution of strength in the population. As both parents pass on their genes, and the strength of females doesn't play a role in their reproduction, we can put their mean strength at 50%. On the other hand males are selected to become increasingly stronger, we can hypothesize that this strength grows and reaches nearly 100%. They transmit this trait to both daughters and sons. Therefore the maximal theoretical mean strength should be 100%.

## Method

In order to run the simulation, I chose a "weighted" gene. This choice was mostly experimental, as with an unweighted gene I couldn't get an increasing strength distribution in the population. The difference between weighted and unweighted genes is that the bits of a weighted gene represent a binary value, whereas an unweighted gene has every bit count as much in the calculations of its value. Therefore I believe it was harder to get the bits to align as two individuals with a very different gene distribution could have the same strength. In comparison with weighted genes, there is more of a glutton approach to strength, the strongest individuals necessarily have the first bit of value 1. I ran 5 simulations in order to reduce the randomness of the observations. I also toggled the attributess to the gene, such as its length, if it was weighted or not, and also how many males competed with each other in order to reproduce.

Losing in a competition means instant death.

## Results

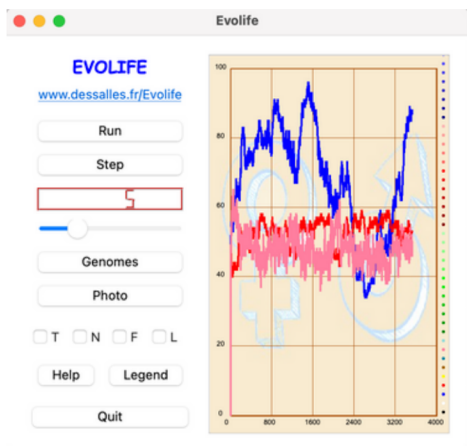


Figure 1: Simulation of evolution of strength on Evolife (1)

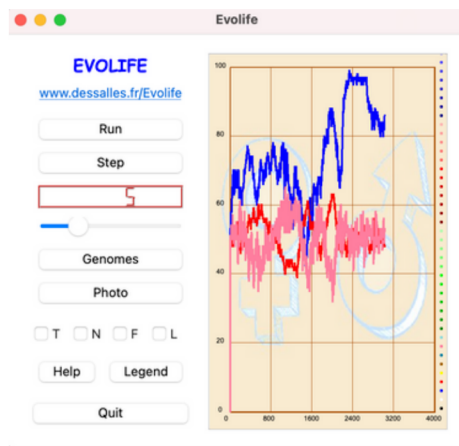


Figure 2: Simulation of evolution of strength on Evolife (2)

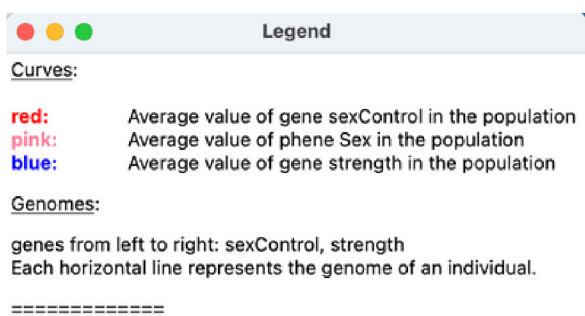


Figure 3: Legend of the graphs on Evolife  
page 78

As expected the strength reaches a value of 100% though we can observe it dramatically decreases by moments.

The ratio stayed a 1:1 during most simulations, and the sexSelection gene compensated when there was an unbalance. Indeed, we can see in figure 1 that when the sex ratio (pink) plummets, the red gene (gene that increases female births) peaks, bringing the ratio back around 1:1.

The more males there are, the more strength increases in the population. Yet after a certain point, the male population kills itself off, and the strength later follows. This does not happen systematically.

## Discussion

More literature seemed to suggest that these mechanics were mostly interesting when the ratio was biased towards females, and apart from flattening the growth, I didn't observe much differences in the strength distribution in the population.

I believe my model is not accurate as I could not find any rational explanation for the decrease of strength once it hits peak values. We can deduce it is not because of low male representation as in the first figure, we can see that the ratio is around 1:1 when strength decreases. This calls for more tuning of the model.

Once the model is tuned it would be interesting to study a "fleeing" attribute to enable males to avoid confrontation over a certain ratio threshold in order to see how that impacts the convergence of strength.

## Bibliography

Pennell, T.M., Field, J., *Split sex ratios and genetic relatedness in a primitively eusocial sweat bee*. Behav Ecol Sociobiol 75, 5 (2021). <https://doi.org/10.1007/s00265-020-02944-8>

Cockburn, A., Michelle P. Scott, & Dickman, C. R. (1985). *Sex Ratio and Intrasexual Kin Competition in Mammals*. Oecologia, 66(3), 427–429. <http://www.jstor.org/stable/4217648>

Queller D., *Sex ratios and social evolution*, Current Biology, 16(17) R664, [https://www.cell.com/current-biology/pdf/S0960-9822\(06\)01991-9.pdf](https://www.cell.com/current-biology/pdf/S0960-9822(06)01991-9.pdf)



November, 2023



ATHENS course : **TP-09**

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)

Name: **Santiago Molina, Beste Tasci**

## Exploring the Correlation between Group Segregation and Tolerance Level of the Population Emergence of Sociality: Emergence of Segregationism

November 24, 2023

### Abstract

This micro-study aims to investigate the intriguing notion that increased segregation leads to decreased tolerance, and reduced segregation inside the population leads to increased tolerance, as individuals would get used to living together with others. To test this hypothesis, we will introduce a segregation factor that dynamically influences the pre-existing tolerance model within the Evolife Segregationism framework.

### Problem

Segregation is a social factor that seems to arise in many communities in the world. The last century was one in which strives were made to be able to eliminate what can be called explicit segregation in which laws marginalized communities or individuals given the race, creed, or other factor that was inherited to their belonging to a group [Chan et al., May]. Explaining segregation is an immense undertaking as one must take into consideration many factors into consideration. Many of these factors are outside an individual's control but there are some factors that can change the dynamic of the group given by the individual.

Nowadays, in the Western world, enforced segregation is nonexistent. Nonetheless, there are still issues of some type of segregation where a group seems to be marginalized given

their creed, religion, race, or socioeconomic status. One of the most relevant scenarios in which this happens is the segregation of metropolitan cities in the United States of America [Greene et al., 2017]. Although segregation as a law has been eliminated in the country, there are still marginalized communities. It can be argued that this marginalization becomes somewhat of a cycle. People live in these areas, and since they do not interact with other members outside their homogeneous community, their tolerance for them is very low. The less they interact and cluster in their homogeneous bubbles, the smaller their tolerance becomes. This can be seen nowadays in these cities by the social mobility of different communities or the high difference in voting choice for political offices [Mehdipanah et al., 2020]. Finally, it does seem to appear on how people live and where to live. But is this going to be like this all the time? Can different factors and the passing of time affect this mobility? These are questions that must be asked when understanding.

A model that highlights individual choices is the model proposed by Thomas Schelling in 1971. This model was based on race differences in an area in the United States. We simulated this model based on the Evolife software developed by Prof. Dessalles. In this simulation, there are variables that can be altered and that are important to mention to better understand its power and limitations. The simulation starts with two groups that move randomly around the plain. The most important variable that determines this dynamic movement by individuals is tolerance. This is a factor from 0 to 100 which is determined by the difference of neighbors that an individual has in a determined radius. If the percentage of different neighbors is bigger than the tolerance, the individual will move [Greene et al., 2017]. Given different types of tolerance, there are different outcomes of segregation. For example, at very high or very low tolerance there does not seem to be segregation (see figure 1. The only time that the outcome was segregated was when the tolerance factor oscillated from 25 to 65.

Even though this model elucidates the need to take into account individual preferences, modifications can be made. The most important of which can be having a dynamic tolerance. As mentioned previously, explicit segregation has been made illegal in most of the world. But how does the level of segregation affect tolerance across time is a relevant question that elucidates the bidirectionally of a person's context and its influence. This is not something that the Schelling model takes into consideration and should be explored.

## Method

In order to make the tolerance level adaptable, we implemented a segregation score for each individual. The score is contingent upon the level of variety that individuals are exposed to in their community. A segregation score is considered high when an individual is surrounded by individuals of the same ethnicity, as represented by their color in the simulation. In this scenario, we assume that the individual is not exposed to a variety of people due to his or her familiarity with his or her neighbors from a similar cultural or ethnic background. This leads to a decline in an individual's tolerance level.

Conversely, suppose the individual's segregation score is low, indicating that individuals of different races surround them. In that case, we infer that the individual would become accustomed to living with others, hence increasing their tolerance level. Here, the dynamic tolerance level established is inversely related to the segregation score. To adjust the tolerance level of a person within the population, we employed the subsequent updating function:

$$T_{n+1} = T_n + I * (T_n - S) \tag{1}$$

where  $T_{n+1}$  indicates the updated tolerance,  $T_n$  the current level,  $S$  the segregation score, and  $I$  is the influence factor.

Segregation score  $S$  is computed using as follows:

$$S = \frac{\textit{same}}{\textit{same} + \textit{different}} \quad (2)$$

The variable *same* indicates the number of neighbors of an individual from the same color, and *different* indicated the number of neighbors of an individual from a different color. So that *same* + *different* would indicate the total number of neighbors of an individual. If an individual is completely surrounded by neighbors of the same color, the numerator and denominator will be equal, resulting in a value of zero for the variable *different*. This will result in a segregation score  $S$  of 1, which represents the maximum score. This number signifies an individual residing in a cluster comprised solely of individuals of the same color.

As determined by equation 1, each person's dynamic tolerance is computed in each iteration individually by averaging their segregation score over the past ten years. Incorporating a ten-year buffer includes the element of "experience with living with others" in updating the tolerance. The participants are subsequently relocated based on their tolerance level.

