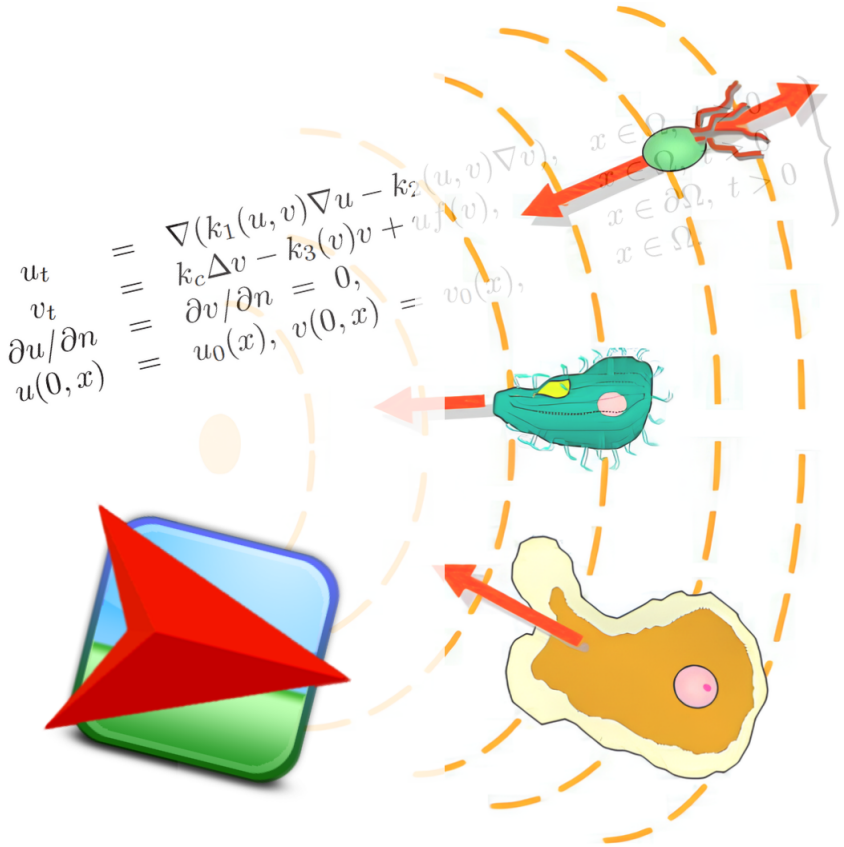
Université Côte d'Azur
Double Licence
Mathematics & Life Sciences

Pluridsciplinary Project: January-May 2024

Agent-Based Models for Eukaryotes & Prokaryotes Chemotaxis

A NetLogo Approach

Morgan Scalabrino
Jules Baldous



Artistic representation of chemotaxis fused with Keller Segel equations and NetLogologo

SUPERVISOR:
Franck Delaunay & Christophe Becavin & Simon Girel

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

UNIVERSITÉ CÔTE D'AZUR

Acknowledgments

We would like to thank our three supervisors: Franck Delaunay, who gave us useful feedback on the biological part of our project and on writing the report, Cristophe Bécavin, who helped us to understand more precisely what we wanted to model, and Simon Girel, who supervised us on the mathematical part of our project. We would also like to thank all our classmates who followed the progress of our project.

Distribution of Work

Morgan Scalabrino :

Bibliography, Eukaryotic model, Immune System model, fitting models to reality, report.

Jules Baldous :

Bibliography, Prokaryotic model, fitting models to reality, video and illustration, report.

Abstract

Chemotaxis is one of the most spread property in organisms. It is found in various field like metastasis or bacteria search for food. In this project we aim to model this phenomena by agent-based model under NetLogo. We adjusted our model with experimental data and various results about chemotaxis found in literature. We created three chemotaxis models: a Prokaryote one, an Eukaryote one and an Immune System model. We then investigate the main results of our models and compare them to the reality.

Key Words :

Chemotaxis, Agent-Based Model, NetLogo, Immune System, Keller-Segel.

Word Count :

4831

Contents

List of Figures

1 Context

1.1 Biological context

The aim of our project is to model chemotaxis in eukaryotes and prokaryotes, which do not follow the same mechanisms. Taxis is the ability of an organism to detect and respond to an external stimulus [1]. It has been widely studied, in particular chemotaxis, which is the ability to detect and migrate across a chemical gradient, which can be food, a signal, etc. [2]

Chemotaxis takes many forms and occurs in many situations, from the immune system detecting pathogens [2] to the ability of tumour cells to form metastases [1].

Chemical gradients can be either attractive (e.g. food) and therefore called chemoattractants, or repulsive (e.g. poison) and therefore called chemorepellents.

This innate behaviour is not only used by complex eukaryotic organisms, but also by prokaryotes, and is particularly well described in bacteria. As a result, the wide range of applications of chemotaxis is a major research interest today.

1.1.1 Prokaryote chemotaxis

We will present the best understood mechanism of prokaryotic chemotaxis, namely bacterial chemotaxis of *E. Coli*.

E. coli (and other bacteria as well) move through space by a run-and-tumble mechanism, alternating a running phase with their flagella, and a tumbling phase to reorient themselves in another direction [2]. See figure 1 for a visual example.

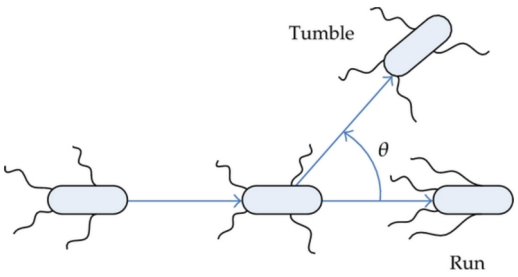


Figure 1: Chemotactic behavior of *E. coli*: run-and-tumble.
Retrieved from: https://doi.org/10.1155/2012/409478

When bacteria are in the presence of a chemoattractant (resp. chemorepellent), a complex mechanism causes the flagella to produce a longer (resp. shorter) running phase and a shorter (resp. longer) tumbling phase. This mechanism alone produces an inclined movement towards an attractive (resp. repulsive) gradient.

For *E. coli*, the chemoattractant may be aspartate or glycine and the chemorepellent may be arginine.

The biased run-and-tumble mechanism is the one we have chosen to model in our project for the prokaryotic model.

1.1.2 Eukaryote chemotaxis

Chemotaxis in eukaryotes follows more complex mechanisms. Here we focus on *Dictyostelium discoideum* (**Dicty**) and the immune system.

Dicty chemotaxis :

Dicty is a species of amoeba found on mats of dead leaves in forests and is an organism that has been extensively studied in the laboratory for its chemotactic ability.

During their life cycle, amoebae aggregate (by chemotaxis) with each other through the diffusion of a chemoattractant (cAMP) when food is depleted. They thus form a multicellular organism called a "slug" and can become a fruiting body [1].

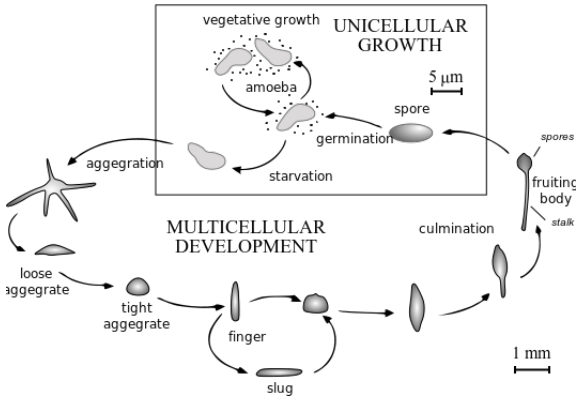


Figure 2: Life cycle of Dicty: from an unicellular organism to a fruiting body.
Retrieved from: https://en.wikipedia.org/wiki/Dictyostelium_discoideum

Without cAMP, amoebae move randomly by extending their pseudopodia. When a cAMP gradient is detected, the amoebae amplify it and this creates a directional polarity, creating a new pseudopod front towards the gradient [1]. See figure 3 for an example of aggregation.

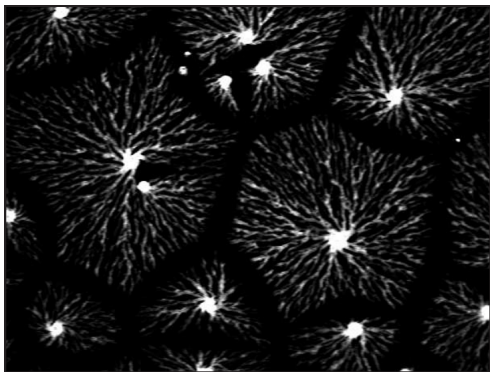


Figure 3: Chemotactic behavior of Dicty: Aggregation in a slug.
Retrieved from: https://d-nb.info/1007859075/34

To model the basic chemotaxis of eukaryotes, we chose this mechanism because it is relatively easy to understand.

Immune System chemotaxis :

The immune system is a highly dynamic network involving various cells such as lymphocytes and leukocytes (neutrophils and macrophages). In order to protect the body against pathogens, they need to reach specific places in the body. So it's not surprising that we find chemotactic behaviour among immune cells, particularly with neutrophils or lymphocytes [3].

Immune cells are guided through the body by chemokines and, once at the site of infection, they can be guided towards the pathogens by pathogen-specific ligands [3].

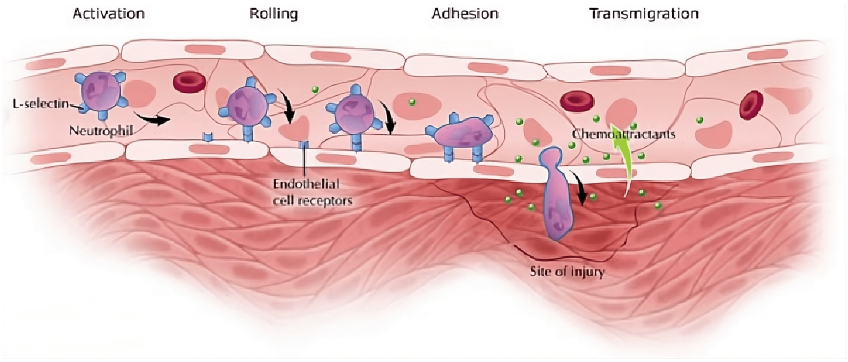


Figure 4: Chemoattractant-mediated diapedesis of a neutrophil in an infectious site.
Retrieved from: https://api.semanticscholar.org/CorpusID:8998046

Here we explore lymphocyte chemotaxis, which follows a complex mechanism involving multiple signalling pathways activated by chemokine receptors. In the absence of chemoattractant, pseudopod formation is random. In the presence of chemoattractant, actin polymerisation is biased to produce more pseudopodia where the cell perceives the highest concentration of chemoattractant. [3].

