Multi-Agent Systems modeling of Quorum sensing: Bioluminescence in *Aliivibrio Fischeri*

Costantino Carugno, Federico di Credico, Francesco Puccioni

September 20, 2018

Abstract

Quorum sensing is a method used by a large variety of bacteria to regulate gene expression depending on cell density. Bacteria can synchronize population behavior using signaling molecules called autoinducers: as the density of quorum sensing bacterial cells increases so does the concentration of the autoinducer. Detection of signal molecules by bacteria acts as stimulation which leads to altered gene expression once the minimal threshold is reached. Quorum sensing plays critical roles in regulating diverse cellular functions in bacteria, including bioluminescence, antibiotic resistance, virulence gene expression and biofilm formation.

Historically, this phenomenon has been firstly observed in *Aliivibrio Fischeri*, a bacterium found in marine environments, which synthesizes an autoinducer that regulates the synthesis of an enzyme to produce bioluminescence. In this short paper, we aim at reproducing this transition via a multi-agent model of a virtual bacterial colony and, using the *GAMA* platform, we run investigative simulations to achieve comprehensive results.

Quorum Sensing

Quorum sensing is the regulation of gene expression in response to fluctuations in cell-population density. Quorum sensing bacteria produce and release chemical signal molecules called autoinducers that increase in concentration as a function of the cell density. The detection of a minimal threshold stimulatory concentration of an autoinducer leads to an alteration in gene expression.

Gram-positive and Gram-negative bacteria use quorum sensing communication circuits to regulate a diverse array of physiological activities. These processes include symbiosis, virulence, competence, conjugation, antibiotic production, motility, sporulation, and biofilm formation [2].

This signaling system relies on the ability of such bacteria to actively synthesize small signal molecules intracellularly, which are then passively, or actively exchanged with the surrounding environment. Accumulation of the signal molecules is thus commensurate with the increase in bacterial population, and,

when the population density exceeds a "quorate" threshold, the corresponding levels of signal can induce a synchronized response in gene expression throughout the population. These signal molecules, named autoinducers, trigger the quorum sensing process by binding to a connected receptors, which in turn regulates transcription of many genes that are involved in the cell-density-dependent behavior [1].

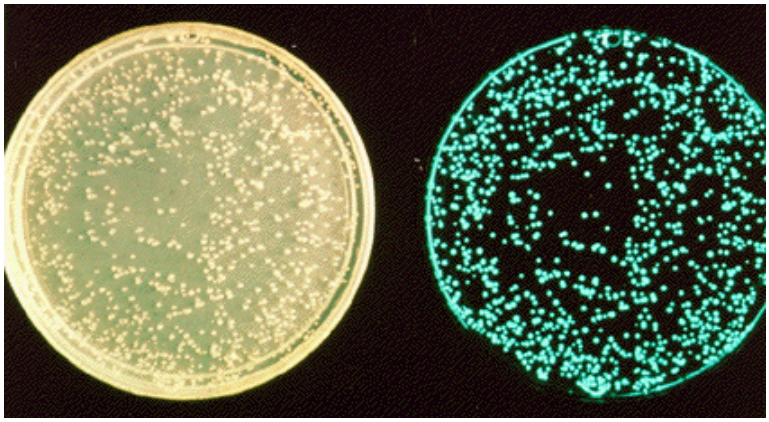

Figure 1: An *Aliivibrio Fischeri* colony on a Petri dish in the light (left) and in the dark (right)

Some of the best-known examples of quorum sensing come from studies of bacteria, as in the biolumiscence transition observed in figure 1. In addition to its function in biological systems, quorum sensing has several useful applications for computing and robotics. In general, quorum sensing can function as a decision-making process in any decentralized system in which the components have: (a) a means of assessing the number of other components they interact with and (b) a standard response once a threshold number of components is detected. Social insect colonies are an excellent example of a decentralized system, because no individual is in charge of directing or making decisions for the colony. Several groups of social insects, from ants to termites to bees, have been shown to use quorum sensing in a process that resembles collective decision-making [3].

Bioluminescence in *Aliivibrio Fischeri*

Autoinduction was first discovered in 1970 in the bacterium *Aliivibrio Fischeri* (named after Bernhard Fischer, a German microbiologist) that has since become a key research organism for examination of microbial bioluminescence, quorum sensing, and bacterial-animal symbiosis. This Gram-negative, rod-shaped bacterium is found globally in marine environments, predominantly in symbiosis with various marine animals, such as the *Hawaiian bobtail squid* (*Euprymna scolopes*). Free-living *Aliivibrio Fischeri* cells are found in very low quantities (almost undetectable) in almost all oceans of the world, preferentially in temperate and subtropical waters and survive on decaying organic matter in the water.

Aliivibrio Fischeri cells in the ocean inoculate the light organs of juvenile squid and fish. Ciliated cells within the animals' photophores (light-producing organs) selectively draw in the symbiotic bacteria. These cells promote the growth of the symbionts and actively reject any competitors. The bacteria cause these cells to die off once the light organ is sufficiently colonised.

The light organs of certain squid contain reflective plates that intensify and direct the light produced, due to proteins known as reflectins. They regulate the light for counter-illumination camouflage, requiring the intensity to match that of the sea surface above, to hunt for food and hide from predators. Each morning, when the camouflage is not needed anymore, the squid expels 90% of the symbiotic bacteria in their light organ in a process known as *venting*. Venting is thought to provide the free-living inoculum source for newly hatched squid. The circadian rhythm controls light expression, which is induced through population-dependent quorum sensing.

The bioluminescence transition in *Aliivibrio Fischeri* is achieved through the transcription of the *Lux* operon, a functioning unit of DNA containing a set of genes - namely *luxCDABEG* - and it's regulated by a two-gene system comprised of the proteins: *luxR* and *luxI*. More precisely, *LuxI* is responsible for synthesizing the autoinducer molecule *Acyl-Homoserine Lactone* - often simply called *AHL* - which quickly diffuses out of the cell, tipically without the chance to interact further with the *Lux* system inside the bacterium. However, when the *AHL* autoinducer is at a high concentration outside of the cell, as in crowded conditions, a number of these cells is likely to re-enter the bacteria and bind to the regulatory molecule *LuxR*. The *LuxR-AHL* complex is now in the condition to bind to the *Lux* operon and activate transcription, enhancing expression of the target genes, including the ones dedicated to light emission. Thus more *LuxI* and more *AHL* are produced, which will ultimately further enhance the expression of this operon. The enzyme responsible for light production, *Luciferase*, which consists of the products of the *LuxA* and *LuxB* genes, catalyzes a redox reaction that produces oxidized and reduced chemical products, as well as blue-green light [4].