## Results

We enhanced the *Segregationism.py* script within the Segregationism App in Evolife by adding some additional functions. After these updates, we executed the program with varied tolerance levels—10, 40, and 80—and compared the outcomes between dynamic and constant tolerance scenarios. As individuals increasingly segregate by dwelling solely with those of the same ethnicity, their tolerance levels diminish. In communities with predominantly low-tolerance individuals, segregation intensifies, causing a further decline in tolerance. Consequently, clustering of different ethnicities in distinct neighborhoods occurs more rapidly than in simulations with a constant tolerance level, where initial tolerance levels are consistent. This mirrors real-world scenarios, where low-tolerance individuals may resist newcomers, leading to rapid ethnic enclaves, akin to the experiences of Turkish immigrants in 1960s Germany who clustered in neighborhoods exclusively populated by fellow Turks, hindering language acquisition.

For an initial tolerance level of 40, clustering was observed with constant tolerance. Clustering appears to occur with the dynamic tolerance level as well (see Figure 2), but when comparing the same time point between the constant tolerance simulation and the dynamic tolerance simulation, the dynamic tolerance simulation causes delayed clustering. Dynamic tolerance eventually reaches zero, preventing the system from reaching equilibrium. Individuals on the edges of neighborhood clusters remain dissatisfied and continually relocate. As individuals move based on their tolerance level calculated from the past 10 years of experience, those on the cluster edges learn to coexist more than those in the middle, leading to increased tolerance.

With an initial dynamic tolerance level of 80, segregation behavior resembled the constant tolerance simulation. Notably, initially relatively tolerant individuals became even more tolerant when surrounded by diversity, reaching equilibrium faster, with no visible clustering or segregation.

These results are intuitive, and the dynamic tolerance concept provides a potential starting point for simulating real-time cities, neighborhoods, or countries. The introduced influence

level can be tailored to demographic and educational properties, such as using average education levels as a reference to affect the influence factor. Simulations with different influence factors can represent countries with diverse education levels. One can identify the minimum average education rate needed to introduce a new community without causing clustering or segregation while boosting people's tolerance levels. And thus trying to find solutions to educate people to a certain point in a certain place. Consequently, it provides a pathway to explore strategies aimed at educating individuals to a specific level within a particular locale.

Additionally, this idea supports further enhancements through manual modifications. Tolerance can be adjusted, and peaks representing community events affecting tolerance levels can be incorporated. For instance, a terror event with a specific nationality as the perpetrator may decrease people's tolerance toward that nationality. Evaluating the long-term impact of such peak decreases on segregationism can be explored through simulations with dynamic tolerance levels and manual adjustments.

Moreover, we acknowledge that the tolerance level may exhibit more non-linear properties than our current implementation. Updating the function to incorporate additional non-linear characteristics offers an avenue for experimentation and improvement.

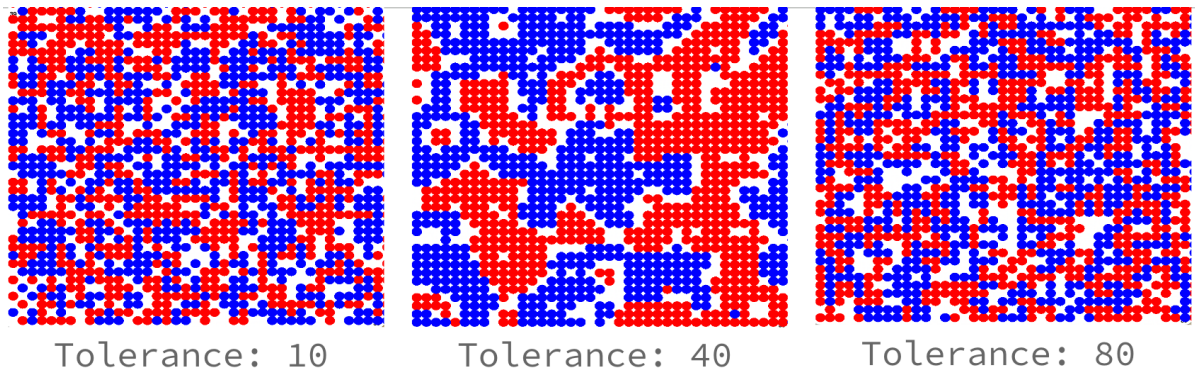


Figure 1: Results from simulations with a **constant** tolerance level at varying thresholds (10, 40, 80) after 2000 iterations.

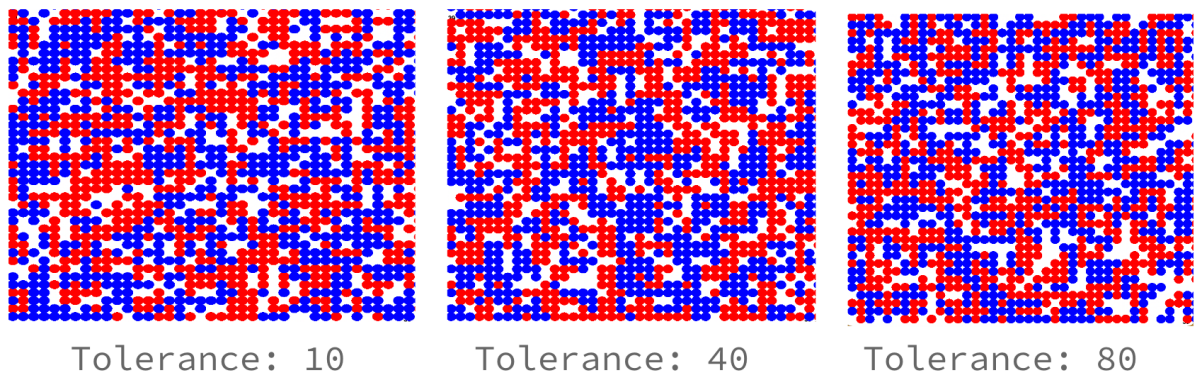


Figure 2: Results from simulations with a **dynamic** tolerance level at varying thresholds (10, 40, 80) after 2000 iterations.



## Discussion

The aim of the project was to explore how the tolerance of people changed over time. For this, as explained in the methods, a formula was created where the tolerance factor was negatively correlated to the segregation factor. Furthermore, the influence factor was introduced to give meaning to the changes in tolerance over time being ideally constructed, by variables such as education, income and historical segregation. Nonetheless, due to the time constraints of the project, these variables were not implemented.



When observed qualitatively, the simulation does show differences with the projections where the dynamic tolerance was not introduced. Especially in high tolerance, there is more dispersion between the two different groups of dots. On the other hand, the low tolerance showed no significant difference in segregation, which indicates that more specific adjustments must be made to the formula, especially to the influence factor, as this modulates the growth or decline of the new tolerance.

This project elucidates the complexity of population dynamics and how individual choice can influence the dynamic of the group. Moreover, the implementation of a dynamic tolerance based on preexistence segregating and influence factors provide insight to the bidirectional influence of the individual with its environment and the environment on the individual. Furthermore, it is important to understand certain limitations of the study. As previously mentioned the influence factor was arbitrary numbers as there was no time to create a proper formula for this variable in particular. Nonetheless, it is worth mentioning some factors that could be considered in the future such as the Human Development Index which involves a thorough calculation of many factors such as economic mobility, longevity and education [Yin et al., 2023]. If the influence factor is given specifically to a city, state or country other factors can be taken into consideration such as criminality rates and cost of living [Greene et al., 2017]. Hence, the influence factor can function as a dynamic variable, adapting to varying scenarios. It possesses the flexibility to be tailored differently when analyzing distinct population types, whether it be cities versus neighborhoods or countries versus countries.

## References

- [Chan et al., May] Chan, H., Irfan, M. T., and Than, C. V. (2020, May). Schelling models with localized social influence: a game-theoretic framework. In *AAMAS Conference proceedings*.
- [Greene et al., 2017] Greene, S., Turner, M. A., and Gourevitch, R. (2017). Racial residential segregation and neighborhood disparities. *Washington, DC: US Partnership on Mobility from Poverty*.
- [Mehdipanah et al., 2020] Mehdipanah, R., Bess, K., Tomkowiak, S., Richardson, A., Stokes, C., White Perkins, D., Cleage, S., Israel, B. A., and Schulz, A. J. (2020). Residential racial and socioeconomic segregation as predictors of housing discrimination in detroit metropolitan area. *Sustainability*, 12(24):10429.
- [Yin et al., 2023] Yin, R., Lepinteur, A., Clark, A. E., and D'ambrosio, C. (2023). Life satisfaction and the human development index across the world. *Journal of Cross-Cultural Psychology*, 54(2):269–282.



 	<p>December, 2023</p> <p style="text-align: center;"><b>Emergence in Complex Systems</b></p> <p style="text-align: center;">Micro-study</p> <p style="text-align: right;"><a href="http://teaching.dessalles.fr/ECS">teaching.dessalles.fr/ECS</a></p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Name: **Domenico Muscillo**

---

---

## **Cocktail party phenomenon optimization**

### ***Abstract***

The Cocktail party phenomenon is an emerging behavior of agents living in a scenario in which although no one is actively trying to speak louder than than normal everyone is forced to scream at the end in order to bypass the noise due to people in the other parts of the room.

### ***Method***

With this small project I tried to put some effort into developing a more realistic simulation of the analyzed scenario.

### ***Discussion***



Among the developments I chose to analyze what happens if people tend to come closer to their interlocutors. To do so I defined a conversation radius within which people that are discussing may attract other individuals of the party. What emerges is, as expected, the creation of spontaneous conversational groups, and the bigger the radius set as parameter the fewer and bigger these circles get.

One other improvement is to impose that in each conversational group there is, at each time, one speaker only, as it happens naturally in common conversations. This should decrease the perceived noise because only few speakers talk at the same time in the room, but it would not prevent the phenomenon to happen. We achieved a slower saturation of the room to noise.

I furnished the simulation of a probability for each participant to get bored by the conversation in which he is into and to wander off the room in search for another group to join. This, unexpectedly, has accelerated the speed of the noise to fill the room. One explanation to this could be that, once a person leaves the group and chooses one another it

decreases the distance between people in the first one and the second one, making it more likely for other members to do the same swap, giving positive feedback to the same process. This ends up in the amplification of the noise since the groups become more affected by each other's noise.