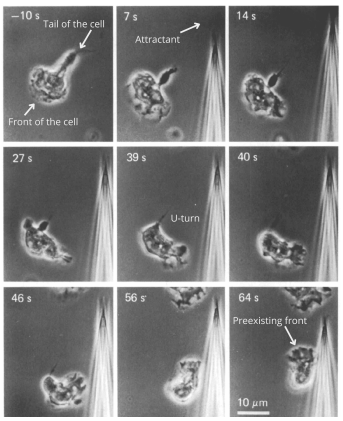


Figure 5: Chemotactic behavior of immune cells: U-turns.
Retrieved from: https://doi.org/10.1242/jcs.52.1.1

Unlike Dicty, there is no directional polarity, which means that instead of generating a new pseudopod front, the cells will produce a U-turn, i.e. they

will retain the pre-existing front and reorient themselves in space [3]. Figure 5 provides a visual illustration of this phenomenon.

1.1.3 Test for chemotaxis

To determine whether cells behave chemotaxically, chemotaxis tests need to be created. We will focus on three tests. The T-maze, the Boyden chamber test and the real-time imaging test (using a pipette tip for the chemoattractant).

T-maze :

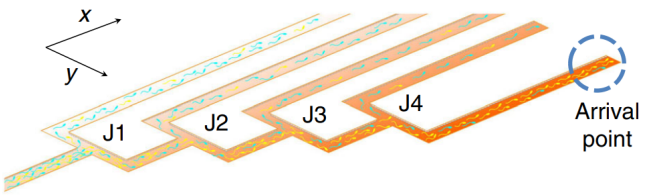


Figure 6: Schematic of cells swimming through a T-maze.
Retrieved from: https://doi.org/10.1038/s41467-019-09521-2

The T-maze is a system that exposes cells to a binary decision in the presence of a chemoattractant. If the cells are sensitive, they should direct their decision towards the source of the chemoattractant [4] through the one lane with higher concentration. If they are not they should go in equal number in both lane of each crossing.

Boyden chamber :

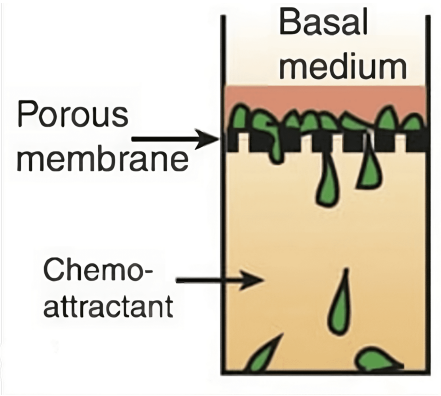


Figure 7: Schematic of cells that navigate in a Boyden chamber.
Retrieved from: https://doi.org/10.1016/j.smim.2004.09.005

For this method, cells are placed on a porous membrane and pass through it more if they are sensitive to the chemoattractant placed across the membrane. By comparing with cells without chemoattractant, we are able to study chemotaxis behavior [4].

Real-time imaging with chemoattractant :

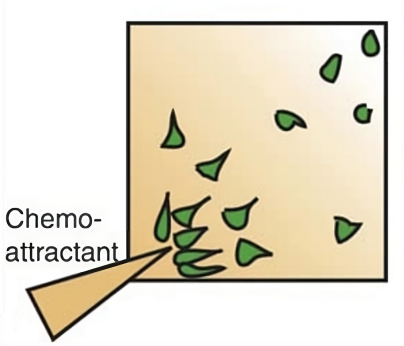


Figure 8: Schematic of a real-time imaging with attractant (pipette) assay.
Retrieved from: https://doi.org/10.1016/j.smim.2004.09.005

This is the most natural method and produces some very fine films on chemotaxis. For example see *this one* by Dicty.

1.2 Modelling context

Chemotaxis has long been studied and modelled. One of the first models to do so was the Keller-Segel model established in the 1970s by two researchers: Evelin Fox Keller and Lee A. Segel [5].

This model is still often used for its simplicity and relatively intuitive aspect. It is a partial differential equation (PDE) model that was originally designed to describe the aggregation of Dicty. Nowadays, this type of formalisation can be found [5]:

$$\begin{cases} \frac{\partial n}{\partial t}(x,t) = \nabla\left(D_n.\nabla n\right) - \nabla\left(\chi.n.\nabla c\right) \\ \frac{\partial c}{\partial t}(x,t) = \nabla\left(D_c.\nabla c\right) + p(n) - d(n) \end{cases}$$

With n(x,t) which is the concentration of cells at time t and position x (in 2D space) and c(x,t) which is the concentration of chemoattractant at time t and position x. These two objects depend on:

For the cells:
- χ which is the sensitivity to the chemoattractant,
- D_n which is the diffusion coefficient, i.e. movement in the absence of chemoattractant.

For the chemoattractant:
- D_c which is the diffusion coefficient of the chemoattractant,
- p which is the production of the chemoattractant (by amoebae if we are modelling them),
- d which is the degradation of chemoattractant by the cells.

The models that followed later were simply variations and adaptations of this one [5].

Like many PDE models, we are unable to obtain an exact solution for n and c. This means that modelling precise behaviour is difficult to formalise mathematically. It is for these reasons that agent-based modelling and approximation methods were created and that we have carried out our analysis on agent-based models (ABM) under NetLogo.

1.2.1 Agent-based models under NetLogo

Agent-based models are a type of modelling that uses agents (which reproduce cells, particles, etc.) and makes them evolve according to rules that mimic reality (biology, physics, etc.) [6] . The advantage of these models over pure equation models is that we can easily add conditions, probability or interactions.

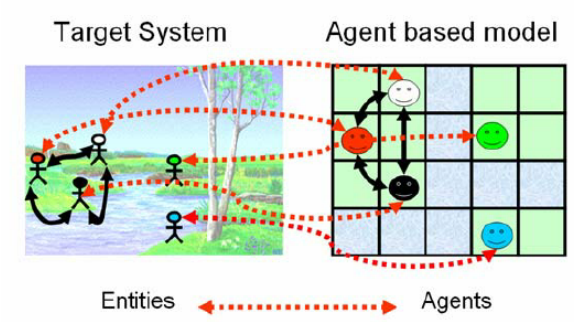


Figure 9: Simplified scheme for the principle of an ABM.
Retrieved from: https://doi.org/10.1007/978-3-540-93813-2_5

We can then try to obtain information and a better understanding of how a phenomenon works. If we find that the model mimics a phenomenon (by comparing it with data from a real experiment), we can use its simulation to overcome certain difficulties associated with experimentation, for example.

NetLogo is one of the easiest tools to use for building an ABM, which is why we've chosen it here.

NetLogo is a programming language entirely focused on ABM. It was developed by Uri Wilenski in 1999 and is still often used today as a first approach for many models [6].

In NetLogo, space is divided into a grid of patches, agents are called turtles and time passes in ticks. The user (called the observer) can give rules to the patches and agents so that they do what he wants. See figure 10 for a simple example.

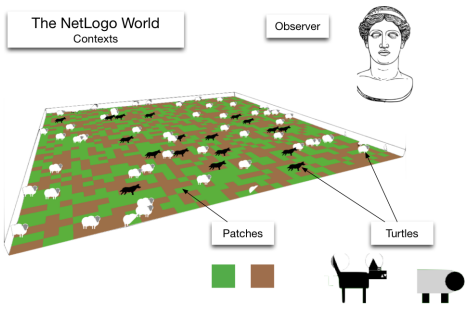


Figure 10: A basic NetLogo world with turtles, patches and observer.
Retrieved from: https://www.geog.leeds.ac.uk/courses/level3/geog3150/practicals/practical1/3_ask.php

1.2.2 Test for chemotaxis under NetLogo

To test whether or not our agents, implemented in NetLogo, perform chemotaxis, we are going to run tests inspired by the 3 chemotaxis tests presented earlier.

First of all, the T-maze will be a maze like the one in figure 11, where the brown spots are the background, the grey spots are the walls of the maze and the red spots are a source of attractant. The agents will be placed on the left side of the maze and, if they are capable of chemotaxis, they will move towards the source of attractants, at the top right of the maze.

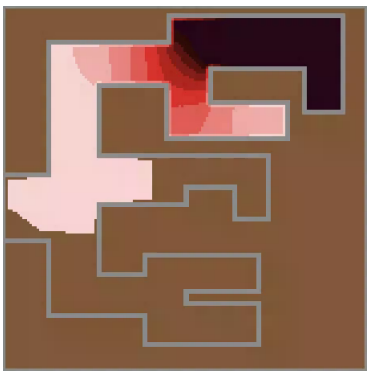


Figure 11: T-maze in NetLogo.

The second test will be the Boyden chamber assay, which is easy to reproduce in NetLogo, see figure 12. We'll place some agents on top of the chamber, simulate for a set period of time and then compare with the same experiment without chemoattractant.

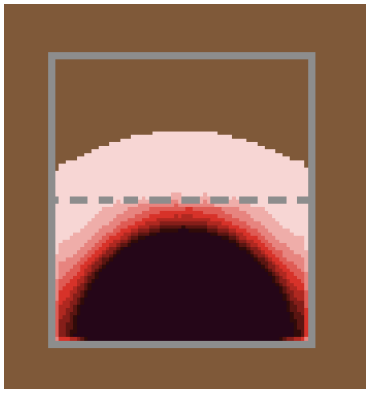


Figure 12: Boyden chamber in NetLogo.

The third test will be inspired by the real-time experiment and will consist of depositing a source of chemoattractant in a space inhabited by an agent and seeing whether it is attracted to it. We can then deposit another source of attractant and see if the agents are more attracted to the new one than to the old one. This technique will answer an interesting question detailed in 3.