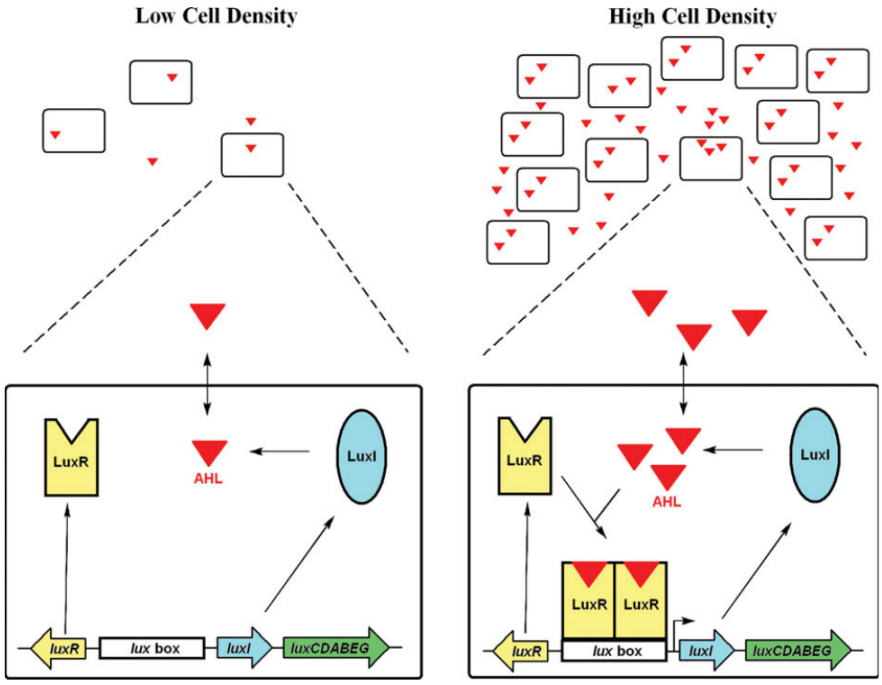

Figure 2: Acyl homoserine lactone (AHL)-dependent quorum sensing system as exemplified by LuxI/R system in V. fischeri.

A dummy model

The set up

In this paper we aim at reproducing the behavior of a bacterial colony using a technique known as agent-based modeling. To achieve this result we decided to use the *GAMA* platform, a powerful framework that provides possibility to use different behavioral architectures for programming agents in an integrated simulation environment.

Our agents (or, *species*, as called in *GAML*) live a closed square-box world of a given dimension and are one three possible kinds: cells, food or hormone.

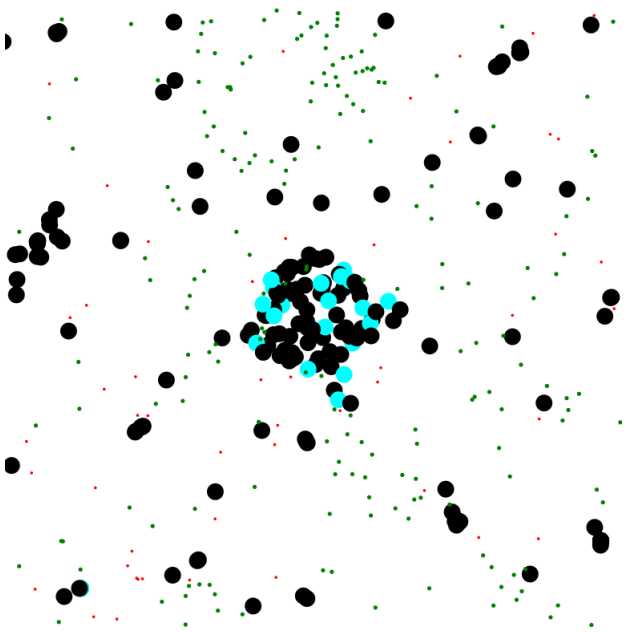

Figure 3: A simulation of the "dummy" model. You can see the different agents: the cells off (black) and on (cyan), the hormones (red) and the food (green).

- The food agents, represented by a small green circle, are chased after by the cells and, when a food agent comes in contact with a cell, it disappears. They are the only thing that the cells are interested in and the food guides the cells' movement. The food agents do not have a precise heading direction, they wander around the world mimicking a diffusion. A certain number of food agents is spawned every cycle each at a random point in space, but, once every two cycles, a number of food agents are generated inside a circle of a given dimension, centered at the origin. Thanks to this trick, more cells are drawn towards the center of the world, prompting the formation of a cluster, which you can clearly see in figure 3. This behavior is what defines this precise model to be "dummy", because the aggregation is artificially constructed via a programming rule, we can check if quorum sensing works in this scenario and we can measure how the crowding effect affects the bioluminescence transition.

- The hormone agents, depicted by a smaller circle of red color, represent in our model the autoinducer *AHL*. They are generated and emitted by the cell agents according to an algorithm explained in the next section. The hormones move in a diffusion fashion similarly to the food agents, but, since they are organic molecules, they "age", meaning that they get old each cycle and they disappear from the world once they turn fifty cycles old. The cell agents don't go after the hormones (just after the food), but if a hormone happens to cross paths with a cell agent it is inglobed

by it and re-emitted later. Once they are emitted their speed in space is progressively diminished, simulating a sort of friction.