In the end I also implemented an attractor point (like a cocktail table or the toilettes) to which people have to go from time to time. This was supposed to give variability in the formation of the conversational groups since people close at the cocktail table would engage in a conversation by themselves. Instead the result was a huge queue of people wanting to reach the attractor, forbidding those who have been served to go back to their original groups, and preventing this way also the others to make any progress. Since all the people would end up together in the queue the noise increased drastically.

 	<p>November, 2023</p> <h2 style="text-align: center;">Emergence in Complex Systems</h2> <p style="text-align: center;">Micro-study</p> <p style="text-align: right;"><a href="https://teaching.dessalles.fr/ECS">teaching.dessalles.fr/ECS</a></p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Name: **Rios Maël**

---

---

## 3D Cellular Automaton and Visualization on Evolife

### **Abstract**

This project focused on implementing a three-dimensional cellular automaton, which means new cellular update rules and new visual representation of the cellular set through Evolife. I will explain in the following paragraphs how I implemented it in detail.

### **Problem**

The same way we can find some surprising solutions and visuals with one-dimensional cellular automata and the different neighbors' ruleset, I wanted to define some for three dimensional evolving cellular structures to see if we could obtain similar results and spectacular patterns. This problem means having to redefine the neighbors, the ruleset and the visualization in order to be able to answer the main question of what would happen if these rulesets were generalized to three dimensions, and what would the main limitations of such a port be?

### **Method**

I mainly refactored the Evolife cellular Automaton python file for it to work with a three-dimensional cellular set:

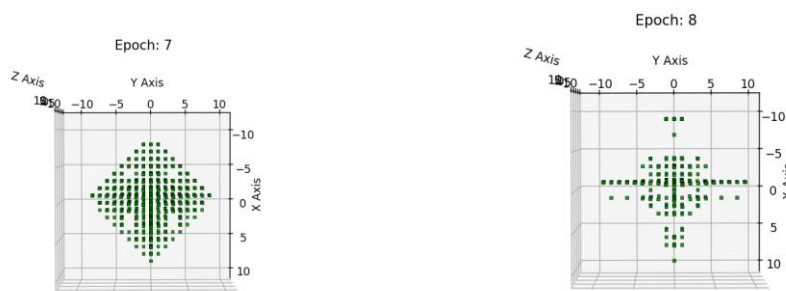
- Instead of being a one-dimensional array, the cells are now a three-dimensional array with a fixed maximum size (for calculation time purposes and visualization).
- We update the cells in the array depending on their neighbors, and I used Von Neumann's definition of neighbors, which means each cell has 4 neighbors unless we are at an edge, corner, or face of the cubic array.

- Depending on how many alive neighbors (whatever their symmetry may be) a cell has, the algorithm chooses to resurrect it, kill it or keep it alive depending on the rule we specified.
- For the rules, I imagined a 7 bit encoded rule (so rules go from 0 to 127) where each bit in its reversed position (so we go from 0 to 6 from left to right) represents the alive/dead state of the cell depending on the number of neighbors being exactly the same as the position of that bit (so what happens when we have exactly 0, 1, 2 neighbors etc.). I coded the rule this way because if we had to consider the positioning of the neighbors and not only how many of them are alive, but that would also create a lot of different rules, too many to mean something significant. I was inspired by the game of life with the ratio of alive/dead neighboring cells.
- For initialization, we must enter in the Evolife starting window a three-dimensional array representing a small cube (the size of the cube has to be smaller than the maximum size of the whole array) and with odd length (it must have a center). This means we can choose how our cube starts like from the center going towards the faces.
- For visualization, as I couldn't port the visuals to the observer, I used matplotlib to freely be able to look around the 3D shape to get better visuals out of the solutions.

## Results

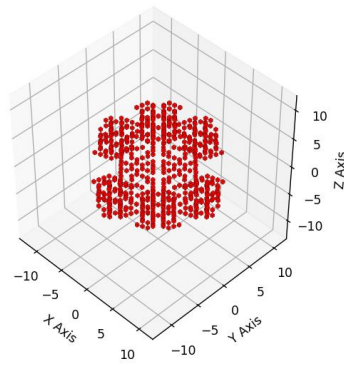
After finishing the implementation, I tested the program with different interesting rules (since my rules do not consider the initial state of the actual cell for its next step, uninteresting rules are for example rules having a 1 on the 0 position of the rule, since at step 1 almost all the empty grids will fill itself instantly) and interesting initialization arrays.

We have some interesting shapes forming such as cross-shapes or pyramids and diamonds. Although the 3D visualization is great with matplotlib, I think it would be interesting to also code a functionality to be able to visualize 2D cuts inside the shape along a given axis.




---

[[[1,1,1],[1,0,1],[1,1,1]],[[0,0,0],[0,0,0],[0,0,0]],[[0,0,0],[0,1,0],[0,0,0]]] initialization with rule 18.



[[[0,0,0],[0,1,0],[0,0,0]],[[0,0,0],[1,1,1],[0,0,0]],[[0,0,0],[0,1,0],[0,0,0]]] initialization with rule 2.

## **Discussion**

The results are what I would have expected with the 3D cellular automata, but there are some details I could have explored to make the final project more complete.

First, it would be interesting to find the best rules or the most surprising rules in the ruleset I defined earlier, since I wasn't able to explore as deep as doing a few tests and checking everything worked correctly.

Then, it would be interesting to add a bit that represents the change of rule depending on the initial state of the cell, as it would greatly impact the outcomes of the cellular growth and make some rules that were not interesting before worth the shot (like rule 3 for example). This would also be logical for my implementation to be closer to the existing one-dimensional cellular automata that considers this state of the cell.

Also, I would like to code the Moore neighborhood along with the von Neumann neighborhood (maybe a choice to make in the Evolife control panel?) to see the different results it could also give compared to the one I implemented for the project. I feel like there are a lot of interesting shapes I was not able to directly reproduce with this first implementation, but that are doable with more time and exploring these details I laid down.

## **Bibliography**

- Tyler, Tim, [The Moore neighborhood](http://cell-auto.com) at [cell-auto.com](http://cell-auto.com)
- [Cellular Automata - Isometric 3D von Neumann neighbourhood \(cell-auto.com\)](http://cell-auto.com)





ATHENS course : **TP-09**

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)Name: **Léo Roux**

## Exploring gender-based success disparities in dating apps

### Abstract

Dating applications have been raising in popularity over the past decade. In this python dating app simulation, we explore gender-based success differentials, shedding light on potential disparities between men and women. The project reveals insights into the dynamics of online dating, providing a nuanced perspective on the challenges and opportunities faced by users of different genders.

### Problem

The problem addressed in this project revolves around the disparities in success rates experienced by men and women on dating apps. Surveys and studies have shown a noticeable behavioural changes among frequent users of dating apps [2][3]. People, especially men, described a certain frustration concerning the amount of like and matches they got on their apps. This investigation aims to uncover whether there are discernible differences in outcomes such as likes, matches, or overall success based on gender. This issue is critical as it delves into the dynamics of online dating, highlighting potential challenges or advantages that users of different genders may encounter, ultimately contributing to a more comprehensive understanding of the intricacies within these platforms.

### Method

To answer this problem, I used as a starting point a simulation made by Luis Calejo [1] on python. His simulation was created to answer the question of the disparities of matches. We

can view it through a broader scope, where the whole distribution of matches is a subject to study. I adapted it to fit in Evolife program. The simulation relies on the following objects:

- an "App" where the simulation take place
  - centralizes the exogenous parameters such as the number of users, the "MenToWomenRatio" and other relevant values.
  - a simulation function (on which Evolife program iterates). This function generate a set of users and make them interact :
    - \* each user "swipes" to a fixed and common number of profile
    - \* for each profile, with a certain probability (depending on each gender) the user may give it a like or not.
    - \* if it happens that this other profile has already liked the user, there is what we call a "match".
    - \* the function then send the data gathered to the "StatProcessor"
- an "User" class containing relevant variables and method for each agent
  - its gender (we can imagine for future improvement implementing the age, the localization, etc.)
  - its "attractiveness". This is my less favorite point of the simulation as it is the less real-life based variable (as it is basically a random number between 0 and 1
  - the "swiping function", in which an agent "decides" (by randomness) whether it likes or not another profile.
- a "StatProcessor" that compute and display the relevant values to study (means, medians, distributions of likes, matches...)

The whole is implemented as an App folder (to insert the Evolife/App directory). The parameters to make vary are :

- NbUsers : the number of users on the app
- ProfilePerDay : the amount of profiles seen each day per agent
- MenToWomenRatio : The proportion of men over women
- LikePercentageMen : The probability for a Man to like a woman profile
- LikePercentageWomen :The probability for a woman to like a man profile

## Results

Through several experiments, the results indeed showed some disparities, but only inside a population of the same gender. I tested the code with different sets of parameters. The program displayed both distributions, which are the amount of individuals from the "MALE" gender distributed along their respective quantity of matches.

On both curves, a minority of males have the higher proportion of matches. I didn't included in it but the distributions for the women follow the same tendencies.

It is interesting to see that the first set of parameters result in a weird distribution with a "stage" before it goes near to zero.