Finally, the last technique for measuring chemotaxis is to prepare a channel

with a source of chemoattractant at one end and measure the time taken for the agents to reach the source of attractant by placing them at the opposite of the source. See figure 13 for a visual example.

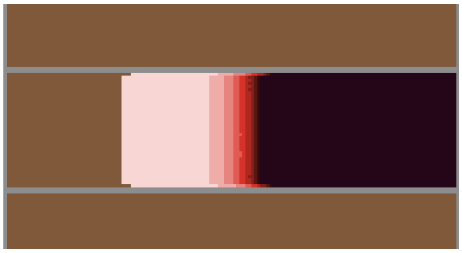


Figure 13: Corridor experiment

We'll use this test to compare agent speed as a function of our model parameters.

2 Agent-based models of chemotaxis

To model chemotaxis, we created 3 different models: a prokaryotic model, a eukaryotic model and an immune system model (derived from the eukaryotic model).

We will first present our models and then examine the results they produce, results that can sometimes prove the accuracy of our models when confronted with reality.

We have produced a table (accessible in the 6.1 section) which summarises all the different agents and parameters that we have in our models.

2.1 Models

The aim of our project was to build 3 different models of chemotaxis, not only by changing the shape of our agents but also by changing the mechanism of each of the models and their specificity.

To do so, we built a model with a chemical species and a diffusion process, then we implemented agents that represent either bacteria, amoebae or immune cells. These agents move towards the highest concentration of chemoattractant and can do more depending on the model. Finally, we added improvements such as walls, sliders for parameters, etc. to test our model and adapt it to real data.

2.1.1 Diffusion process

First of all, we created a diffusion code. For each patch, there is a variable that describes the concentration of the chemical species found there. At each tick and for each patch, part of the chemical species found there will be redistributed to the 8 surrounding patches.

To make this process as general as possible, we have chosen to offer the possibility of regulating this diffusion process and its speed by changing the percentage of the chemical species on each patch that will move towards the

other patches (see section 6.2 for the detailed code). This parameter correspond to the D_c in the Keller Segel model.

(a) Diffusion under NetLogo (b) Diffusion by Allen-Cahn equations

Figure 14: Comparison between our model and under Allen-Cahn's model

To test the accuracy of our diffusion process, we looked for other examples of diffusion processes, in particular using mathematical models. For example, we found (in figure 14) that by simulating a reaction diffusion process using either the heat, Fisher or Allen-Cahn equations [7], our diffusion process is similar to those created by the mathematical equations, attesting to its robustness. We wanted to create our own diffusion process, despite the existing one ("diffuse" primitive in NetLogo), to be able to modify it and be more accurate.

2.1.2 Walls

Next, we needed to create the walls already shown in the image in section 1.2.2. To do this, all we need to do is color certain patches in grey and prevent the attractant from diffusing onto them (by means of an if condition in our diffusion process, see section 6.2 for detailed code).

2.1.3 Bacteria agents for Prokaryotic model

The aim of this model is to reproduce the chemotactic behaviour of prokaryotes by focusing on a run-and-tumble mechanism and therefore by focusing on bacteria such as *E. coli*.

First we need to produce a run-and-tumble mechanism to allow the agents to move and then bias their movement in the presence of a chemoattractant. The run-and-tumble mechanism is very basic: with each tick, there's an 80% chance of running and a 20% chance of tumbling. We biased it by reducing the chance of tumbling to 1% in the presence of sufficient differences in chemoattractant.

To pursue the goal of replicating reality, we calibrate the probability of running and tumbling by comparing the average time spent running and the average time spent tumbling in *E. coli*, which is approximately 1 second for the running phase and 0.2 seconds for the tumbling phase [4].

In addition, we know that bacteria discern up to 0.0001% differences in chemoattractant [8], so we adapt the sensitivity of our agents to this value.

Figure 15 shows our basic model on NetLogo with the bacteria in blue, the chemoattractant and the walls in grey.

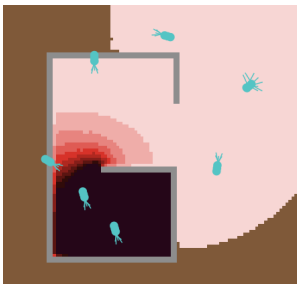


Figure 15: Example of our Prokaryotic model

2.1.4 Amoebas agents for Eukaryotic model

The aim of this model is to reproduce the chemotactic behaviour of eukaryotes by focusing on a mechanism such as that of amoebae described in the section 1.1.2.

To do this, we coded an agent modelling an amoeba that moves towards the patch with the highest concentration in its direct environment (8 cells around it). We created a loop to go through each of these patches and record the position of the one with the highest concentration. We then ask the agent to move towards this patch.

Several exceptions have been dealt with, notably in the case where there are several patches with the same concentration, we store each of their positions in a list and choose randomly from them.

By reading articles on the mechanism of the amoeba, we adapted the sensitivity of the amoeba to 1% differences in chemoattractant [9].

Figure 16 shows our basic model on NetLogo with the amoeba in blue, the chemoattractant and the walls in grey.

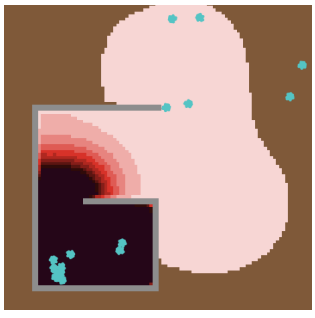


Figure 16: Example of our Eukaryotic model

2.1.5 Pathogens, Macrophages and Lymphocytes agents for Immune System model

To model a basic immune system, we need three types of agent :

- Pathogens, which multiply at the site of infection and damage tissues, producing a signal that can be considered as a chemoattractant for immune cells.

- Macrophages, which are the first response immune cells and are found in tissues awaiting infection or injury.
- Lymphocytes waiting in the lymph node to be alerted by antigen-presenting cells from the site of infection. These lymphocytes of two types will then either go to the site to kill the pathogens (T lymphocytes), or remain in the lymph node and produce antibodies against the pathogens (B lymphocytes).

To model this situation, we took three different types of agent: pathogens, macrophages and lymphocytes. We simplified the situation by creating pathogens that spread randomly and produce a chemoattractant for macrophages. This chemoattractant represents the DAMPs (Damage-associated proteins) and pathogen antigens and proteins.

Our first type of immune cells (called macrophages for simplicity's sake), which can be considered as a mixture of neutrophils and macrophages, are attracted to pathogens and kill them (macrophage). This destruction represents phagocytosis of the pathogens. Our immune cells then produce a chemoattractant for the lymphocytes, representing the neutrophils, which break down the pathogenic proteins into antigenic peptides before reaching the lymph node to present them to the lymphocytes (for activation).

In the lymph node, although there are different types of lymphocytes (T CD8, T CD4 or B) which perform different actions, we have decided to create just one. Our lymphocytes can be seen as a mixture of B lymphocytes and T lymphocytes, because they produce antibody that kill pathogens (L. B) or kill pathogens on contact (L. T).

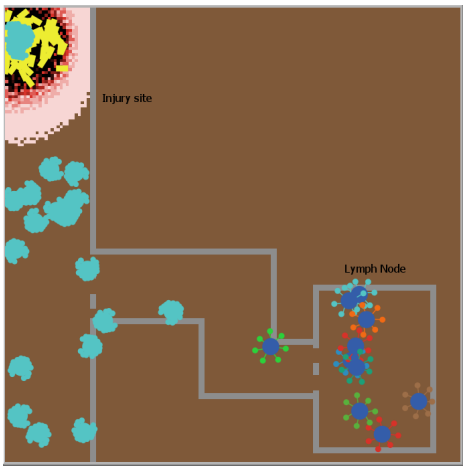


Figure 17: Example of our Immune System model

We have tried to follow some of the ideas in the papers by Tokarski et al. [10] and Chiacchio et al. [6].

3 Results & Fidelity

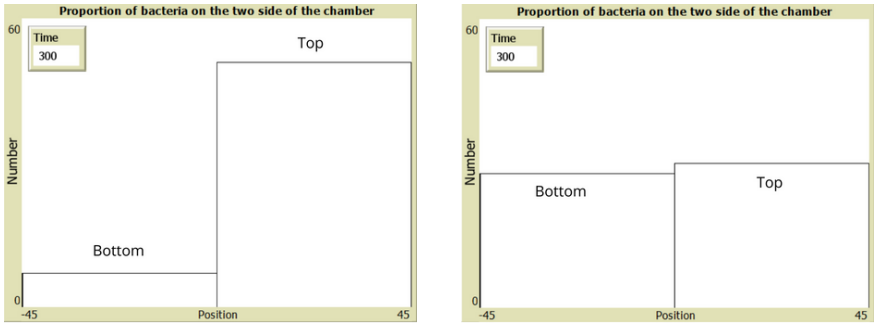
In this section, we present the main results and properties that we have observed from our models, results that can sometimes prove the accuracy of our model in the face of reality.

3.1 Results in Prokaryotes model

First of all, we checked that our agents were indeed performing chemotaxis with the tests presented in section 1.2.2. Next, we looked for other arguments to prove the accuracy of our model. Finally, we introduced how to make a chemorepellent in our prokaryotic model.

Boyden chamber assay

In the Boyden chamber test, we find that the proportion of agents in the bottom part (where the source of the chemoattractant is) is higher when we place the chemoattractant there. This means that our agents are more attracted to the bottom of the room when it contains a chemoattractant, which implies that the movement of our agents is biased in the presence of chemoattractant.



(a) Without chemoattractant. (b) With chemoattractant.

Figure 18: Proportion of the bacteria in the bottom or the top part of the Boyden chamber. Chemoattractant is in the bottom part.

Solving maze

We also found that our agents were able to solve the T-maze perfectly. This is further evidence that our agents can bias their walks towards chemoattractants. We found similar results to Tweedy's 2020 paper [9], which is an argument to support the accuracy of our model against reality.