- The cell agents are a little more complex than the previous agents. They are portayed as big black circles at the beginning of the simulation, and they are generated in a certain number at a random location in this experiment. They wander around freely (well, with a smaller wander amplitude than the other agents) but when a food agents enters their ray of perception, they head right at it, to eat it. If more than one food agent happens to be in their food-sensing range they will go after the closest one. After they have eaten a food agent they gain health but, since they are also organic cells, they are subject to aging and they lose health every cycle. Thus, once in a while, a cell dies, but the number of dead cells is not really significant, is just a way of having a sustainable population in the colony. The cell agents are the originally generators of the hormone, and they undergo a bioluminescence transition once they have inglobed enough of it, turning their color to cyan. If they don't absorb the hormone with enough frequency they turn off black once again. To achieve this behavior we had to implement a memory in the cell that accounts for the hormone intake, to be explained in detail in the next section.

Bioluminescence transition modeling

According to the biochemistry of quorum sensing we know that the bacteria produce a small amount of *AHL* via the *LuxI* protein. We also know that this production rate increases substantially with the intake of *AHL*, since when the *Lux* operon activates, more *LuxI* is produced, thus prompting to a non-linear model for the hormone production. However, an important factor to take into consideration is that when *AHL* is inglobed by the cell, it binds to *LuxR* and is used up by the transcription reaction. This means that we might get away simplifying our model so that the hormone are produced very rarely but they are emitted with a frequency proportional to the absorption frequency via a simple numerical factor:

$$f_{emission} = \alpha * f_{absorption}, \tag{1}$$

where α is a fixed proportionality constant. A possible improvement for a more accurate model would involve writing a (non-linear) differential equation to see if it leads to chaotic or predictable behavior. Bioluminescence in each cell is triggered via a hard threshold: if the absorption frequency is higher than a certain given number threshold the cell turns cyan, if is lower than the threshold it turns black again.

To calculate the absorption frequency, we have provided each cell agent with a "memory", a dynamic list of 11 slots that keeps track of the cycle number (the timestamp) in which a hormone is inglobed, as seen in figure 4 and 5.

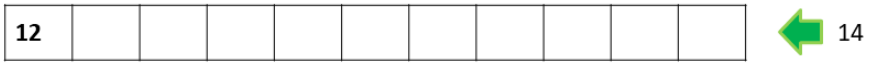

Figure 4: Example of a memory of a cell, it has absorbed a hormone at cycle 12 and at the current cycle (14).

Figure 5: The 14th cycle is added to the list and takes the first empty memory slot.

When the list reaches the maximum of 11 elements is full and for a new cycle to be stored, we need to forget about one element, so we empty the first slot, which contains the first value that has been recorded (figure 6), following the First-In-First-Out (FIFO) queuing protocol.

Figure 6: How the memory is managed once full, the FIFO method.

Each cycle of the simulation, if an agent has 11 elements stored in its memory, it calculates its absorption frequency. It should be noted that at the beginning of each simulation the agents are initialized with an empty memory and an absorption frequency equal to zero, thus this value can start to change only after having filled its list with 11 elements. As shown in figure 7, we first calculate the temporal distances between one absorption cycle and the other, the 10 results obtained are averaged over, thus obtaining an average period of absorption, $\bar{T}$, a number that expresses how many cycles of iteration, on average, pass between the absorption of a hormone and the next one, for each cell.

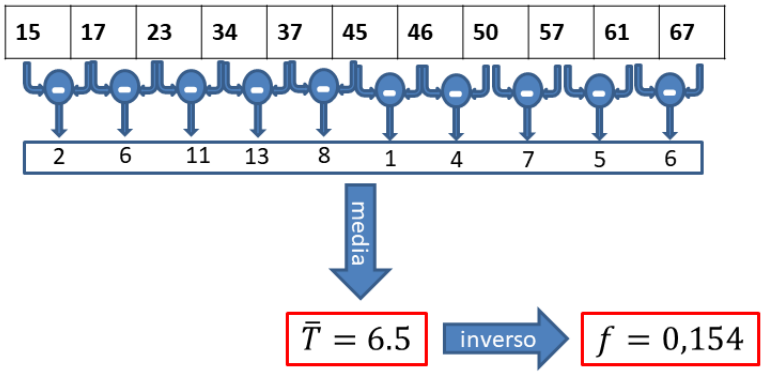

Figure 7: Calculation of the absorption frequency

To calculate the absorption frequency, we only need to take the inverse:

$$f_{absorption} = 1/\bar{T} \tag{2}$$

Finally, the agent, after having calculated the absorption frequency, determines the emission rate according to the equation (1).

Note that this mechanism is still incomplete, in fact according to the methodology described above, as the agents are initialized, no one emits and no one absorbs hormones at the beginning of the simulation, all the memories will be empty, the absorption frequencies will remain zero for an infinite time and none of the cells will become bioluminescent. On the other hand. it should also be noted that, with this methodology, if an agent no longer absorbs any hormones for a considerable period of time, it will continue to emit hormones always with the same frequency, which is an undesirable behavior, since we would like that if an agent does not absorb any hormone, then it should decrease its emission to zero.

There is a simple a trick that solves both problems at once, and is pictured in figure 8.

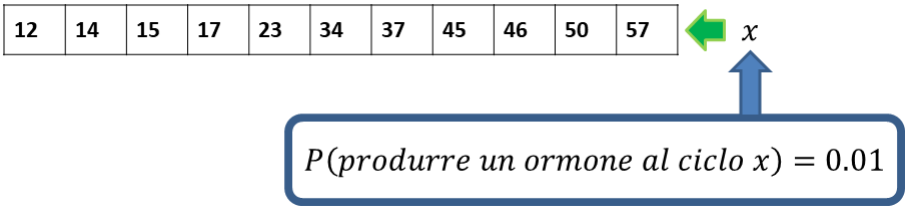

Figure 8: *Basic-memory* mechanism

At each iteration cycle, every agent adds to its memory an *absorption cycle* element with a 1 in 100 probability, without necessarily having to absorb a

hormone; if an agent triggers this probability and absorbs a hormone in the same iteration cycle, the element added to the list is only one. Thanks to this expedient, the agents will begin to produce hormones even if they are initialized with an empty memory, and there will also be a check in the particular case in which the absorption does not occur for a considerable period of time, as previously described. As a matter of fact, even such a small probability, causes the memory of each cell to refresh itself while still keeping the *absorption frequency* low enough, allowing the cell to turn off when too few hormones are inglobed. In a way this *basic-memory* mechanism resembles the natural synthesis of *AHL* by *LuxI*, which is supposed to be a small but crucial effect to trigger quorum sensing.