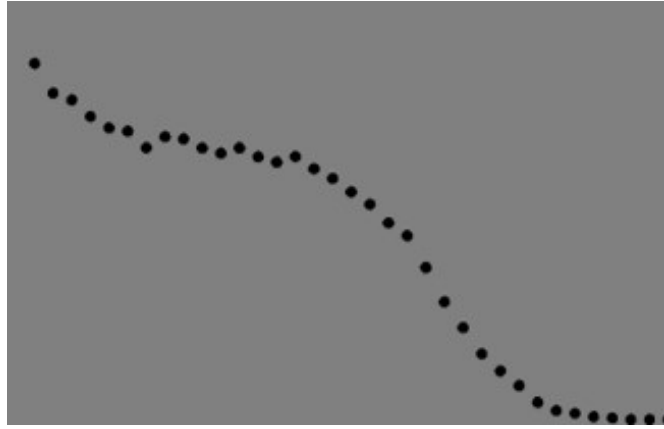


Figure 1: Distribution of men per matches number for exp.1

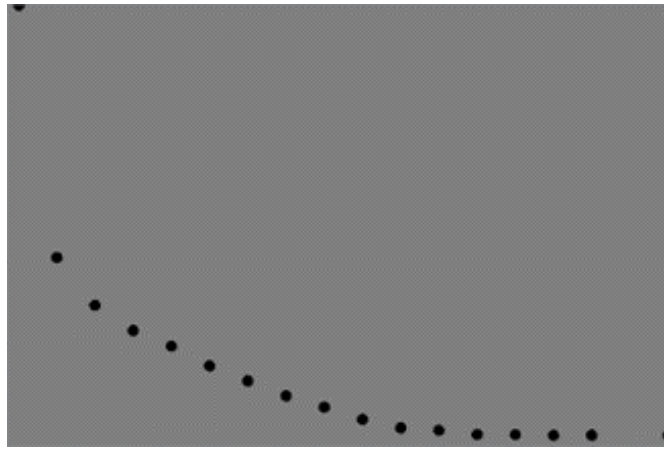


Figure 2: Distribution of men per matches number for exp.2

## Discussion

The simulation was originally made to show disparities between genders but ended up producing disparities inside of the same gender population. It is quite logical because there is a symmetry between both genders (except their quantities of users). However, we can see that particular distribution may emerge from the simulation.

It would be very interesting to include a more precise notion of attractiveness. Instead of it being random and most of all absolute, it would be interesting to generate relative attractiveness, more like a (Manhattan) distance on a  $n$  dimensions vector space. In that case, the attractiveness would be based on several factors (several personality traits, taste for certain field, on a scale of 0 to 1).

## References

- [1] Luis Calejo. *Dating apps simulation*. URL: <https://github.com/LuisCalejo/dating-apps>.
- [2] Jessica Strubel and Trent A. Petrie. “Love me Tinder: Body image and psychosocial functioning among men and women”. In: *Body Image* 21 (2017), pp. 34–38. ISSN: 1740-1445. DOI: <https://doi.org/10.1016/j.bodyim.2017>.

02.006. URL: <https://www.sciencedirect.com/science/article/pii/S1740144516303254>.

- [3] Shangwei Wu and Daniel Trottier. “Dating apps: a literature review”. In: *Annals of the International Communication Association* 46.2 (2022), pp. 91–115. DOI: 10.1080/23808985.2022.2069046. eprint: <https://doi.org/10.1080/23808985.2022.2069046>. URL: <https://doi.org/10.1080/23808985.2022.2069046>.

November, 2023



ATHENS course : TPT-09

# Social Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/SECS](http://teaching.dessalles.fr/SECS)

Name: Maciej ŚWIĘCH

---

Development of the simulation in the Evolve "Ants" application with the possibility to add constraints to ant colony movement and negative optima to the environment

November 28, 2023

## Abstract

I worked on development of Evolve "Ants" application. The application was developed with the ability to locate negative local optima in the environment (opposite to food sources, "poison sources") and the ability to locate constraints on which ants cannot stand.

## Problem

Ants application provides user with an easy-to-use interface to test and tune parameters of ant colony algorithm. However, the cases, that it allows to test are rather simple. In the original version they consist of  $n$  local maxima. In order to allow students to visualise and test their algorithms on in more complex environments, I added a possibility to locate local minima in the environment, as well as to locate constraints on where the ant colony can stand in the environment.

## Method

The obstacles and poison sources can be added programmatically in the code. The code has been comprehensively commented to make it easier for users to modify the parameters and

test their solutions. The parameters are defined within the *main* function. The declaration is presented in the following snippets of code.

Listing 1: Definition of obstacles parameters

```
% Number of obstacles
NbObstacles = 2
# maximum circuit of an generated obstacle
ObstacleCircuit = 20
# minimal length/width of an generated obstacle
ObstacleMinWidthLength = 3
```

As presented on lst.1, users can define how many obstacles will be created in the environment by setting the value of *NbObstacles* to a desired value. If not stated otherwise (explanation in further section), the obstacles are generated randomly. The obstacles are generated within the area between the ant colony and food sources. Every obstacle is a rectangle made of dots. The minimum width/height of an obstacle is 3 units. The maximum circuit is 20 units. At first the rectangles width is generated. Then, taking under consideration the "left" part of the circuit, the height is generated. To generate those values, I used *randint()* function from Python's *random* package.

Obstacles are instances of *Obstacle* class, that was defined analogically to *FoodSource* class.

Listing 2: Creation of random obstacles

```
obs_x_location = random.randint(
    int((fs_x_location_landsize_coefficient + 0.1) * LandSize),
    int((ants_x_location_landsize_coefficient - 0.1) * LandSize))

obs_y_location = random.randint(min(fs_y_locations),
    max(fs_y_locations))

y_axis = range(max(ob_center[0]-(ob_height//2), 1),
    max(ob_center[0]+(ob_height//2), 1+ob_height//2))
x_axis = range(max(ob_center[1] - (ob_width // 2), 1),
    max(ob_center[1] + (ob_width // 2), 1 + ob_width//2))
xy_points = itertools.product(x_axis, y_axis)
```

As shown in 2, centers of obstacles are generated within the area between the food sources and an colony in both axes (*obs\_x\_location*, *obs\_y\_location*). *y\_axis* and *x\_axis*, define the sets of point that are the height and width of the obstacle. Those two vectors are multiplied, and create the area matrix *xy\_points*.

Listing 3: Definition of poison sources parameters

```
# setting the number of poisons
NbPoisons = 2
# Setting the strength of poison released in the poison source
# This value indicates how many negative pheromones, an ant will
# leave, when it reaches it
PoisonStrength = 10
```

As presented on lst.3, users can define how many poison sources will be created in the environment by setting the value of *NbPoisons* to a desired value. Similar, as with the obstacles, if not stated otherwise, the poison sources are generated randomly within the area between the ant colony and food sources. The default radius of poison sources is 5 (2,5x greater

comparing to food sources). *PoisonStrength* defines how many negative pheromones will be subtracted from the LandCell where a poison source is located, when an ant steps on it.

Listing 4: Creation of random poison sources

```
# Poison sources are generated (until they don't
lie in the area of an obstacle)
while ps_centre_in_obstacle_area:
    ps_x_location = random.randint(
        int((fs_x_location_landsize_coefficient + 0.1)
            *LandSize),
        int((ants_x_location_landsize_coefficient - 0.1)
            *LandSize))
    ps_y_location = random.randint(min(fs_y_locations),
        max(fs_y_locations))
    if (ps_x_location, ps_y_location) not in obstacle_area:
        ps_centre_in_obstacle_area = False
```

Poison sources are generated randomly, similarly to obstacles. The only difference is the constraint, that they should not be located within the area of any existing obstacle (lst.4).

Listing 5: Food parameters

```
# Setting the amount of food that is added to the collected
food, when ant steps on a food source
FAdd = 1
# Setting the amount of food that is subtracted to the collected
food, when ant steps on a poison source
FSubtr = 2
# Initial quantity of collected food
FoodCollected = 100
```

In lst.5, food parameters are defined. The *FoodCollected* defines the current amount of food collected by the colony. It may be useful to set it to non-0 to see that at the beginning of a simulation, ants may lose some food, due to stepping on the poison sources. *FAdd* defines how much food the colony gets, when an ant steps on a food source. *FSubtr* defines how much food the colony loses, when an ant steps on a poison source. The total amount of food collected by the colony is never  $< 0$ .

Listing 6: Test mode/Random mode

```
# Test obstacle and poison placement (yes/no)
# Test placement is deterministic (obstacles and poisons are placed
# in the same locations in every execution)
Test = True
# if test, the number of obstacles equals the number of poison
sources and = 2
# then their placement is deterministic. The placement coefficients
# of poison sources and obstacles are defined as below:
# poison source 1 (x, y):
ps_1 = (0.2, 0.375)
# poison source 2 (x, y):
ps_2 = (0.2, 0.575)
# poison sources centers
p = (ps_1, ps_2)
```

```

# obstacle 1 (x, y):
o_1 = (0.35, 0.5)
# obstacle 2 (x, y):
o_2 = (0.45, 0.4)
# Obstacles centers
o = (o_1, o_2)
# Food sources and ant colony are always spawned in the same
locations.
# Regardless of the "Test" parameter. In order to switch into
default behaviour
# user can switch the value of parameter DefaultFSColonySpawn
to True
DefaultFSColonySpawn = False
# The parameters below allow users to set constant spawn point
for the ant colony and food sources
# Center of spawn for ant colony
ac = (0.6, 0.5)
# Center of spawn fo food sources (the food sources will be located
symmetrically along the vertical line ,
# with an offset of 0.05)
fs = (0.1, 0.5)

```

The simulation can be run in test mode or in random mode. Test mode is useful to observe the behaviour of an algorithm in a deterministic environment. In test mode, obstacles and poison sources are always located in the same LandCells (as shown in lst.6).

## Results

### 0.1 Explanation

Ants are spawned in an environment. In the environment there are obstacles, poison sources and food sources. The goal of the colony is to collect as much food as possible.

At the beginnig, ants are wandering in random directions. When an ant reaches the food source, it takes the food ant goes back to the colony. On its way back, it releases positive pheromones to show the other aths, that her path might by optimal, because following it, will lead to the food source.

Ants "sniff" to find out which direction is the best to follow. There is also a randomization factor included in order to keep a reasonable ratio between exploitation and exploration of the state space.

If an ant steps on the poison source, the colony looses a set amount of food. Ant immediately releases a set amount of phereomone in the area of the poison source in order to "tell" the others, to avoid this area.

Obstacles are just areas in environment, where no ant can stand.

Food sources can symbolise the minima and poison sources the maxima in a minimisation problem with constraints.

### 0.2 Tests

Examples of deterministic and non-deterministic test environments are shown in fig.1 and fig.2 (pink - obstacles, red - poison sources).