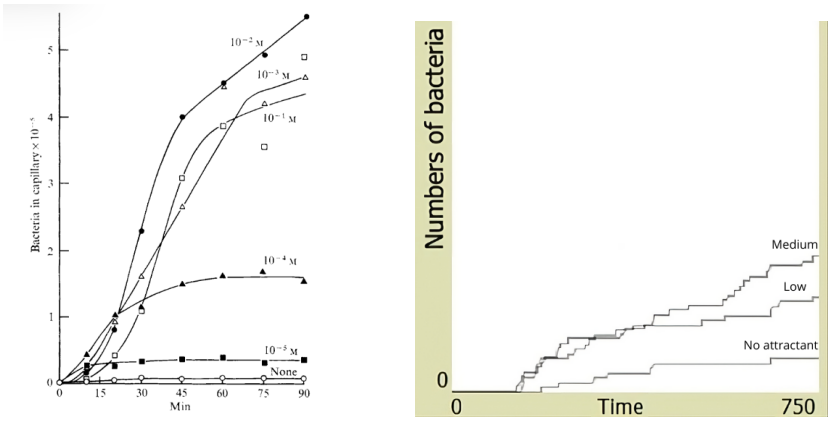
(a) Screenshot at the beginning. (b) Screenshot after 7 min. of simulation.

Figure 19: Prokaryotes agents solving a T-maze.

Another chemotaxis experiment was carried out by Adler in 1973 [11]. He

measured the rate of accumulation of bacteria (*E. coli*) in a capillary with different concentrations of attractant (L-aspartate). We reproduced the experimental set-up in NetLogo and measured the rate of accumulation of our agents in the capillary.

Our model in NetLogo was able to recreate the same aspect of the graphs presented in figure 20. We see that for a low concentration of chemoattractant, a small proportion of bacteria can reach the capillary with a distinct curve shape that we were able to reproduce in NetLogo.



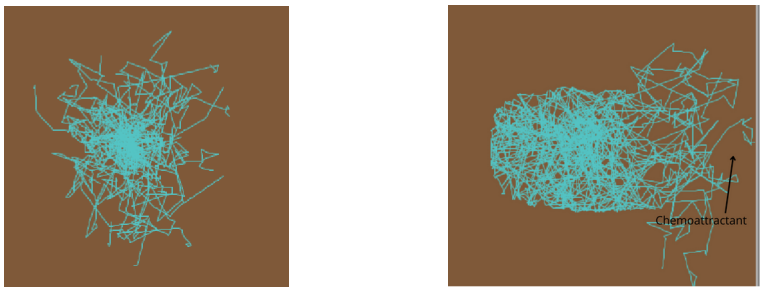
(a) Real data (b) Under NetLogo

Figure 20: Rate of accumulation of bacteria/agents in capillaries containing various concentrations of chemoattractant.

The results follow the same pattern in NetLogo and in real life, proving once again the accuracy of our model.

Measurement of the biased movement

Another way of proving the fidelity of our model to reality is to simply measure the biased movement by tracing the path that our agent follows in the presence or absence of a chemoattractant. This method is common in chemotaxis tests to obtain a qualitative result, as in Tweedy's 2023 paper [12] or Singh's 2022 paper [13].



(a) Random walk (b) Biased walk

Figure 21: Path of agents in presence or not of a source of chemoattractant

Chemorepellent in models

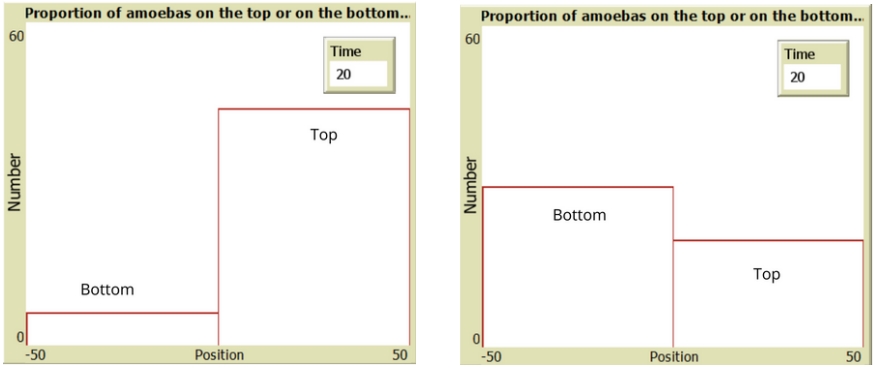
So far, we have only discussed chemoattractants in our model. In this paragraph, we examine the possibility of creating a chemorepellent. In fact, this is an easy task; by simply inverting our chemotaxis mechanism, we were able to create a repellent species. For example, in the prokaryote model, we introduced a slower running phase and a longer tumbling phase in the presence of a chemical compared to the absence of a chemical.

3.2 Results in Eukaryotes model

In this section, we have continued with the same chemotactic test as in the previous section to validate the chemotactic behaviour of our model. We will see that these tests tend to prove the accuracy of our model and we will finally look at some results of interest from the eukaryotic model.

Boyden chamber Assay

As in 3.1, we find that the proportion of agents is higher at the bottom of the chamber in the presence of a chemoattractant than in the case without chemoattractant. This is consistent with the general result expected for Boyden chamber tests, as for example in Figure 1 of Schagat's 2002 paper [14].

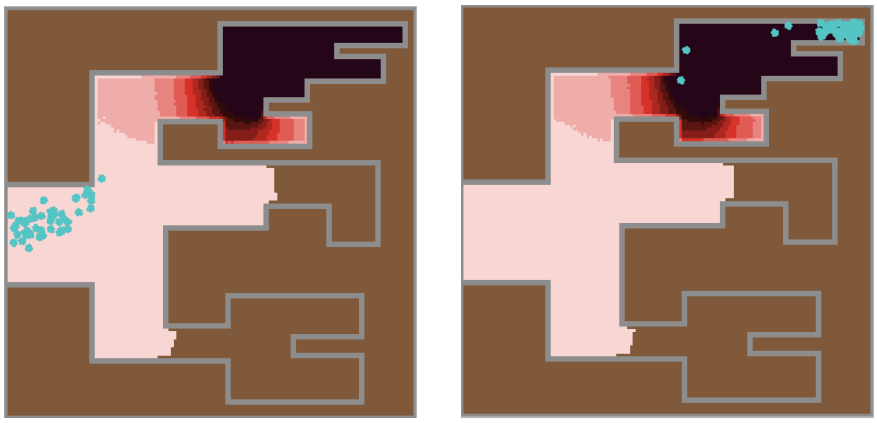


(a) Without chemoattractant. (b) With chemoattractant.

Figure 22: Proportion of the amoebas in the bottom or the top part of the Boyden chamber. Chemoattractant is in the bottom part.

Solving Maze

Our agent showed great efficiency in solving the T-maze. This is evidence of chemotactic behaviour. These results are also a proof of the fidelity of our model because our results are very close to those found in Tweedy's 2020 paper [9].



(a) Screenshot at the beginning. (b) Screenshot after 10 sec. of simulation.

Figure 23: Eukaryotes agents solving a T-maze.

Degradation of the chemoattractant

Cells sometimes have to find their way around highly complex environments. Since simple chemotaxis cannot explain this phenomenon, notably because of a problem of sensitivity, an underlying mechanism is hidden. Cells can perceive a difference of up to 1% between their front and back faces [9], which is ineffective in environments with small differences in concentration (uniform concentration or environments very poor in chemoattractant).

One explanation proposed and confirmed by Tweedy in 2020 [9] is that the destruction/consumption of the chemoattractant by the cell creates a sharp self-generated local gradient around the cell. The creation of this sharp local gradient allows the cells to perceive very small differences in environmental gradients and efficiently move up to the highest concentration.

By giving the possibility to our agents (prokaryotes and eukaryotes) to consume chemoattractant we observed results of interest.

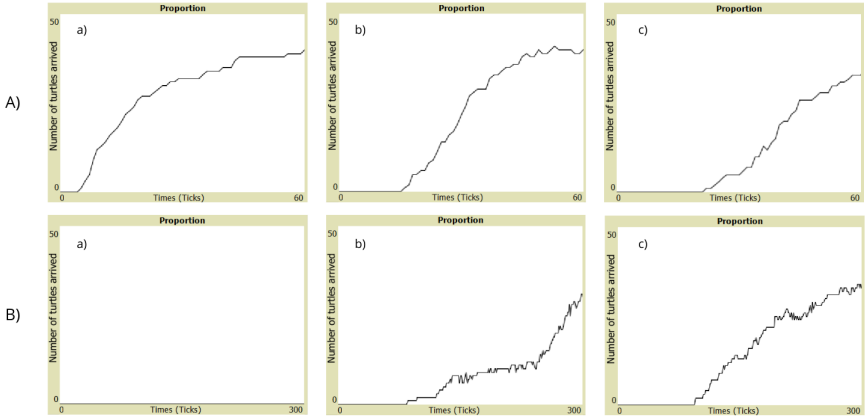


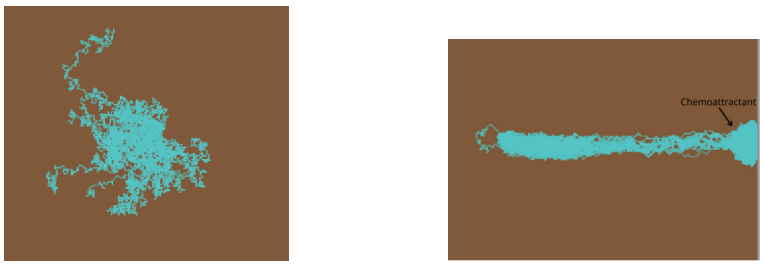
Figure 24: Number of amoebas arrived at the end of a path by time. *Condition A): On a unsaturated environment ; condition B): On a saturated environment ; a) = consumption of 0%, b) = consumption of 50% and c) = consumption of 95%*

We tested our amoebas agents and our bacteria agents and they gave the same results, on the next part we described the results only for amoebas. We used the corridor experiment described in section 1.2.2.

First, we observed a phenomenon similar to those in Tweedy's article [9]. For a classic gradient with enough differences (condition A) we don't see much differences between the agents who consume and the agents who doesn't. On the opposite when the space is saturated in chemoattractant the agents who consume are more effective than the non-consuming agents. We see in figure 24 that for an equal period of time and for a saturated environment (full of chemoattractant) the non-consuming agents don't arrive at the end of the path in contrary of the consuming one's who reach the end of the path (near the source of attractant).