In figure 9 you can see a classical example of the bioluminescence transition in a simulation of the dummy model, divided in four stages:

1. cells are randomly scattered in the environment but food is generated more heavily in a circle at the center of the world;
2. cells are thus attracted to the center and begin to cluster;
3. hormones are started to appear, note that there are few in the center since they are swiftly inglobed;
4. bioluminescence is reached, there are some cells outside the cluster that are on but they will soon turn off, while lots of the cells inside the cluster are going to stay on.

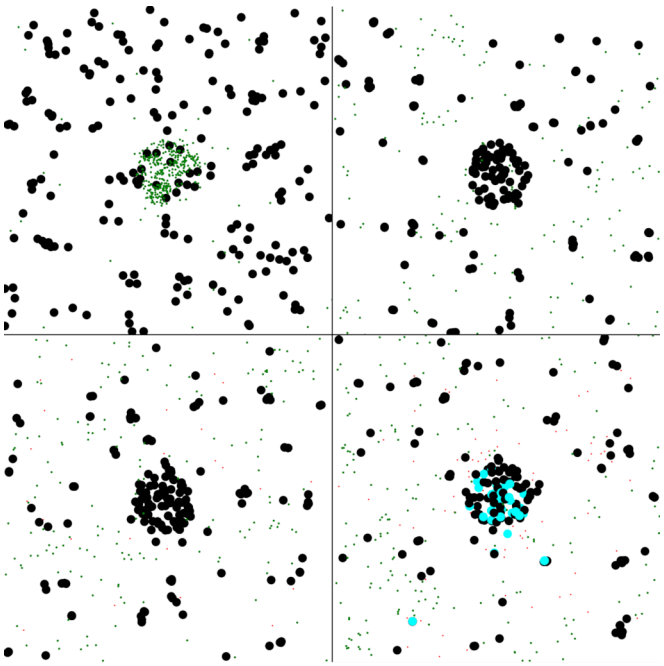

Figure 9: A typical run of the dummy model

Simulations

Our species have many parameters that can be tweaked (even at runtime!) to alter the simulation, using sliders in the experiment view. For example we can alter global variables such as the dimension of the environment, the number of the cells or the dimension of the cluster; we can change agent-specific parameters, lowering the bioluminescence threshold for cells or increasing their food-sensing horizon. There are lots of possible experiments to run just by changing these values, but we decided to focus on other kinds of simulations.

In this section we look at the results of an experiment aimed at understanding the connection between the density of the cells in a given area and their luminosity. The density is defined as:

$$d = \frac{N}{A}, \tag{3}$$

where N is the number of the cells that compose the system, and A is the total surface of the area in which they happen to be located, as calculated by the unit measure of the development platform *GAMA*.

Our luminosity is defined as the fraction of the cells that show bioluminescence, so:

$$P = \frac{N_l}{N}, \tag{4}$$

where N_l is the number of lit cells.

The experiment has been set up with these premises:

- cells don't have a *health* parameter;
- cells don't die, all agents instantiated from the beginning of the simulations are kept, none of them will be generated during runtime;
- there isn't any food present in the environment so the cells wander in a random way;
- the environment is a thorus;
- all simulations last 10000 iterations.

With these assumptions we have carried on 12 simulations, in each of which there are 10, 20, 30 and so on number of agents. This way, while keeping the environment at the same size, we obtain a different cell density at each simulation. In each of the simulations we saved the data N and N_l every 20 iterations. When the simulations were over we used *python* to plot the time series of the fraction of lit cells (figure 10).

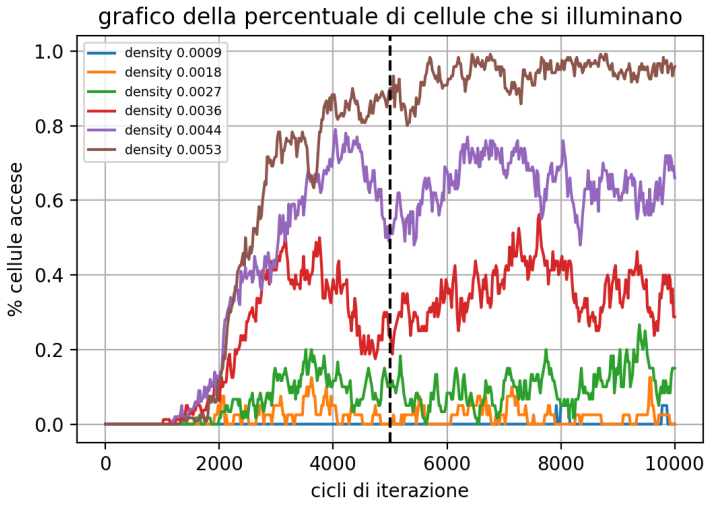

Figure 10: Time-series percentage of lit cells.

We measure a response time of about 2000 iterations, the time needed to wait to see the first cells turn on. Also from the same figure we estimate a relaxation time of about 5000 iterations, i.e. the time necessary for the system to reach equilibrium, from that moment on the fraction of bioluminescent cells is approximately constant, since it varies only within a small interval. This consideration has served us to make the next statistic, to calculate luminosity as a function of the density, we first calculated the time average of N_l from 5.000 iterations on, and then we divided by the density d. Given the fluctuations, we also calculated the standard deviation. Finally we have created the graph shown in figure 11.