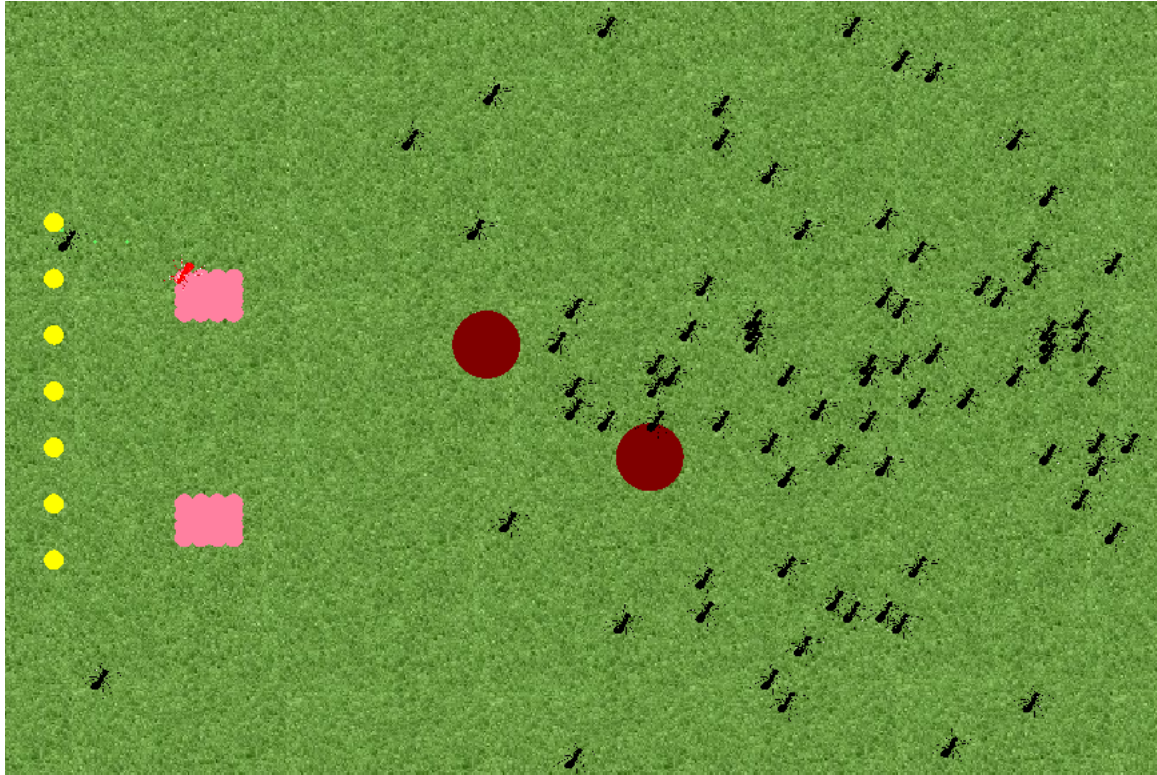


Figure 1: Deterministic environment (Test = True)

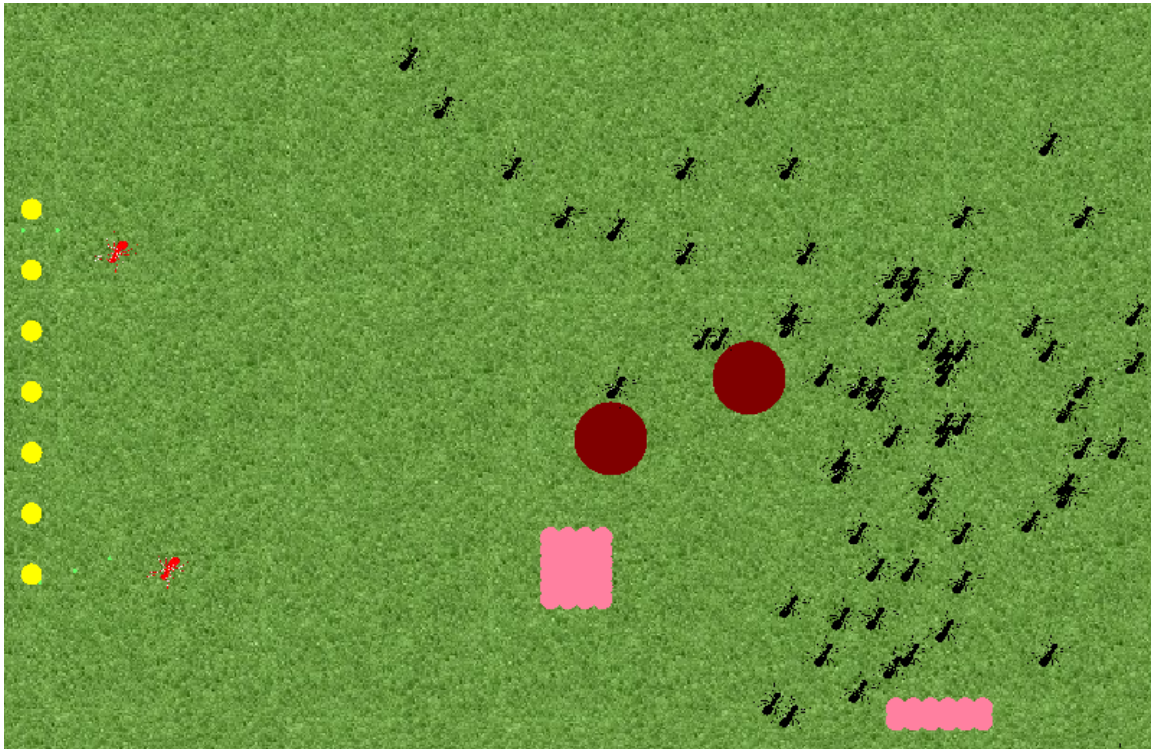


Figure 2: Non-deterministic environment (Test = False)

Tests were carried out to observe the behaviour of the colony in new environment. Test environment:

1. Food sources: 7

2. Poison sources: 2
3. Obstacles: 2
4. Ants: 100
5. LandSize: 100
6. Initial food quantity: 100
7. FAdd = 1
8. FSubtr = 1
9. Environment: Deterministic

Negative pheromone coefficient (the amount of pheromone left by an ant in the poison source when it steps on it) was the parameter, that's influence on the colony's performance was tested.

The graphs (fig.3, fig.4, fig.5, fig.6, fig.7) show the amount of food collected by the ants. Decreases in the values on the y-axis indicate that the ants visited a source of poison, while increases indicate that they visited a source of food.

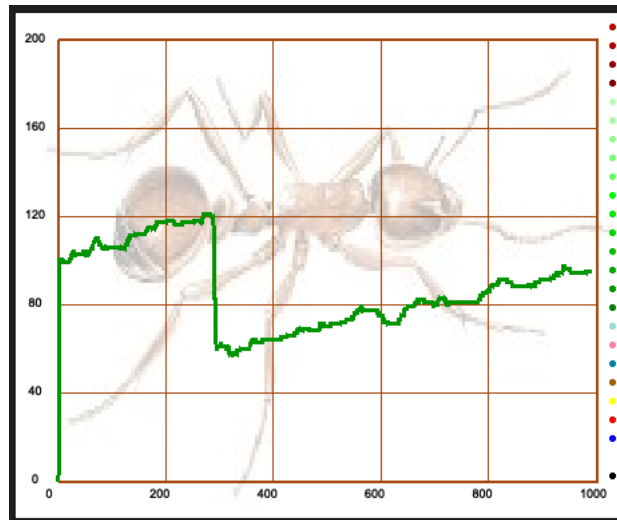


Figure 3: Amount of food accumulated by the colony with Negative Pheromone Coefficient = 1

As shown in the figures fig.3, 4, 5, 6. the algorithm learns to avoid sources of poison more quickly with higher coefficient values. For coefficient = 1, the colony only started to learn that it should avoid the poison around step 300. For coefficient = 10, it was much faster, around 100 steps. For higher values ( $20 >$ ) no clear negative peaks are actually noticeable. Interestingly, for a very high value of the coefficient (100) the algorithm was slow. This may be due to a very significant caution on the poison.

## Discussion

The results of the tests are close to expected. With the low values of negative pheromones that the ants left behind when entering the poison source, the colony took a long time to learn that it should not step on it. This manifests itself in a prolonged loss of food over time. When the negative pheromone values left by the ants were high, the colony learned quickly to avoid them. The problem I encountered is that the obstacles are areas where ants should never enter. This functionality is strictly hardcoded. In tests carried out, however, it was found that such a phenomenon occurs when the ants return to the colony, having reached the

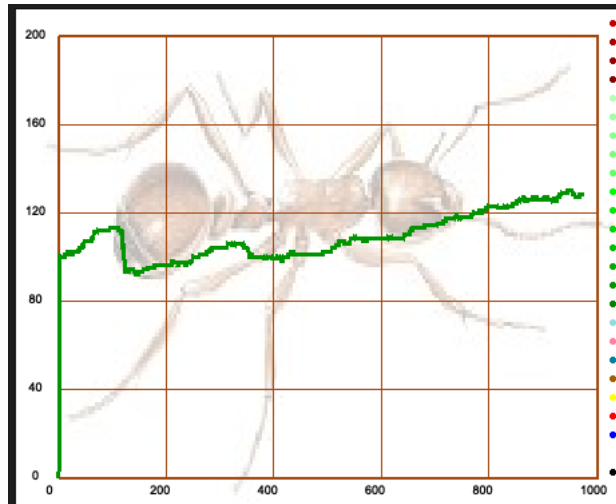


Figure 4: Amount of food accumulated by the colony with Negative Pheromone Coefficient = 10

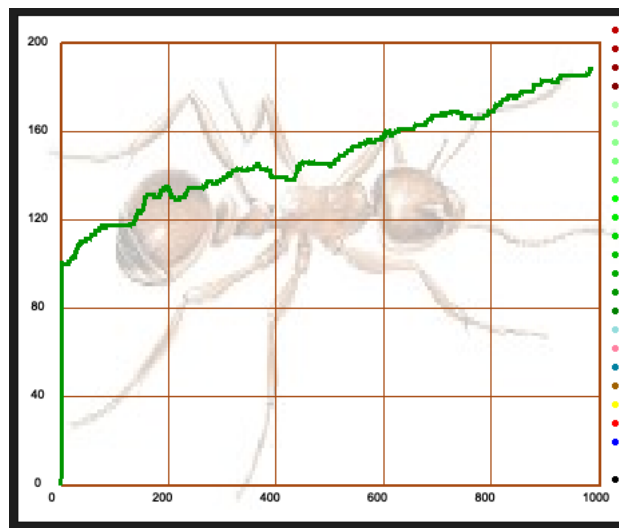


Figure 5: Amount of food accumulated by the colony with Negative Pheromone Coefficient = 20

food source.