Measurement of the biased movement

Another proof of the accuracy of our model is the measurement of the biased motion as realized in Tweedy's 2023 paper [12] or Singh's 2022 paper [13].



(a) Random walk (b) Biased walk

Figure 25: Path of agents in presence or not of a source of chemoattractant

Local/Global Maximum

By simulating the real-time imaging test in the eukaryotic model, we observed that amoebae tend to orientate themselves towards the nearest source of attractant, even if there is another very abundant one. We call this phenomenon the local/global maximum problem. See figure 26 for a visual example.

Figure 26: Illustration of the local/global problem.

On a real in-vitro assay this problem doesn't appears, see for example *this movie*.

By allowing our agents to consume the chemoattractant, we have solved this problem, as the agents quickly consume the maximum local concentration and reach the maximum global concentration.

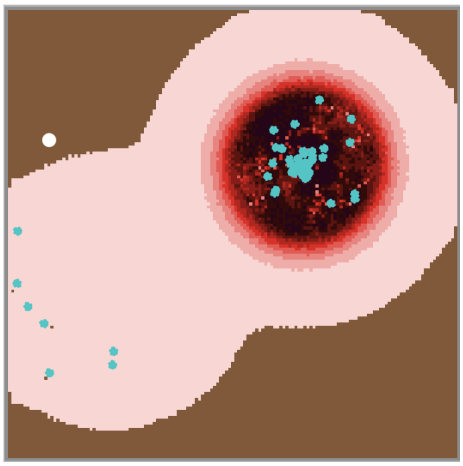


Figure 27: The local/global problem solved by degradation of chemoattractant.

Other evidence of the accuracy of the model

Further evidence for the accuracy of our model is the measurement of biased motion as performed in Tweedy's 2023 preprint [12] with the Chemotaxis Efficiency Index (CEI).

The CEI is the distance travelled on the gradient divided by the total distance travelled. We measured the CEI of our model and obtained values of 0.35 ± 0.02, which is realistic compared to the values presented in Tweedy's preprint 2023 [12]. One way to check that our model is realistic for specific cell types would be to modify it to have the corresponding CEI values for each cell type (0.3 is low for neutrophils or amoebae but a good value for cancer cells).

3.3 Results in Immune System model

Although our model is a great simplification of the real situation (cell types missing, mechanisms simplified or ignored), it is nevertheless capable of representing simple, real biological events such as the calling of lymphocytes from a distance by chemotaxis, healing and infection and the return to a normal situation without DAMPs (no inflammation).

Chemotaxis in Immune System model

As this model is derived from eukaryotes and follows the same rules under NetLogo for chemotaxis, we know that our agents follow chemotactic behaviour.

Local/Global Maximum Problem in Immune System model

The problem of the local/global maximum seen above is very interesting here. Without the consumption of the attractant by the macrophages, we have

macrophages that remain trapped for a very long time in a local maximum without effectively driving out the pathogens and above all without progressing towards healing the area of infection/injury. By consuming the attractant, the macrophages are able to hunt more effectively and completely absorb the remnants of damaged tissue, simulating the healing process.

4 Discussion and Insights

In this section, we provide some interesting insights into our model and possible improvements.

4.1 Chemokinesis in Eukaryotes model ?

A problem that arises quickly in the chemotactic behaviour of our eukaryotes is that the cells are too efficient, they can move up the gradient with very small differences in concentration. We controlled this by reducing the sensitivity to a 1% difference. But before that, we controlled this by giving the agents a concentration-dependent probability of random movement instead of always moving towards the patches with the highest concentration in the presence of a very low concentration of chemoattractant.

We later realised that there is a mechanism called chemokinesis, which states that a change in the environment (temperature, pressure or concentration of chemical species) causes the cell to move randomly more than usual.

It would be very interesting to study whether, by increasing the random movement of our agents in the eukaryotic model, we can simulate this chemokinesis phenomenon. After confirmation, we could try to put chemokinesis and chemotaxis in competition as in [15]. Due to lack of time, we did not go to the end of this analysis.

4.2 Explanatory & explained variables of our models

To test the accuracy of our model, we need to isolate the variables that we can vary in our model, the explanatory variables, and the variables that we can measure in NetLogo, the explained variables. We propose here a list of explanatory and explained variables in order to carry out other tests like the one in section 3.

The explanatory variables can be either variables that we have added to modify the environment in NetLogo, or the coefficients present in the Keller-Segel model:

- Diffusion of the chemoattractant, by modifying the diffusion coefficient in our model.
- Agent sensitivity, by modifying the sensitivity threshold.
- Random agent motility (distance/frequency of movement).
- Run-and-Tumble efficiency, by changing the probability to tumble in our prokaryote model.
- Rate of consumption or production of chemoattractant by our agents, by changing the corresponding constants in our model.

- Space, by changing the place where we measure an explained variable.
- Time, because we can simply vary the time taken to make measurements in our model.
- Concentration of chemoattractant, our chemoattractant being between 0 and 1.

The explained variables, which can be linked to the real data, can be:

- Measurement of the chemotactic index using a Boyden chamber device.
- Measuring the drift of our bacterial agents in space.
- Number of agents in a capillary, as in 3.1.
- Proportion of cells that solved the T-maze, as in Tweedy's 2020 paper [9].
- Speed of our agents.
- Directionality, for both model.
- Efficiency of the macrophages, for the immune system model.
- Time taken for lymphocytes to arrive at the site of injury and cure it.

4.3 Possible improvements of our models

For the basis of our model, a first way of improvement would be to do as in Tokarski's 2012 paper [10] and implement a number of times that the chemoattractant diffuses per units of time. Indeed, they found that this improved the chemotactic behaviour of the agents.

4.3.1 Prokaryotes model

There's a lot to do for improvement in this model, but the key thing would be to bring the probability of tumbling and running into line with reality. We couldn't find any such data, so we made sure that the time spent running and the time spent tumbling were roughly equivalent to the real data [4].

We could also have improved the way our agents perceive gradient differences and the way they react to them.

4.3.2 Eukaryotes model

To improve the eukaryotic model, we could try to implement extended sensitivity. We could implement this by scanning not only the direct surrounding plots, but also the 2nd or 3rd surrounding plots and analysing their chemoattractant concentration. Another solution would be to do the same thing, but for degradation, by degrading part of the surrounding plots instead of just the plot where our agent is located. We could then compare the performance of the cells with one, both or none of these improvements.

An interesting situation to model would be the classic aggregation of Dicty during the starvation period. The cells would produce their own

chemoattractant and aggregate, and then we could model the fact that after sporulation, the new generation of Dicty is more effective and more resistant.

Another interesting supplement would be to implement short term persistence. persistence is the fact that the cell doesn't change of direction for a given persistence time, this is a way to biased the movement of the cells other than chemotaxis. In [12] Tweedy and all. suggest that it can be an upgrade of the chemotaxis to filter noises in the journey up the gradient.

In [16], we read that amoebas can sense attractant in waves and not only in a continuous flow. The amoebas around are still able to find the source of the wave of attractant even if the wave with a high concentration has passed and kept going in the other direction. This simple example shows us the complexity of the real mechanism of chemotaxis and it would be very interesting in the long term to try to model it.

4.3.3 Immune System model

To improve our model, we could add agents that are more specific for B lymphocytes (instead of mixing them with T lymphocytes). We could also create memory lymphocytes with better capacities than those from the first infection.

We could also add another breed of neutrophils (instead of mixing them with macrophages) which could go into the lymph nodes to activate the lymphocytes.

It would also be possible to test different types of pathogen with a specific behaviour or mechanism for reproducing viruses, fungi or bacteria and the different ways in which they act.

The applications and developments are infinite but the difficulty is to remain within a framework that can be reproduced and studied in vitro or in vivo. We have found numerous publications testing the chemotaxis of specific neutrophil/immune cells and it could be interesting (although difficult and time consuming) to extract specific parameters for our model from these data.

Another development could be to model the communication between immune cells as in Tokarski's 2012 paper [10]. Based on this communication, we could then model cancer cells and their interactions with immune cells.

5 Conclusion

To conclude we started from Keller-Segel PDE model for chemotaxis and inspired by it we built an ABM model with chemoattractant, diffusion and our agent that represent either prokaryotes or eukaryotes and can have a chemotactic behavior and go toward the chemoattractant.

We then tested the accuracy of our model against the experimental data. The demonstration of the accuracy of our model confirmed the results we were able to obtain for our models.

Nevertheless, there are still many improvements to be made to our model and we can now, by improving our models, try to obtain other interesting results.

6 Appendix: Tables of Variables and Codes

6.1 Agents & Patches Table

In this section we created a table that contains all the agents and patches of our model to make them easier to understand.

Agent/Patch	Appearance
Bacteria	
Amoebas	
Pathogens	
Macrophages	
Lymphocytes	
Background	
Chemoattractant	
Walls	

Table 1: Table of the different agents and patches of our models

6.2 NetLogo Codes

We used profusely the NetLogo code and documentation already existent to create our models. You can find the official documentation *here*.
We've also created a repository on Github to give people the chance to download the code for our models and this report. Our repository is available *here*.