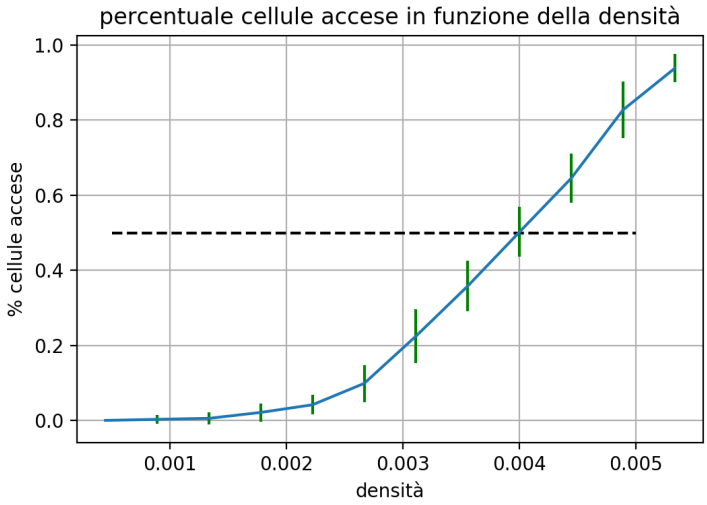

Figure 11: Percentage of lit cells as function of the density

We conclude this experiment observing that, through this mechanism of hormone exchange, in order to illuminate 50% of the cells there must be an average of 4 every 100 square units of measurement of the environment.

An improved model

Using the dummy model we have seen that agent-based programming can be suited to modeling quorum sensing and reproducing the bioluminescence transition. In the dummy model this behavior is forced upon using an artifical clustering, exploiting the food agent to cause the cells to crowd in the center. In nature this does not happen. The photophores of the Hawaiian bobtail squid provide a confortable environment for the bacteria, which are fed a solution of sugar and amino-acid to improve their health, so that they reproduce quickly. The clustering in the symbiotic environment is thus produced by bacterial replication.

To improve on our model we decide to remove the artificial clustering and substitute it with a replication process. We already provided the cell agents with a health parameter, so we can use it as a reproduction threshold: when it is reached the cell duplicates, calling an action named *reproduce* which creates a new cell on the spot while the health parameter of the original cell is dropped back at the minimum value.

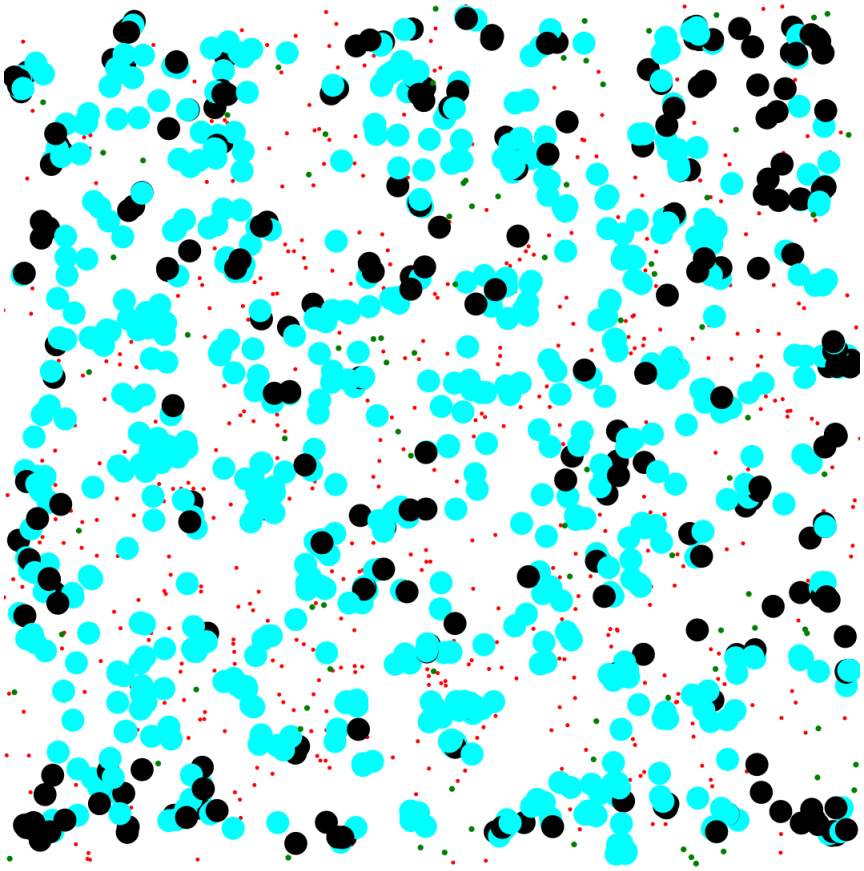

Figure 12: A final stage run of the improved model

A more accurate model should also account for the *venting* behavior. As explained in the introduction, venting is the process that regenerates a new bacterial colony every night, expelling about 90% of the *Vibrio* population and starting over with the ones left. To implement this phenomenon we decided that every 2000 cycle the day is to be over, and 95% of the cells, chosen randomly between lit and not lit, should die, while at the same time taking out all the hormone and all the food as well.

With this new model in our hand, we carried on new simulations that provided interesting insights for this phenomenon.

Simulations

Our aim in this experiment is to reproduce more faithfully the bioluminescence behavior as observed in nature. Our setup for the simulation is based on the following assumptions:

- set the initial cell number to 30;
- increase the food to 30 and generate it randomly;
- venting every 2000 cycles 95% of the cells.

Starting the simulation, as we can observe in figure 13, the number of cells increases very rapidly, given the abundance of the food available as we see in the second frame. In the third frame we notice the first cells becoming bioluminescent, as few hormones are produced and the food becomes scarce, while in the fourth frame the hormone is widespread and many cells are turned on.