Summing up, I think the added functionalities develop the application with the ability to test ant algorithms in more complex environments with local minima (food sources), maxima (poison sources) and constraints (obstacles). Among the possibilities I see for the development of the application: the possibility to generate obstacles while the program is running, setting different amounts of poison/food in the sources.

## Bibliography

<https://www.sciencedirect.com/topics/engineering/ant-colony-optimization>

<https://towardsdatascience.com/the-inspiration-of-an-ant-colony-optimization-f377568ea03f>

<https://docs.python.org/3/library/random.html>

<https://evolife.telecom-paris.fr>

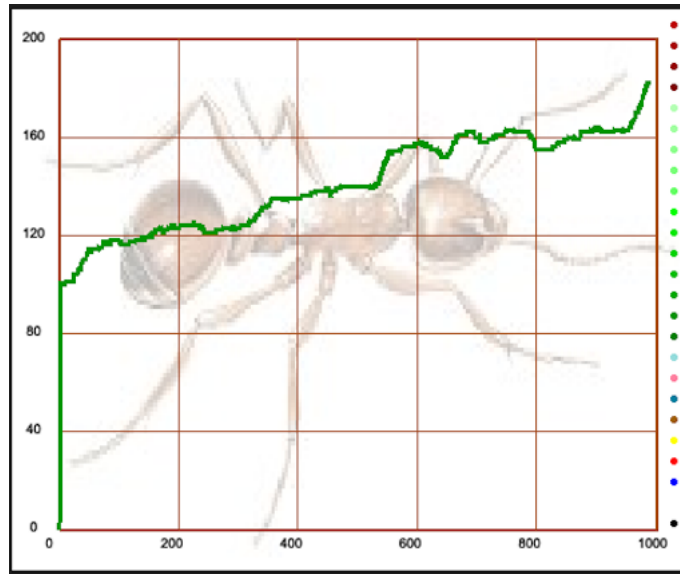


Figure 6: Amount of food accumulated by the colony with Negative Pheromone Coefficient = 50

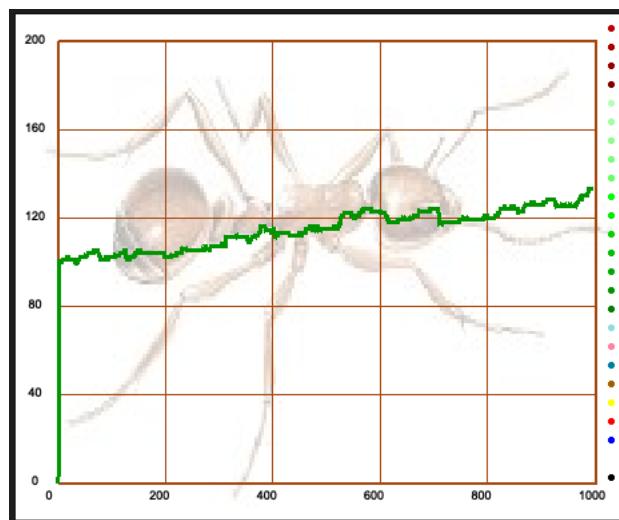


Figure 7: Amount of food accumulated by the colony with Negative Pheromone Coefficient = 100

ATHENS course : **TP-09**

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](http://teaching.dessalles.fr/ECS)Name: **Raphaël THIREAU**

---

## The regulator's role in regulating polarized public opinion

### Abstract

The "filter bubble" problem is growing due to the growing impact of social medias in our lives. Despite that, we may want to fight against this isolation because it might polarize the public opinion, lead to tension and violence and reduce diversity. This is why I want to study the impact of several strategy to homogenize the "social distribution" over different features. I also will try to implement the "filter bubble" experience in more than two dimensions.

### Problem

The social bubble effect of preference algorithms such as those used by social networks is problematic. It implies a polarization of ideas and opinions. This is, at best, a loss of diversity and, at worst, a threat to democracy, which requires intelligent, non-violent debate. This is why public action may be necessary to erase this separation effect. Also, as public opinion is diverse, we need to be able to influence more than just 2 parameters. That's why an implementation of social bubble simulation of any dimension is desirable.

### Method

To generalize the simulation code to any dimension, we had to modify the `LandCell()` and `Landscape()` classes in the `Graphics/Landscape.py` file. This involved changing coordinate tuples to a list of size  $N$ . This meant adding quite a few loops to go through the indices. Then, as the dimension is not known in advance, recursive functions were needed to, for

example, traverse the entire  $N$ -dimensional matrix representing the grid. As for Bubble.py, all the code had to be changed to make it compatible with the new landscape format. Also, a user-modifiable dimension parameter had to be added to the starter window. Normally, this implementation should have worked, at least with the  $N = 2$  case, but I couldn't make it work. I also had to change the Bubbles.evo file to add this new parameter. Next, I had planned to implement a "public" action, such as a stimulus centered on 0 that would affect everyone. For example, the public school ideally has this role. It should provide a common base of knowledge and values for all children in the same country. Another idea I wanted to implement was to modify the original distribution on the grid, for example with a Gaussian centered at 0 for all coordinates. These improvements should have enabled me to provide at least a partial answer to the problem I set out above.

## Results

Unfortunately, I haven't seen any results. In fact, I modified the Bubble.py and Landscape.py files to generalize the 2-dimensional grid to  $N$  dimensions. The simulation seems to run, but nothing is displayed even in the case of  $N = 2$ , which should produce the same result as the original simulation. As the experiment didn't work, I wasn't able to test all the other features I wanted to add (described in the Method section).

## Discussion

One of the limitations of this experiment is its computing and storage capacity. Indeed, for a grid of dimension  $N$  and size 100 (as in the basic example), the data size will be  $100^N$ . This figure can quickly become too large. When it comes to the radius of influence of different stimuli (posts on social networks, demonstrations, advertisements...) and the influence of the neighborhood, it would be essential to carry out real studies to quantify them.





December, 2023

# Emergence in Complex Systems

Micro-study

[teaching.dessalles.fr/ECS](https://teaching.dessalles.fr/ECS)

Name: **Benedikt Wimmer**

---

---

## War of Attrition

### Abstract

Most symmetric games such as Rock, Paper, Scissors or the War of Attrition [3] game have only a symmetric Nash equilibria. This study will modify the War of Attrition game in order to evaluate the emergence of asymmetric Nash equilibria and therefore the emergence of two different strategies in the same population.

### Problem

When analyzing simple symmetric interaction games, such as Rock, Paper, Scissors one finds, that most emergent strategies converge to a mixed Nash equilibrium, where the whole population plays the same specific strategy. This is however not something we see in nature. For example, in a captive squirrel monkey population an emergent hierarchy can be observed where the very top of the hierarchy is occupied by very strong and aggressive males.[1]

This work tries to gain a better understanding of this phenomenon by modifying the pay of matrix of the War of Attrition. The payoff matrix of the War of Attrition Game can be defined as follows, where A and B describe the willingness of entity a and b to fight. The individual whose willingness to fight is higher gets a payout V and if the willingness is the same both receive a payout, that is half the total payout. Each individual has to pay a price for each round that is thought. Note that only the minimum of A and B have to be paid by each individual, because once one entity has given up to fight, the other one does not have to fight anymore.

	$A < B$	$A = B$	$A > B$
Payoff a:	- A	- A + 0.5V	- B + V
Payoff b:	- A + V	- B + 0.5V	- B

## Method

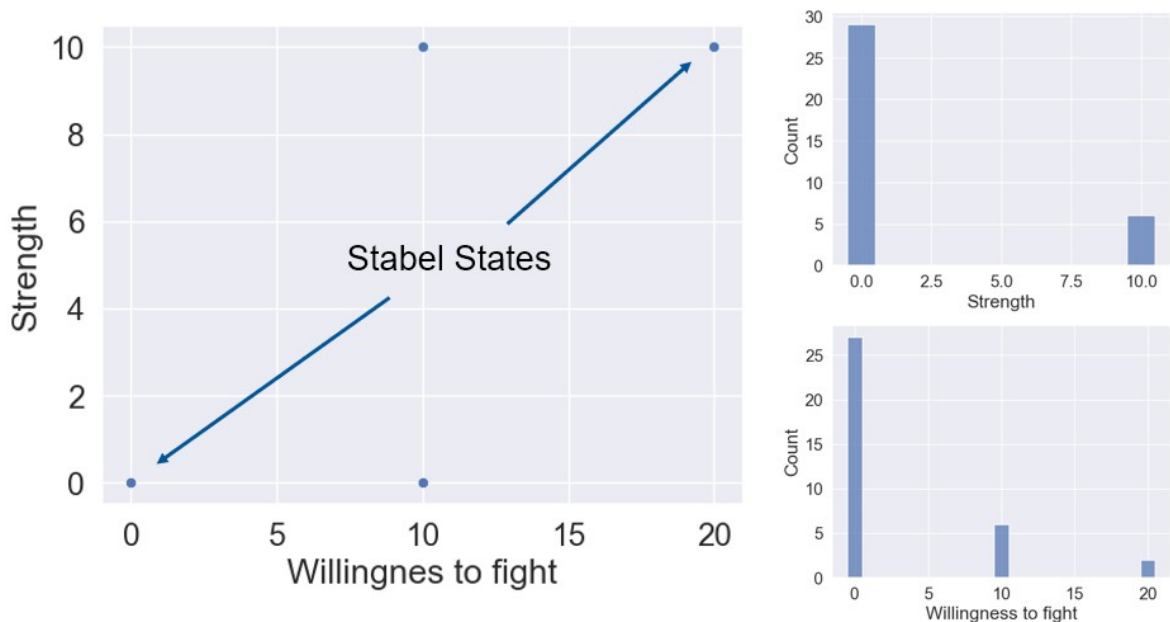
I modified the original payoff matrix, to include a second gene, which can be interpreted as strength, which gives the individual a better change of winning the encounter. However, maintaining this strength does come at a cost and each individual has to pay a constant amount for each interaction. With X and Y representing the strength gene of individual a and b the modified payoff matrix looks as follows:

	$A * X < B * Y$	$A * X = B * Y$	$A * X > B * Y$
Payoff a:	$- A - X$	$- A + 0.5V - X$	$- B + V - X$
Payoff b:	$- A + V - Y$	$- B + 0.5V - Y$	$- B - Y$

To analyze the properties of this new game, I created an evolutionary simulation using Evolife using a very similar setup to the Hawk Dove Dilemma.