6.2.1 Prokaryotic model

```
1 ;Extension: "vid" to make movies
2 extensions [ vid ]
3
4 ;;Global variables: "colour" to represent chemoattractant, "tumble" to
      make movement of bacteria, "fuite" to make chemoattractant diffuse,
      "i_low" lower bound for concentration of chemoattractant, "i_up"
      upper bound for concentration of chemoattractant, "index" for
      colours
5 globals [colour i_low i_up index tumble fuite compteur_run
      compteur_tumble]
```

```
6
7   ;;Breed: "bacteria" represent all the turtles, "bacterium" is one of
        the turtles
8   breed [ bacteria bacterium ]
9
10  ;;Patches variables: "c" is the concentration of chemoattractant on
        each patch, "source" is a boolean to put a constant source of
        chemoattractant
11  patches-own [c source]
12
13  ;;Setup Procedure :
14  to setup
15    ;;Reset the world :
16    clear-all
17    reset-ticks
18
19    ;;Vid extension :
20    if vid:recorder-status = "recording"
21      [vid:record-view]
22
23    ;;Set the background to brown :
24    ask patches
25      [set pcolor 34]
26
27    ;;Create walls around the world to prevent bacteria from escaping :
28    ask patches with [count neighbors != 8]
29      [set pcolor grey]
30
31    ;;Set a list of index :
32    set index [0 123 4 5 6 7 8 9]
33
34    ;;Set a list of color for the chemoattractant :
35    set colour [19 181716 15 14 13 12 11 121]
36
37    ;;Set a list for lower bound, to link concentration of chemoattractant
          to coulour :
38    set i_low [0.001 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
39
40    ;;Set a list for upper bound, to link concentration of chemoattractant
          to coulour :
41    set i_up [0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 100]
42
43    ;;Initialise a variable for tumbling to false :
44    set tumble True
45
46    set compteur_tumble 1
47
48    set compteur_run 1
49  end
50
51  ;;Go Procedure :
52  to go
53    ;;Bacteria evolution :
54    ask bacteria
55      [bacteria-movement]
```

```
56
57    ;;Patches evolution :
58    ask patches
59      [evolution-patches]
60
61    ;;Patches of Chemoattractant evolution :
62    maj_patches_colour
63
64    ;;Video Extension
65    if vid:recorder-status = "recording"
66      [vid:record-view]
67
68    ;;We tick :
69    tick
70  end
71
72  ;CREATION PROCEDURES :
73
74  ;;Command to create bacteria on the coordinates specified by the mouse
75  to put-turtle-mouse
76    if mouse-down?
77      [create-bacteria 1
78        [setxy mouse-xcor mouse-ycor
79        set shape "bacteria_not_tumbling"
80        set color 85
81        set size 20]
82      ]
83  end
84
85  ;;Command to create source of attractant :
86  to source-mouse
87    if mouse-down?
88      [ask patch mouse-xcor mouse-ycor
89        [;;We place "value_source" of attractant on the case :
90        set c (c + value_source)
91
92        ;;If we switch source_cst to "On" we renew the quantity of
              attractant on the source :
93        if source_cst = true
94          [set source true
95          ask neighbors
96            [set source true]
97          ]
98
99        ;;We change the color of the patch :
100       set pcolor item 4colour
101
102       ;;Whe also put attractant on the neighbors :
103       ask neighbors
104         [set c (c + value_source)
105         set pcolor item 4colour]
106         display
107       ]
108     ]
109 end
```

```
110
111 ;WALL PROCEDURES :
112
113 ;;Command to draw wall :
114 to draw-wall
115   if mouse-down?
116     [ask patch mouse-xcor mouse-ycor
117         [set pcolor grey
118
119         ;;We ask also the neighbors to be walls :
120         ask neighbors
121           [ask neighbors
122              [set pcolor grey]
123              display
124           ]
125         ]
126     ]
127 end
128
129 ;;Command to remove wall :
130 to remove-wall
131   if mouse-down?
132     [ask patch mouse-xcor mouse-ycor
133       [set pcolor 34
134       set c 0
135
136       ;;We also remove wall around :
137       ask neighbors
138         [set pcolor 34
139         set c 0]
140         display
141       ]
142     ]
143 end
144
145 ;;Command to draw maze easily :
146 to draw-walls-height
147   if mouse-down?   [
148     foreach index [ [iii] ->
149       ask patch mouse-xcor (mouse-ycor + iii)
150         [ set pcolor grey
151           ask neighbors [set pcolor grey]
152      display ]
153   ]]
154 end
155
156 ;;Command to draw maze easily :
157 to draw-walls-width
158   if mouse-down?   [
159     foreach index [ [iii] ->
160       ask patch (mouse-xcor + iii) mouse-ycor
161         [ set pcolor grey
162           ask neighbors [set pcolor grey]
163      display ]
164   ]]
```

```
end

;MOVEMENT PROCEDURES :

;;Procedure for bacteria movement :
to bacteria-movement
  ;;If the patch ahead is nobody (the limit of the world) or grey (the
      patch color that represents walls in our model) we turn in a
      random direction, else we move :
  ifelse [pcolor] of patch-ahead 1= grey
     ;;Turn in a random direction :
     [lt random-float 360]

     ;;Biaised movement relative to concentration :
     [;;"c_devant" is the concentration of the patch ahead the bacterium
         :
     let c_devant 0
     ask patch-ahead 1
       [set c_devant c]

     ;;"diff" is the difference between the concentration ahead and the
         concentration where the bacterium is :
     let diff (c_devant - c)

     ;;We can change the sensitivity of the bacteria: Sensitivity has
         been mesured by the observed fact that bacteria can go up
         exponential gradient and it
     ;;imply that they can spot differences about 0.0001%
     ifelse diff > 0.000001
       [run-tumble-biaised]
       [run-tumble]
     ]

end

;;Command that represent the run-and-tumble movement of the bacteria
    (if there is not enough attractant) :
to run-tumble
  ;;If tumble is true we tumble and then we run, we have 3/10 chance to
      tumble when we are on a run phase :
  ifelse tumble
    ;;Tumbling :
    [set shape "bacteria_tumbling"
    set compteur_tumble compteur_tumble + 1
    set heading random 360
    set tumble False]

    ;;Run :
    [set shape "bacteria_not_tumbling"
    ifelse random 100<= 20
      [set tumble True]
      [forward 1
      set compteur_run compteur_run + 1]
    ]
end
```

```
212
213 ;;Command that represent the biaised movement of bacteria in response
        to attractant :
214 to run-tumble-biaised
215   ;;If tumble is true we tumble and then we run, we have 1/10 chance to
          tumble when we are on a run phase --> the movement is then
          biaised :
216   ifelse tumble
217     ;;Tumbling :
218     [set shape "bacteria_tumbling"
219     set heading random 360
220     set compteur_tumble compteur_tumble + 1
221     set tumble False]
222
223     ;;Run phase are longer :
224     [set shape "bacteria_not_tumbling"
225     ifelse random 100<= 1
226       [set tumble True]
227       [forward 1
228       set compteur_run compteur_run + 1]
229     ]
230 end
231
232 ;PATCHES PROCEDURE :
233
234 ;;Procedure to evolve patches :
235 to evolution-patches
236   ;;We set "fuite" as the quantity of attractant that will leave the
          patch, "diffusion" is a parameter (slider) that ponderate the
          quantitity of attractant :
237   set fuite (diffusion * c)
238
239   ;;"number_case" is the number of cases that aren't grey (not walls),
          we set at 0:
240   let number_case 0
241
242   ;;Analyse the number of case that aren't grey :
243   ask neighbors
244     [
245     if pcolor != grey
246       [set number_case (number_case + 1)]
247     ]
248
249   ;;We substract the amount of attractant that leave the patch to the
          concentration of the patch :
250   if number_case != 0
251     [set c (c - fuite)]
252
253   ;;If the color is not grey we diffuse the amount of attractant divide
          by the number of case on wich we can diffuse (those who aren't
          grey) :
254   ask neighbors
255     [
256     if pcolor != grey
257       [set c (c + (fuite / number_case))]
```

```
258     ]
259
260   ;;If we switch source_cst to "On" we renew the quantity of attractant
          on the source :
261   if source = true
262     [set c c + 1]
263 end
264
265
266 ;;Procedure to update the colour of the patches :
267 to maj_patches_colour
268   ask patches with [pcolor != grey]
269     [
270     foreach index [[i] ->
271         if c > item i i_low and c <= item i i_up
272           [set pcolor item i colour]
273       ]
274     if c < item 0i_low
275       [set pcolor 34]
276     ]
277 end
278
279 ;VIDEO PROCEDURES :
280
281 ;;Video Extension: Taken from the "Movie Recording Example" model
        available on NetLogo Library :
282 to start-recorder
283   carefully [ vid:start-recorder ] [ user-message error-message ]
284 end
285
286 ;;Video Extension: Taken from the "Movie Recording Example" model
        available on NetLogo Library :
287 to reset-recorder
288   let message (word
289     "If you reset the recorder, the current recording will be lost."
290     "Are you sure you want to reset the recorder?")
291   if vid:recorder-status = "inactive" or user-yes-or-no? message [
292     vid:reset-recorder
293   ]
294 end
295
296 ;;Video Extension: Taken from the "Movie Recording Example" model
        available on NetLogo Library :
297 to save-recording
298   if vid:recorder-status = "inactive" [
299     user-message "The recorder is inactive. There is nothing to save."
300     stop
301   ]
302   ;;Prompt user for movie location
303   user-message (word
304     "Choose a name for your movie file (the "
305     ".mp4 extension will be automatically added).")
306   let path user-new-file
307   if not is-string? path [ stop ] ;;Stop if user canceled
308   ;;Export the movie
```

```
  carefully [
    vid:save-recording path
    user-message (word "Exported movie to " path ".")
  ] [
    user-message error-message
  ]
end
```

6.2.2 Eukaryotic model

```
;;Extension: "vid" to make movies
extensions [ vid ]

;;Global variables: "colour" to represent chemoattractant, "tumble" to
    make movement of bacteria, "fuite" to make chemoattractant diffuse,
    "i_low" lower bound for concentration of chemoattractant, "i_up"
    upper bound for concentration of chemoattractant, "index" for
    colours
globals [colour i_low i_up index fuite]

;;Breed: "bacteria" represent all the turtles, "bacterium" is one of
    the turtles
breed [ amibes amibe ]

;;Patches variables: "c" is the concentration of chemoattractant on
    each patch, "source" is a boolean to put a constant source of
    chemoattractant
patches-own [c source]

;;Setup Procedure :
to setup
  ;;Reset the world :
  clear-all
  reset-ticks

  ;;Vid extension :
  if vid:recorder-status = "recording"
    [vid:record-view]

  ;;Set the background to brown :
  ask patches
    [set pcolor 34]

  ;;Create walls around the world to prevent bacteria from escaping :
  ask patches with [count neighbors != 8]
    [set pcolor grey]

  ;;Set a list of index :
  set index [0 123 4 5 6 7 8 9]

  ;;Set a list of color for the chemoattractant :
  set colour [19 181716 15 14 13 12 11 121]