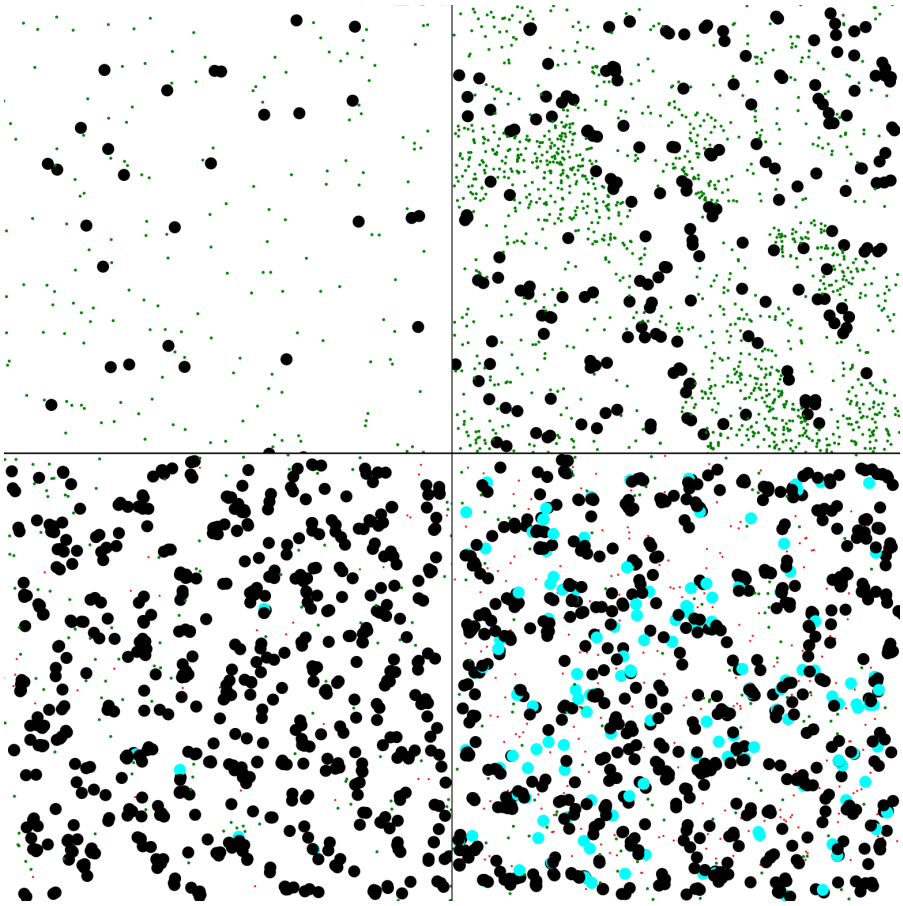

Figure 13: 4 different stages of the improved model

In the figure 14 we can see different graphs related to stage 2, 3 and 4 (one stage per line), where we observe the total cell population on the left side and the ordinary vs. bioluminescent cells on the right side. At the beginning the

population growth is quadratic, up until the point that the food is consumed, at about 300 cells, where duplication slows down and it begins to become linear, and it stays linear at least up to 1000 cycles. We can see that at this stage the bioluminescent cells are started to grow rapidly, pretty much exponentially.

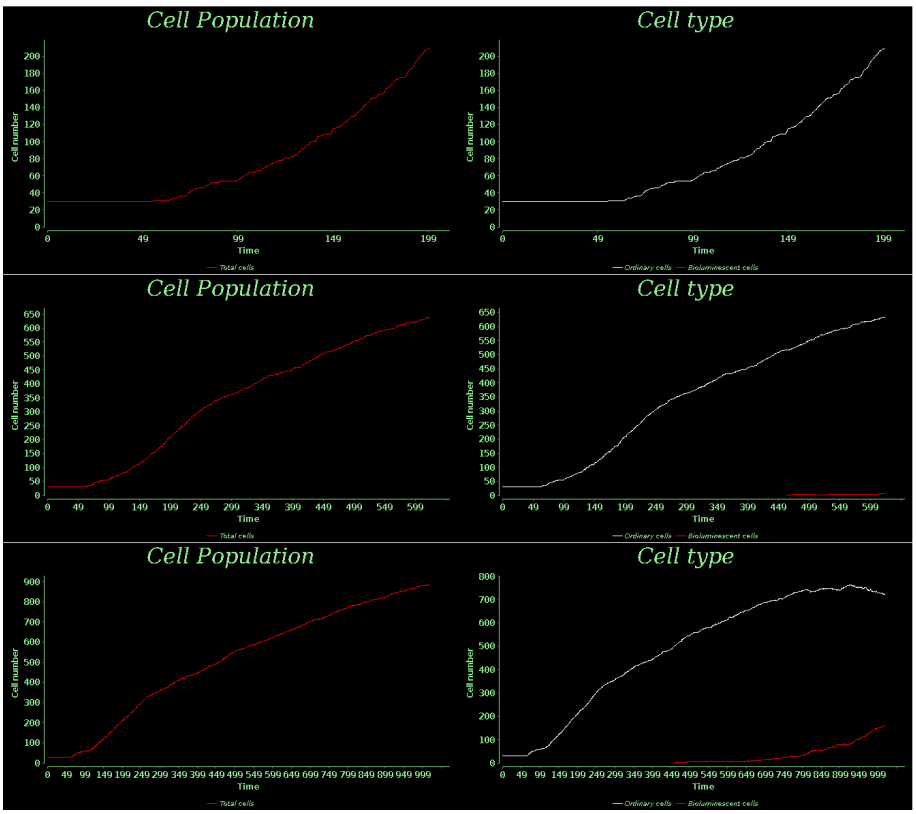

Figure 14: Graphs related to stage 2,3 and 4

In another run of the simulations, the results follow an almost identical pattern, as observed in figure 15. We highlight this graph to point out the moment in which the bioluminescent cells overtake the regular cells.

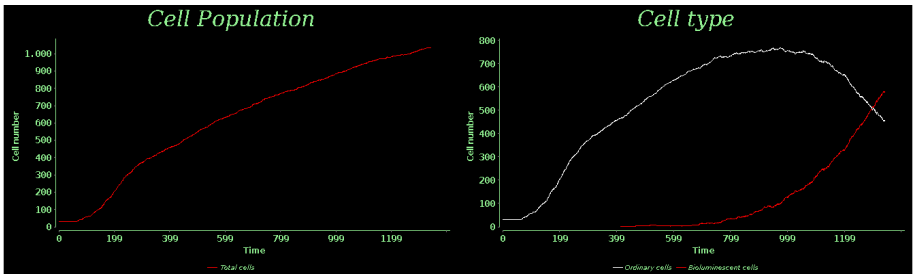

Figure 15: Another simulation after stage 4

Another very interesting moment not to be missed, is when venting occurs. In figure 16 we can see that right before venting the cell population is not increasing linearly anymore, but it has exhausted its capacity, it has saturated. This trend of "quadratic (or even stronger) growth - linear growth - saturation" is observed in every simulation and it's quite close to the natural behavior. At the same time we see that the bioluminescent cells draw an almost perfect sigmoid shape, following the saturation of the population. After venting occurs there are still about 50 bioluminescent cells active that proceed to turn off since they can't find any hormone around and the cycle can begin once again, more or less similarly as it was at the beginning.