## Results

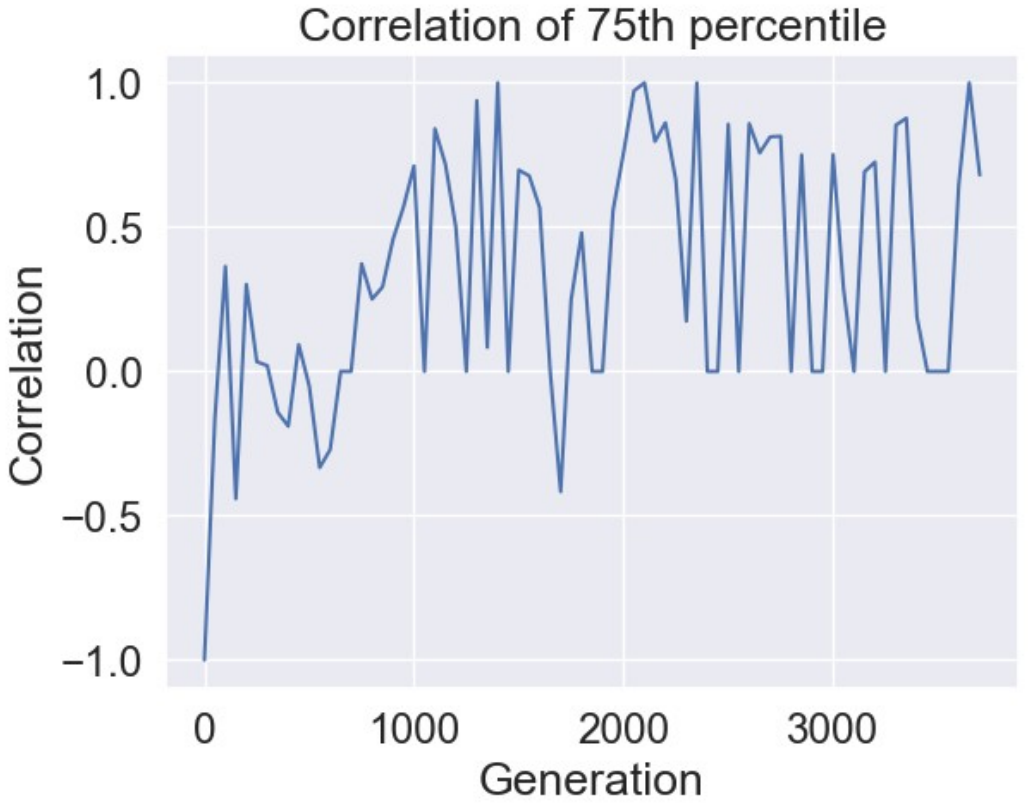
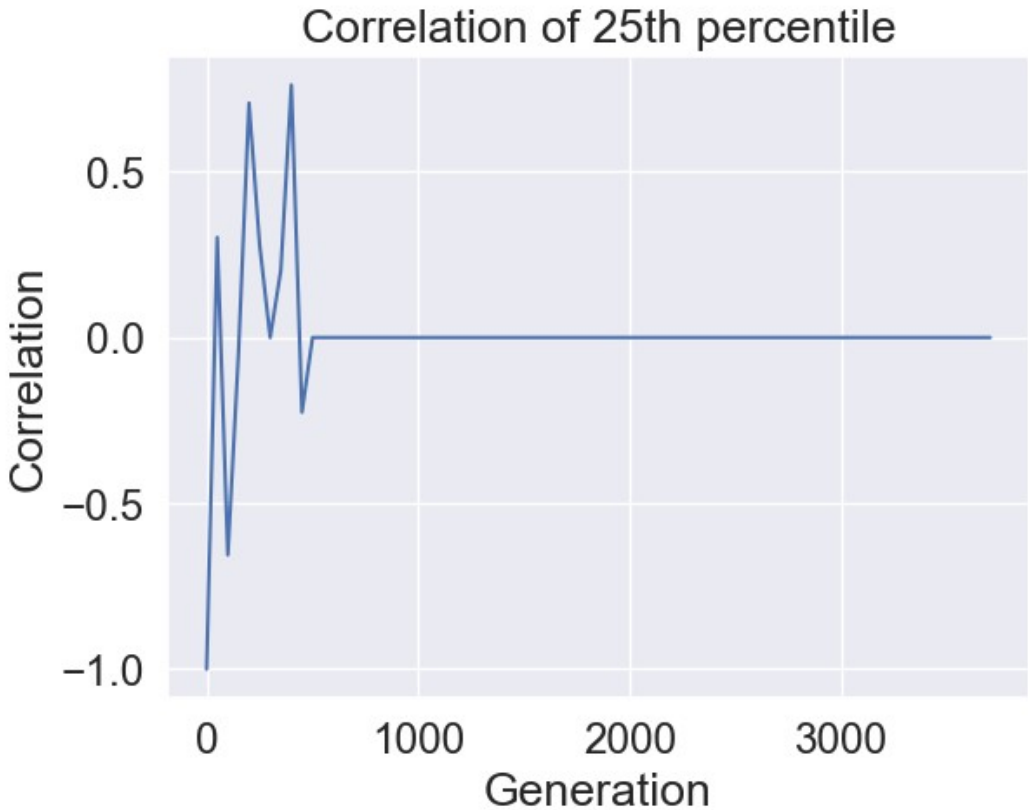
Initially one sees the emergence of two stable states in the corners of the scatterplot. The left bottom one being a population which is not strong, but also not willing to take any risks. In the top right corner, we can see a strong population, with a higher willingness to fight. Furthermore, there are also a couple of states between these two extremes, which are probably the result of the crossover between the two stable states. The weak population is also a lot more numerous than the strong population, which is to be expected, because if the population of strong individuals would be too high, they would be too likely to encounter each other and fight to their death.



Furthermore, I computed the correlation between the lower 25th percentile of the individual's willingness to fight with the strength gene. As you can see after the initial phase no correlation can be observed between those two genes, because both are zero. On the other hand, when looking at the 75th percentile one can see a very strong correlation between the



two genes. However, you can see a lot of oscillation, which is caused by two strong individuals meeting each other and fighting to there death.



## **Discussion**

I was able to achieve a splitting of the strategies, but there appears to be limitations, since the strong aggressive population is not always stable and can die out from time to time, which is probably caused by a too small population size, where it is possible for all the aggressive individuals to meet each other and kill each other.




Surprisingly the strong aggressive population is not using the max strength and is not willing to fight for an infinite amount of time. Which makes sense, since there is still a high cost involved, when meeting other strong individuals and it might make more sense to cut their losses instead of continuing an already lost battle, in terms of the final payout.

This work clearly lacks a strong fundamental game theoretic analysis of this new payoff matrix, which could be the focus of future investigations. The current literature about the war of attrition does not fit the current setup very well, because most model the willingness to fight as a probability and not as a fixed number of rounds. [1]

Another interesting direction is the introduction of locality between the interactions. So that we have distinct groups within the whole population. This would be interesting to see, because this would give a more realistic modelling of primate groups, where each localized group has a single leader. This would allow the study of how groups leaders are challenged by internal and external competitors.

## **Bibliography**

- [1] Pinheiro, T., & Lopes, M. A. (2018). Hierarchical structure and the influence of individual attributes in the captive squirrel monkey (*Saimiri collinsi*). *Primates*, 59(5), 475-482.  
<https://link.springer.com/article/10.1007/s10329-018-0668-5>
- [2] Levin, J. (2004). Wars of attrition. *Lecture Note*. <https://web.stanford.edu/~jdlevin/Econ%20286/Wars%20of%20Attrition.pdf>
- [3] [https://en.wikipedia.org/wiki/War\\_of\\_attrition\\_\(game\)](https://en.wikipedia.org/wiki/War_of_attrition_(game))

  	<p>December, 2023</p> <h2 style="text-align: center;">Emergence in Complex Systems</h2> <p style="text-align: center;">Micro-study</p> <p style="text-align: right;"><a href="http://teaching.dessalles.fr/ECS">teaching.dessalles.fr/ECS</a></p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Name: **Zhiheng Wu**

---

---

## French Flag model of Morphogenesis

### **Abstract**

In the context of morphogenesis, in addition to Turing's reaction-diffusion model, French flag like patterns can directly emerge through the diffusion of a single morphogen and local signal thresholding. This is because cells as living organisms can process the local information and threshold it. But the task of thresholding for cells is not that easy, they need to rely on their internal control network to process the input and give their output. So in this study, I first tried to use Evolife to create a French flag-like pattern. Then I further discussed how can cell achieve signal thresholding.

### **Problem**

Turing's original reaction-diffusion model was a very insightful and powerful explanation for how pattern emerges. It was successfully applied to explain why oscillating chemical reactions like Belousov-Zhabotinsky reaction could react spatial patterns like traveling waves, spirals and so on. Turing's reaction diffusion model also has wide application in morphogenesis, where people used it to explain the formation patterns on animals' skins like stripes of zebras.

But in the context of biology, one should notice that the local environment are some active agents, namely the cells, which make it different from simple oscillating chemical reactions. Because these local agents are not just some passive representer of their local morphogen concentration, they can further process their local information and make action based on it. So one simple feature we could add to the original reaction diffusion model is local concentration thresholding, which means cells can assess whether the morphogen concentration is below or above a threshold and based on that they determine what type of cells they will end up into.