```

```
  ;;Set a list for lower bound, to link concentration of chemoattractant to coulour :
  set i_low [0.001 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]

  ;;Set a list for upper bound, to link concentration of chemoattractant to coulour :
  set i_up [0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 100]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; Main procedure
to go
  ;;Bacteria evolution :
  ask amibes
    [amibe-movement]

  ;;Patches evolution :
  ask patches
    [patches-evolution]

  ;;Patches of Chemoattractant evolution :
  maj_patches_colour

  ;;Video Extension
  if vid:recorder-status = "recording"
    [vid:record-view]

  ;;We tick :
  tick
end

;MOVEMENT PROCEDURES :

;;Procedure to move the amibes :
to amibe-movement
  ifelse (random 10) <= (5 * (c + 0.5))
    [mouvement-aleatoire]
    [chemotaxis_amibe]
    set c (c - (conso-c-amibes * c))
end

;;Procedure to move amibes when they are attracted :
to chemotaxis_amibe
  ask amibes [
    let c_temp 0
    ask neighbors [ set c_temp (c_temp + c)] ;; detection of the absence of chemical
    ifelse c_temp = 0
    [mouvement-aleatoire] ;; absence of chemical specie
    [ let c_max [0]
      let pxmax [0]    ;; creation of the temporary lists for the max concentration and
      let pymax [0]    ;;     the x and y coordinates of these patches
      let k 0          ;; temporary index
      ask neighbors with [not any? amibes-here] [
        if c > item 0c_max [
```

```
88            set c_max replace-item 0c_max c
89            set pxmax replace-item 0pxmax pxcor ;; replacement by higher
                  value
90            set pymax replace-item 0pymax pycor
91          ]
92          if c = item 0c_max [
93            set k (k + 1)
94            set c_max insert-item k c_max c
95            set pxmax insert-item k pxmax pxcor ;; exception if there is
                  several patches with the same max concentraion
96            set pymax insert-item k pymax pycor
97          ]
98        ]
99        let i_temp random ((length c_max));; exception where after two
              first patches with the same max concentration
100       foreach c_max [ [iii] ->       ;;    there is a new patch with
              higher concentration => we take the first index of the list to
              avoid error
101         if (iii) != item 0c_max [set i_temp 0] ;; iii= element of the
                list
102       ]
103       ifelse (item i_temp c_max - c) >= 0.01 ;; respect of the 1%
              sensibility of concentration difference (move only if
              difference big enough) [
104       if c_max != [0] [
105         setxy (item i_temp pxmax) (item i_temp pymax) ;; movement
106       ]]
107         [mouvement-aleatoire]
108
109     ]
110   ]
111 end
112
113 ;;Procedure to move amibes when they're not attracted :
114 to mouvement-aleatoire
115   set heading random 360   ;; random movement
116   if patch-ahead 1!= nobody and [pcolor] of patch-ahead 1!= grey[
117     forward 1]
118 end
119 ;CREATION PROCEDURES :
120
121 ;;Command to create agent on the coordinates specified by the mouse
122 to put-turtle-mouse
123   if mouse-down?
124     [reset-ticks
125      clear-plot
126       create-amibes 50
127       [setxy mouse-xcor mouse-ycor
128       set shape "amibe_shape"
129       set color 85
130       set size 10]
131     ]
132 end
133
134 ;;Command to create source of attractant :
```

```
135 to source-mouse
136   if mouse-down?
137     [ask patch mouse-xcor mouse-ycor
138       [;;We place "value_source" of attractant on the case :
139       set c (c + value_source)
140
141       ;;If we switch source_cst to "On" we renew the quantity of
              attractant on the source :
142       if source_cst = true
143         [set source true
144         ask neighbors
145           [set source true]
146         ]
147
148       ;;We change the color of the patch :
149       set pcolor item 4colour
150
151       ;;Whe also put attractant on the neighbors :
152       ask neighbors
153         [set c (c + value_source)
154         set pcolor item 4colour]
155         display
156       ]
157     ]
158 end
159
160 ;WALL PROCEDURES :
161
162 ;;Command to draw wall :
163 to draw-wall
164   if mouse-down?
165     [ask patch mouse-xcor mouse-ycor
166         [set pcolor grey
167
168         ;;We ask also the neighbors to be walls :
169         ask neighbors
170           [ask neighbors
171              [set pcolor grey]
172              display
173           ]
174         ]
175     ]
176 end
177
178 ;;Command to remove wall :
179 to remove-wall
180   if mouse-down?
181     [ask patch mouse-xcor mouse-ycor
182       [set pcolor 34
183       set c 0
184
185       ;;We also remove wall around :
186       ask neighbors
187         [set pcolor 34
188         set c 0]
```

```
189         display
190       ]
191     ]
192 end
193
194 ;;Command to draw maze easily :
195 to draw-walls-height
196   if mouse-down?   [
197     foreach index [ [iii] ->
198       ask patch mouse-xcor (mouse-ycor + iii)
199         [ set pcolor grey
200           ask neighbors [set pcolor grey]
201      display ]
202   ]]
203 end
204
205 ;;Command to draw maze easily :
206 to draw-walls-width
207   if mouse-down?   [
208     foreach index [ [iii] ->
209       ask patch (mouse-xcor + iii) mouse-ycor
210         [ set pcolor grey
211           ask neighbors [set pcolor grey]
212     display ]
213   ]]
214 end
215
216 ;PATCHES PROCEDURE :
217
218 ;;Procedure to update the colour of the patches :
219 to maj_patches_colour
220   ask patches with [pcolor != grey]
221     [
222     foreach index [[i] ->
223         if c > item i i_low and c <= item i i_up
224           [set pcolor item i colour]
225       ]
226     if c < item 0i_low
227       [set pcolor 34]
228     ]
229 end
230
231
232 to patches-evolution
233   ;;We set "fuite" as the quantity of attractant that will leave the
        patch, "diffusion" is a parameter (slider) that ponderate the
        quantitity of attractant :
234   set fuite (diffusion * c)
235
236   ;;"number_case" is the number of cases that aren't grey (not walls),
        we set at 0:
237   let number_case 0
238
239   ;;Analyse the number of case that aren't grey :
240   ask neighbors
```

```
241     [
242     if pcolor != grey
243         [set number_case (number_case + 1)]
244     ]
245
246   ;;We substract the amount of attractant that leave the patch to the
        concentration of the patch :
247   if number_case != 0
248     [set c (c - fuite)]
249
250   ;;If the color is not grey we diffuse the amount of attractant divide
        by the number of case on wich we can diffuse (those who aren't
        grey) :
251   ask neighbors
252     [
253     if pcolor != grey
254       [set c (c + (fuite / number_case))]
255     ]
256
257   ;;If we switch source_cst to "On" we renew the quantity of attractant
        on the source :
258   if source = true
259     [set c c + 1]
260 end
261
262 ;VIDEO PROCEDURES :
263
264 ;;Video Extension: Taken from the "Movie Recording Example" model
      available on NetLogo Library :
265 to start-recorder
266   carefully [ vid:start-recorder ] [ user-message error-message ]
267 end
268
269 ;;Video Extension: Taken from the "Movie Recording Example" model
      available on NetLogo Library :
270 to reset-recorder
271   let message (word
272     "If you reset the recorder, the current recording will be lost."
273     "Are you sure you want to reset the recorder?")
274   if vid:recorder-status = "inactive" or user-yes-or-no? message [
275     vid:reset-recorder
276   ]
277 end
278
279 ;;Video Extension: Taken from the "Movie Recording Example" model
      available on NetLogo Library :
280 to save-recording
281   if vid:recorder-status = "inactive" [
282     user-message "The recorder is inactive. There is nothing to save."
283     stop
284   ]
285   ;;Prompt user for movie location
286   user-message (word
287     "Choose a name for your movie file (the "
288     ".mp4 extension will be automatically added).")
```

```
  let path user-new-file
  if not is-string? path [ stop ] ;;Stop if user canceled
  ;;Export the movie
  carefully [
    vid:save-recording path
    user-message (word "Exported movie to " path ".")
  ] [
    user-message error-message
  ]
end
```