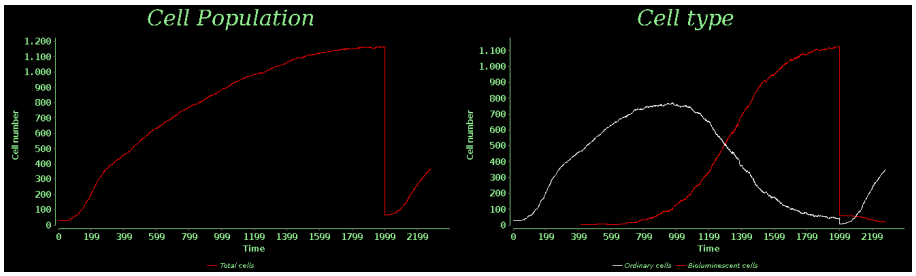

Figure 16: A graph of the simulation, right after venting

In figure 17 we have a look at a simulation after 4 venting cycles and the similarity between one cycle and the other it's quite strikingly. The simulation is incredibly consistent in its progress, following the precise cyrcadian rhythm of the natural phenomenon. The populations change smoothly through time and we can fine-tune the parameters that spring the bioluminescence transition (the α coefficient, the bioluminescence threshold etc.) whether we want to make it easier or harder, without changing the shape of the curves that characterize the system.

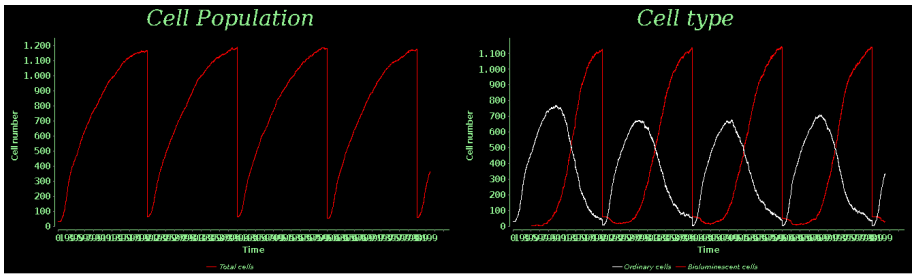

Figure 17: A graph of a simulation after 4 venting cycles

Conclusions

In this paper we have modeled a well known study-case phenomenon, quorum sensing, from a new perspective, the one of agent-based modeling. This innovative approach allowed us to model an emergent, global phenomenon, such as bioluminescece, encoding directly the behavior of single agents and retrieving it the emergent property via a simple linear interaction with added random fluctuations. The simulations carried on conclusively prove the validity of this model in reproducing the transition and mimicking the natural biological behavior. Numerical experiments were conducted to produce a quantitative analysis of the parameters used and physical quantities that can be compared with real-world experiments. Although we see ways of improving our model substantially, we value simplicity over complexity and we feel that the introduction of stochastics differential equations or any other sort of non-linear interaction might come with an added computational cost and carry many more problems than it aims to solve.