And it was shown that in fruit fly embryos, French flag like three color pattern can emerge simply from diffusion and thresholding. So this study will try to show such kind of process using Evolife program. One should also notice that this kind of thresholding is not a trivial

task for cells. So I further discussed how could cells achieve capability of thresholding based on their internal molecules.

## Method

Diffusion module provided by Evolife. I first tried to modify the Reaction-Diffusion program in Evolife. Because for my purpose, the reaction is not necessary, I simply disabled the reaction part from the model, so there's only morphogen diffusion in the model. To have enough morphogens diffuse through the space, and create a reasonable concentration gradient, I also made the initial points as 'sources' which means that they are not only the place where molecules start to diffuse, but they also produce morphogens. Then, in each bin, the color was displayed based on whether the local concentration is above or below certain thresholds. The French flag has three colors so that we need two thresholds to segment a continuous concentration gradient created by pure diffusion into three separated regions.

For thresholding, the simplest way, and also what I did in Evolife is by manually adding if conditions to assess whether cell content is above the higher threshold or below the lower threshold or in between. To create a French flag like pattern, I initialized a whole line of grids as the source where morphogens start to diffuse.

I also constructed some possible regulatory networks and used differential equations to analyze the consequence of these networks to see how they can achieve thresholding of input signal. But for this part, I have not successfully integrated it into Evolife thus was also not shown in my presentation.

## Results

With simple implementation for morphogen diffusion and local cell thresholding, the program indeed generated 3 tripe French flag-like patterns in 2d space. Notice that the morphogen diffuses to both sides

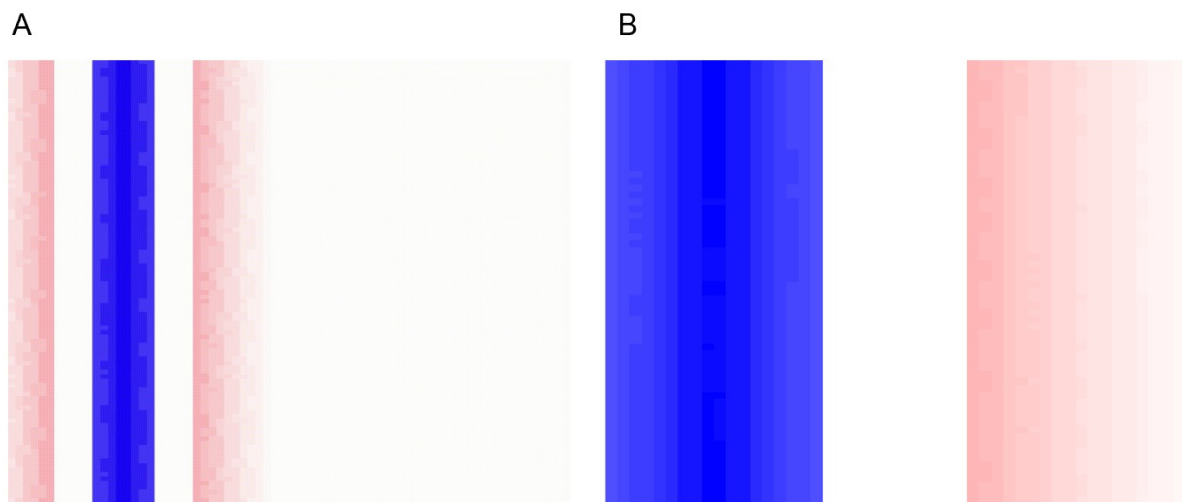


Figure 1. French flag-like pattern.

Next, for signal thresholding, I designed a signaling network as follows. The processing node simply copies the value from input signal. Then it can degrade output1 and activate output2. At the same time, output1 is undergoing self-activation while output2 is under self-degradation. Then we can represent the three different final state using the value of output1 and output2. For example, we can use high value of output1 and low value of output2 to

represent one state, the converse to represent another state, and the third state can be expressed as having both output1 and output2.

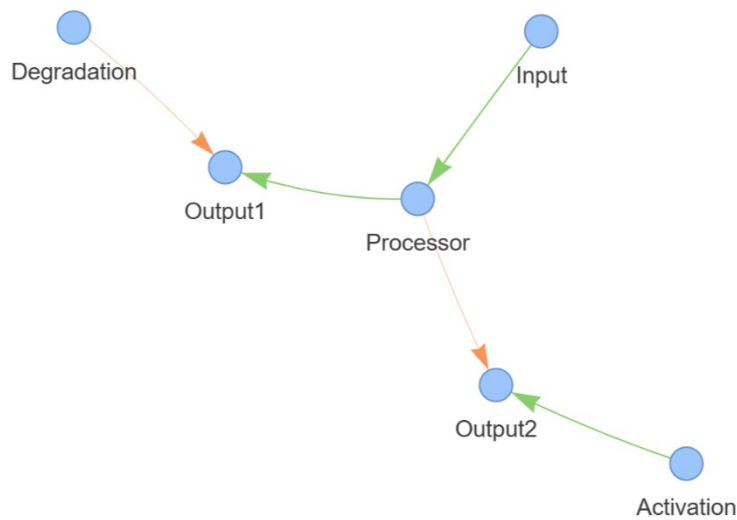


Figure 2. Signaling network for thresholding, orange edges denote inhibition, green edges denote activation.

Because living cells cannot make very sharp binary switch between 0 and 1 like computers, I represented the outcome of this signaling network as a set of differential equations of output1 and output2. I tried to use the mass law of chemical reactions to give a more realistic variation of output1 and output2, equations is like below:

$$\frac{d o_1}{dt} = P - D \cdot o_1$$

$$\frac{d o_2}{dt} = A - P \cdot o_2$$

Where I give both D and A the value of 0.5. Initializing the system with different input value between 0 and 1 indeed gives me different output. When input is very small, the output2 will dominate, when input has a higher value (like 1), output1 will take over. While with the input value in between, we will have a mix of output1 and output2.

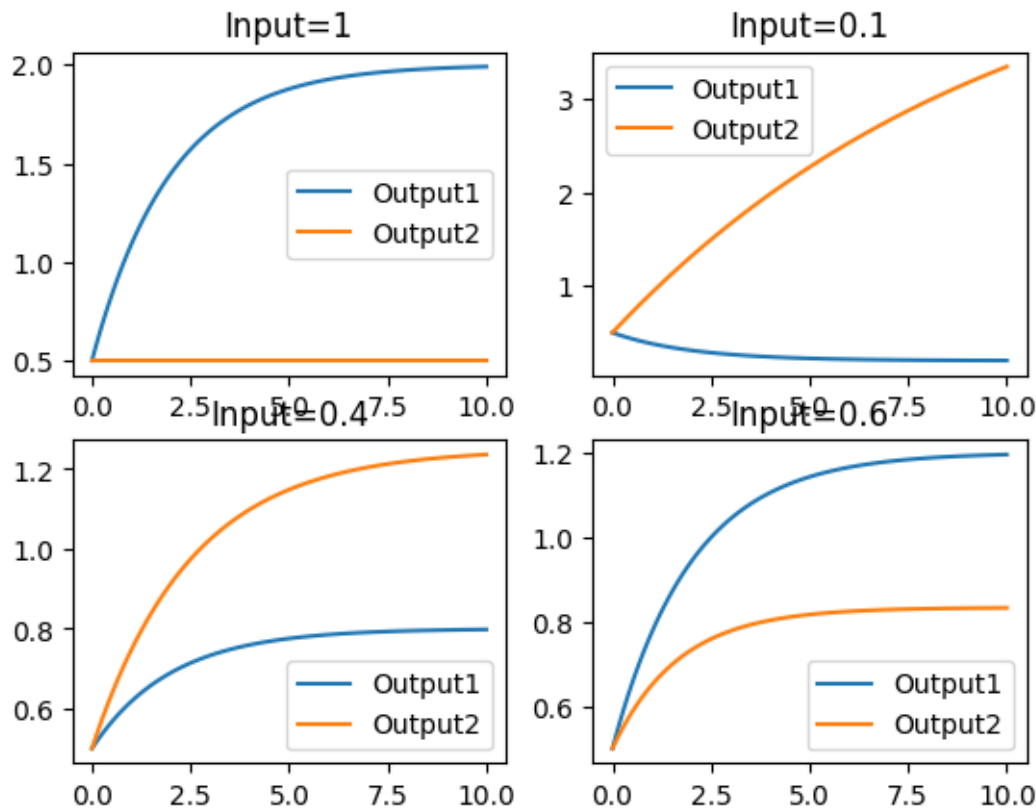


Figure 3, The dynamic change of Output1 and Output2 with time under different input conditions.

## Discussion

The original French flag model was applied to fruit fly embryos, where one morphogen diffuses from one side to another side of the embryo. But in what I show, the morphogen is diffusing to both sides. This is because of the boundary condition in the diffusion model in Evolife was set as cyclic, so that the whole plane behaves like a sphere and there is no end on the simulated plane. I was not able to understand how to modify the boundary condition of the model. So I did not succeed in allowing the morphogens to diffuse only from one side to another and create a real French flag-like pattern.

In the differential equations, I demanded the inhibition of output1 and output2 to depend on their own concentration. This may make sense in biological systems because this usually depends on the protein degradation catalyzed by some enzyme. While as chemical reactions, they should obey the mass law of reaction, which gives these degradations the dependence on the mass of concentration of themselves. Otherwise, continuing degradation even though the amount of target reactant becomes zero would be very unrealistic. Limited by time, I did not do any analysis on the system. It would be interesting to find the steady state of the system and test the local stability of it. Drawing phase portrait could also be interesting to see the global stability of the system.

The limitation in my analysis is that I only looked at deterministic model, if I allow random fluctuation, it is possible to observe some interesting behaviours. In Evolife, the random fluctuation seemed to be achieved by adding one random number to the original model, this could be a choice, but may not be the best solution. One better way of doing this might be using the Gillespie Algorithm, where every interaction between agents was taken as a Poisson process. But also limited by time, I was not able to implement Gillespie Algorithm.

## ***Bibliography***

1. Wolpert, Lewis. "Positional Information and the Spatial Pattern of Cellular Differentiation." *Journal of Theoretical Biology* 25 (1969): 1–47.