6.2.3 Immune System model

```
extensions [ vid ]

globals [couleur intervalles_bas intervalles_haut index pathogene-death
    macrophage-death]

breed [ pathogenes pathogene ]

breed [ macrophages macrophage ]

breed [ lymphocyte_Bs lympocyte_B ]

macrophages-own [ compteur ]

patches-own [c1 c2 c3]

to setup
  clear-all

  if vid:recorder-status = "recording"
    [vid:record-view]

  ask patches with [count neighbors != 8]
    [set pcolor grey]

  set index [0 123 4 5 6 7 8 9]

  set couleur [19 181716 15 14 13 12 11 black]

  set intervalles_bas [0.001 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]

  set intervalles_haut [0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 100]

  ask patches [
    set pcolor 34
    set c1 0
    set c2 0
    set c3 0]

  reset-ticks
end
```

```
40
41  to go
42    procedure_macrophage
43
44    procedure_lymphocytes
45
46    procedure_pathogenes
47
48    diffusion_c1
49
50    diffusion_c2
51
52    diffusion_c3
53
54    maj_couleur_patches
55
56    tick
57  end
58
59  to procedure_macrophage
60    ask macrophages [
61      let target one-of pathogenes-here
62      ifelse target != nobody and compteur < 70
63        [move-to target
64        ask target [ die ]
65        set compteur (compteur + 1)
66        set pathogene-death pathogene-death + 1]
67
68        [ifelse (random 10) <= (5 * (c1 + 0.5))
69          [mouvement-aleatoire]
70
71          [chemotaxis_macrophage]]
72
73    if compteur > 3
74      [set c2 c2 + 5]
75
76    set c1 (c1 - (conso-c1-macro * c1))
77    ]
78  end
79
80  to procedure_lymphocytes
81    ask lymphocyte_Bs [
82      ifelse (random 10) <= (5 * (c1 + 0.5))
83        [mouvement-aleatoire]
84        [chemotaxis_lymphocyte_B]
85      if c2 >= 0.5 [set c3 c3 + 10]
86      set c2 (c2 - (conso-c2-lympho * c2))
87    ]
88  end
89
90  to procedure_pathogenes
91    ask pathogenes [
92      set c1 c1 + 1
93      if c3 >= 0.5
94        [set pathogene-death pathogene-death + 1
```

```
95        ask pathogenes-here [die]]
96      mouvement-aleatoire
97    ]
98  end
99
100 to maj_couleur_patches
101   ask patches with [pcolor != grey]
102     [foreach index [ [ii] ->
103       if c1 > item ii intervalles_bas and c1 <= item ii intervalles_haut
104       [set pcolor item ii couleur]
105      ]
106     if c1 < item 0intervalles_bas
107       [set pcolor 34]
108     ]
109 end
110
111 to diffusion_c1
112   ask patches [
113     let fuite (taux_diffusion * c1)
114     let cc c1
115     let o 0
116     ask neighbors
117       [if pcolor != grey and c1 < cc
118         [ set o (o + 1)]]
119     if o != 0
120       [set c1 (c1 - fuite)
121       ask neighbors
122         [if pcolor != grey and c1 < cc
123           [set c1 (c1 + (fuite / o))]]
124       ]
125   ]
126 end
127
128 to diffusion_c2
129   ask patches [
130     let fuite (taux_diffusion * c2)
131     let o 0
132     ask neighbors
133       [if pcolor != grey
134         [set o (o + 1)]]
135     if o != 0
136       [set c2 (c2 - fuite)]
137     ask neighbors
138       [if pcolor != grey
139         [set c2 (c2 + (fuite / o))]]
140   ]
141 end
142
143 to diffusion_c3
144   ask patches [
145     let fuite (taux_diffusion * c3)
146     let o 0
147     ask neighbors
148       [if pcolor != grey
149         [set o (o + 1)]]
```

```
150       if o != 0
151         [set c3 (c3 - fuite)]
152       ask neighbors [ if pcolor != grey [set c3 (c3 + (fuite / o))]]
153   ]
154 end
155
156 to  chemotaxis_macrophage
157   ask macrophages [
158     let c_temp 0
159     ask neighbors
160       [set c_temp (c_temp + c1)]
161     ifelse c_temp = 0
162       [mouvement-aleatoire]
163       [let c_max [0]
164       let pxmax [0]
165       let pymax [0]
166       let k 0
167       ask neighbors with [not any? macrophages-here]
168       [if c1 > item 0c_max [
169          set c_max replace-item 0c_max c1
170          set pxmax replace-item 0pxmax pxcor
171          set pymax replace-item 0pymax pycor]
172       if c1 = item 0c_max [
173         set k (k + 1)
174         set c_max insert-item k c_max c1
175         set pxmax insert-item k pxmax pxcor
176         set pymax insert-item k pymax pycor
177         ]
178       ]
179       let i_temp random ((length c_max))
180       foreach c_max [ [iii] ->
181         if (iii) != item 0c_max [set i_temp 0]
182       ]
183
184       ifelse (item i_temp c_max - c1) >= 0.01 and c_max != [0]
185         [setxy (item i_temp pxmax) (item i_temp pymax)]
186         [mouvement-aleatoire] ;;]
187
188     ]
189   ]
190 end
191
192 to  chemotaxis_lymphocyte_B
193   ask lymphocyte_Bs [
194     let c_temp 0
195     ask neighbors [ set c_temp (c_temp + c2)]
196     ifelse c_temp = 0
197     [set heading random 360
198       if patch-ahead 1!= nobody and [pcolor] of patch-ahead 1!= grey[
199           forward 1]]
200     [ let c_max [0]
201       let pxmax [0]
202       let pymax [0]
203       let k 0
204       ask neighbors with [not any? lymphocyte_Bs-here] [
```

```
205         if c2 > item 0c_max [
206             set c_max replace-item 0c_max c2
207             set pxmax replace-item 0pxmax pxcor
208             set pymax replace-item 0pymax pycor
209         ]
210         if c2 = item 0c_max [
211           set k (k + 1)
212           set c_max insert-item k c_max c2
213           set pxmax insert-item k pxmax pxcor
214           set pymax insert-item k pymax pycor
215         ]
216       ]
217       let i_temp random ((length c_max))
218       foreach c_max [ [iii] ->
219         if (iii) != item 0c_max [set i_temp 0]
220       ]
221       ifelse (item i_temp c_max - c2) >= 0.001
222       [
223       if c_max != [0] [
224         setxy (item i_temp pxmax) (item i_temp pymax)
225       ]]
226       [mouvement-aleatoire]
227       ]
228     ]
229 end
230
231 to mouvement-aleatoire
232   ifelse patch-ahead 1!= nobody and [pcolor] of patch-ahead 1!= grey and
          random 10< 5
233      [forward 1]
234      [set heading random 360]
235 end
236
237 to draw-wall
238   if mouse-down?
239     [ask patch mouse-xcor mouse-ycor
240       [set pcolor grey
241        ask neighbors [set pcolor grey
242        ask neighbors[set pcolor grey]]
243        display ]
244     ]
245 end
246
247 to remove-wall-espece
248   if mouse-down?
249     [ask patch mouse-xcor mouse-ycor
250       [ set pcolor 34
251         set c1 0
252         set c2 0
253         set c3 0
254         ask neighbors[
255           set pcolor 34
256           set c1 0
257           set c2 0
258           set c3 0
```

```
          ask neighbors [set pcolor 34
            set c1 0
            set c2 0
            set c3 0]]
          display ]
    ]
end

;;Command to draw maze easily :
to draw-walls-height
  if mouse-down?   [
    foreach index [ [iii] ->
      ask patch mouse-xcor (mouse-ycor + iii)
        [ set pcolor grey
          ask neighbors [set pcolor grey]
     display ]
  ]]
end

;;Command to draw maze easily :
to draw-walls-width
  if mouse-down?   [
     foreach index [ [iii] ->
      ask patch (mouse-xcor + iii) mouse-ycor
        [ set pcolor grey
          ask neighbors [set pcolor grey]
     display ]
  ]]
end

to mobile-source
  if mouse-down?
    [ask patch mouse-xcor mouse-ycor
      [ set c1 (c1 + value_mobile_source)
        set pcolor item 7couleur
        ask neighbors [
            set c1 (c1 + value_mobile_source)
            set pcolor item 7couleur]
          ]
        display
    ]
end

to put-pathogenes
  if mouse-down? [
  create-pathogenes quantite-turtle-mobile [
    setxy mouse-xcor mouse-ycor
    set shape "pathogen"
    set size 20
    ]
  display]
end

to put-macrophages
  if mouse-down? [
```

```
  create-macrophages quantite-turtle-mobile [
    setxy mouse-xcor mouse-ycor
    set shape "macrophage"
    set size 20
    set compteur 0
    ]
  display]
end

to put-lymphocyte_Bs
  if mouse-down? [
  create-lymphocyte_Bs quantite-turtle-mobile [
    setxy mouse-xcor mouse-ycor
    set shape "lymphocyte"
    set size 20
    ]
  display]
end

to start-recorder
  carefully [ vid:start-recorder ] [ user-message error-message ]
end

to reset-recorder
  let message (word
    "If you reset the recorder, the current recording will be lost."
    "Are you sure you want to reset the recorder?")
  if vid:recorder-status = "inactive" or user-yes-or-no? message [
    vid:reset-recorder
  ]
end

to save-recording
  if vid:recorder-status = "inactive" [
    user-message "The recorder is inactive. There is nothing to save."
    stop
  ]
  user-message (word
    "Choose a name for your movie file (the "
    ".mp4 extension will be automatically added).")
  let path user-new-file
  if not is-string? path [ stop ]
  carefully [
    vid:save-recording path
    user-message (word "Exported movie to " path ".")
  ] [
    user-message error-message
  ]
end
```

References

1. Painter, K. J. Mathematical Models for Chemotaxis and Their Applications in Self-Organisation Phenomena. *Journal of Theoretical Biology* **481,** 162–182 (2019).
2. Wilkinson, P. C. Chemotaxis. *Encyclopedia of Immunology,* 533–37 (1998).
3. Manes, S. Mastering time and space: immune cell polarization and chemotaxis. *Seminars in Immunology* **17,** 77–86 (2005).
4. Salek, M. Bacterial chemotaxis in a microfluidic T-maze reveals strong phenotypic heterogeneity in chemotactic sensitivity. *Nat Commun* **10** (2019).
5. Hillen T., e. K. J. P. A User's Guide to PDE Models for Chemotaxis. *Journal of Mathematical Biology* **58.1,** 183–217 (2009).
6. Chiacchio Ferdinando, e. a. Agent-Based Modeling of the Immune System: NetLogo, a Promising Framework. *BioMed Research International* (2014).
7. Yongho Kim, Y. C. Learning finite difference methods for reaction-diffusion type equations with FCNN. *Computers & Mathematics with Applications* **123,** 115–122 (2022).
8. Macnab, R. M. & Koshland, D. E. The Gradient-Sensing Mechanism in Bacterial Chemotaxis. *Proceedings of the National Academy of Sciences* **69,** 2509–2512 (1972).
9. Tweedy L Thomason PA, P. P. Seeing around corners: Cells solve mazes and respond at a distance using attractant breakdown. *Science* (2020).
10. Tokarski Christian, e. a. Agent-Based Modeling Approach of Immune Defense against Spores of Opportunistic Human Pathogenic Fungi. *Frontiers in Microbiology* **3,** 129 (2012).
11. Adler, J. A Method for Measuring Chemotaxis and Use of the Method to Determine Optimum Conditions for Chemotaxis by Escherichia coli. *Microbiology* **74,** 77–91 (1973).
12. Tweedy, L. How Should Eukaryotic Chemotaxis be Measured? (2023).
13. Singh S. P. Paschke P., T. L. & H., I. R. AKT and SGK kinases regulate cell migration by altering Scar/WAVE complex activation and Arp2/3 complex recruitment. *Front Mol Biosci* **9** (2022).
14. Schagat TL, W. J. Surfactant protein A differentially regulates peripheral and inflammatory neutrophil chemotaxis. *Am J Physiol Lung Cell Mol Physiol* (2002).
15. D'Orsogna MR Suchard MA, C. T. Interplay of chemotaxis and chemokinesis mechanisms in bacterial dynamics. *Phys Rev E Stat Nonlin Soft Matter Phys* (2003).
16. Huang CH, I. P. Cell memory and adaptation in chemotaxis. *Proc Natl Acad Sci U S A.* (2014).