Appendix A: The source code of the improved model

```
model QuorumSensing

global {
        /* Global variables */
        int world_dimension <- 300 parameter:"World
            Dimension" min:20 max:1000 category:"Global";
        int ncell <- 30 parameter:"Number of Cells" min:1
             max:3000 category:"Global";
        float alpha <- 4/5 parameter:"Spreading factor"
            min:0.1 max:5.0 category:"Global";

        /* Cells attributes */
```

```
        int ray_of_perception <- 30 parameter:"Food
            sensing horizon" min:5 max: 100 category:"Cell
            ";
        int increase_of_health <- 40 parameter:"Health
            gained eating food" min:10 max:100 category:"
            Cell";
        float lightning_threshold <- 0.05 parameter:"
            Bioluminescence threshold" min:0.01 max: 0.1
            category:"Cell";
        int reproducing_threshold <- 1100 parameter:"
            Reproducing threshold" min:1000 max: 2000
            category:"Cell";

        /* Food attributes */
        int food_rnd <- 30 parameter:"Food generated
            randomly" min:1 category:"Food";

        geometry shape <- square(world_dimension);

        init{
                create cell number: ncell with: (location
                    : any_location_in(shape));//initial
                    cells are created randomly in our
                    space
        }

        reflex generate_food when: cycle mod 1 = 0 {
                create food number: food_rnd;
        }
        reflex save_result {
        save ("cycle: "+ cycle + "; num of cells: " +
            list(cell) count (each.health>0)) to: "results
            .csv" type: "csv" rewrite:false;
}

}

species cell skills: [moving] {
        /* Other cell attributes, hardcoded */
        float spreading_speed;
        int aging_speed;
        int dimension;
        float absorbing_frequency;
        rgb my_color;
        int max_health;
        int health;
```

```
int wandering_amplitude;
list<int> memory <- [];

init {
        spreading_speed <- 1.5;
        aging_speed <- 1;
        dimension <- 4;
        absorbing_frequency <- 0.0;
        my_color <- #black;
        health <- 500;
        wandering_amplitude <- 10;
}

reflex move_to_food when: !empty(food at_distance
    ray_of_perception) {   //cells go after
   species food if detected
        list<food> targets <- food at_distance
           ray_of_perception;
        do goto(targets with_min_of (distance_to(
           each, self)));
}

reflex move when: empty(food at_distance
   ray_of_perception) { //if food is not
   available cells just wander around
        do wander amplitude: wandering_amplitude;
}

reflex eat when: !empty(food at_distance
   dimension) {//cells gain health eating food
        ask one_of(food at_distance dimension) {
                do die;
        }
        health <- health + increase_of_health;
        if health > reproducing_threshold{
                do reproduce;
                health <- 500;
        }
}

reflex inglobe_hormone when: !empty(hormone
   at_distance dimension) {//hormons are also
   inglobed, if found
        ask one_of(hormone at_distance dimension)
            {
                do die;
```

```
        }
        add cycle to: memory;//number of cycle is
            added to a "memory" list, this is
           used later to calculate the frequency
        if length(memory) > 11 {//memory is of
           fixed length
                remove from: memory index: 0;
        }
}

reflex basic_memory when: flip(0.02) {
        add cycle to: memory;
        if length(memory) > 1 {
                if memory[length(memory)-2] =
                   memory[length(memory)-1] {
                        remove from: memory index
                           : length(memory) - 1;
                }
        }
        if length(memory) > 11 {
                remove from: memory index: 0;
        }
}

reflex get_old { //cells age each cycle
        health <- health - 1;
        if health <= 0 {
                do die;
        }
}

action reproduce{
        create cell number:1 with: (location:
           self.location+{rnd(5), rnd(5)});
}

reflex reset when: (cycle mod 2000 = 0 and cycle
   != 0 and flip(0.95)){
        do die;
}

reflex calculate_frequency { //this reflex
   calculates the frequency absorption of the
   hormons
        if length(memory) = 11 {
                list<int> periods <- [];
```

```
                        loop i from:1 to: length(memory)
                            -1 {
                                add item: memory[i]-
                                    memory[i-1] to:
                                    periods;
                        }
                        absorbing_frequency <- 1/mean(
                            periods);
                }
        }

        reflex spread_hormone when: flip(
            absorbing_frequency*alpha){ //the hormones are
             spread randomly in space, with a spreading
            frequency that is a multiple of the absorbing
            frequency
                int random_direction <- rnd(360);
                create hormone with: [location::{location
                    .x + dimension*cos(random_direction),
                    location.y + dimension*sin(
                    random_direction)}, heading::
                    random_direction, speed::
                    spreading_speed];
        }

        reflex switch_on when: absorbing_frequency >
            lightning_threshold {//if the absorbing
            frequency is high enough, cells begin to
            bioluminescence (switch color to red)
                my_color <- #cyan;
        }

        reflex switch_off when: absorbing_frequency <=
            lightning_threshold {//if the absorbing
            frequency is NOT high enough, cells stop to
            bioluminescence (switch color to black)
                my_color <- #black;
        }

        aspect base {
                draw circle(dimension) color: my_color;
        }
}

species food skills: [moving]{//we use the food agent to
    aggregate cells; it's very basic, they are spawned and
```

```
    only diffused in space
        reflex diffuse {
                do wander;
        }

        aspect base {
                draw circle(1) color: #green;
        }

        reflex out_of_board when: location.x <= 1 or
           location.x >= world_dimension-1 or location.y
           <= 1 or location.y >= world_dimension - 1{
                do die;
        }

        reflex reset when: cycle mod 2000 = 0{
                do die;
        }

}

species hormone skills: [moving]{
        int age <- 50;
        reflex get_old {//like cells, hormons age as well
                age <- age - 1;
                if age <= 0 {
                        do die;
                }
        }

        reflex diffuse {//diffusion is not simple
           wandering, the speed is progressively
           diminished, simulating a sort of friction
                if speed > 1 {
                        do move;
                        speed <- speed - 0.05;
                }
                else {
                        speed <- 0.95;
                        do wander;
                }
        }

        reflex out_of_board when: location.x <= 1 or
           location.x >= world_dimension - 1 or location.
           y <= 1 or location.y >= world_dimension - 1{
```

```
                do die;
        }

        reflex reset when: cycle mod 2000 = 0{
                do die;
        }

        aspect base {
                draw circle(0.7) color: #red;
        }

}

experiment quorum_sensing type:gui {
        output {
                        display "Cell Population" type:
                           java2D{
                        chart "Cell Population" type:
                            series background: #black
                            color: #lightgreen axes: #
                            lightgreen    size: {0.5,0.5}
                            position: {0, 0}
                        title_font_size: 32.0
                            title_font_style:'italic'
                            tick_font: 'Monospaced'
                            tick_font_size:14
                        tick_font_style:'bold'
                            x_serie_labels:cycle
                            x_tick_unit:200 x_label:'Time'
                             y_label:'Cell number'{
                        data "Total cells" value: (list(
                            cell) count (each.health>0))
                            accumulate_values:true color:#
                            red style:line marker_shape:
                            marker_empty;
                        }
                        chart "Cell type" type: series
                            background: #black color: #
                            lightgreen axes: #lightgreen
                            size: {0.5,0.5} position:
                            {0.5, 0}
                        title_font_size: 32.0
                            title_font_style:'italic'
                            tick_font: 'Monospaced'
                            tick_font_size:14
```

```
                tick_font_style:'bold'
                    x_serie_labels:cycle
                    x_tick_unit:200 x_label:'Time'
                     y_label:'Cell number'{
                data "Ordinary cells" value: (
                    list(cell) count (each.
                    my_color=#black))
                    accumulate_values:true color:#
                    white style:line marker_shape:
                    marker_empty;
                data "Bioluminescent cells" value
                    : (list(cell) count (each.
                    my_color=#cyan))
                    accumulate_values:true color:#
                    red style:line marker_shape:
                    marker_empty;
                }

            }
            display Experiment {
                    species hormone aspect: base;
                    species cell aspect: base;
                    species food aspect: base;
                    }
            }
}
```

References

[1] Zhi Li and Satish K. Nair. *Quorum sensing: How bacteria can coordinate activity and synchronize their response to external signals? Protein Science*, 21(10):1403–1417, 2012.

[2] Melissa B. Miller and Bonnie L. Bassler. *Quorum Sensing in Bacteria. Annual Review of Microbiology*, 55:165–199, 2001.

[3] Pratt S.C. *Quorum sensing by encounter rates in the ant Temnothorax albipennis. Behavioral Ecology*, 16(2):488–496, 2005.

[4] S.C. Winans W.C. Fuqua and E.P. Greenberg. *Quorum sensing in bacteria: the LuxR-LuxI family of cell density-responsive transcriptional regulators. Journal of Bacteriology*, 176(2):269–275, 1994.