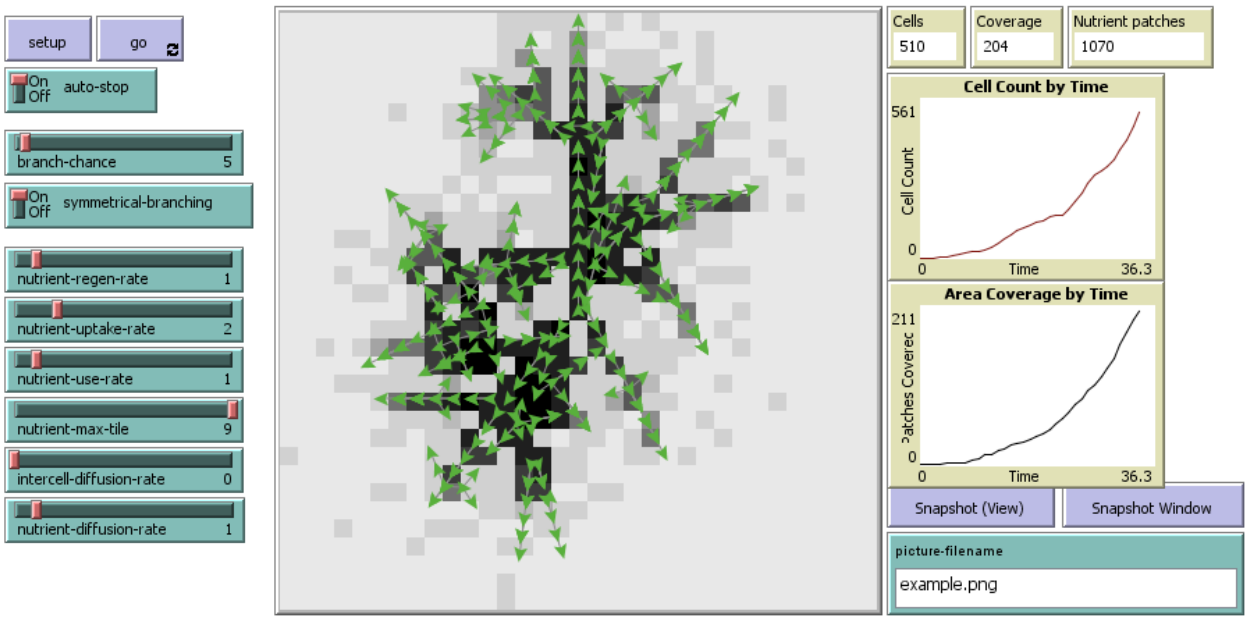
NetLogo, an Agent-Based Modelling Platform for Microbial Communities

Daniel Taylor

Abstract

The modelling of microbial communities can be used to predict emergent behaviours as well as internal cell mechanics that drive that behaviour. The agent-based modelling software NetLogo was tested as a tool for modelling microbial communities by a novice programmer. NetLogo was found to be quick to learn and user-friendly, with the ability to support a wide array of programmed agent behaviours as well as to quickly build a highly functional graphical user interface. NetLogo was successfully used to mmic a number of natural processes that are of current scientific interest, and one of these models was successfully used as an educational aid in a demonstration.



(Abstract Image – a screenshot of NetLogo open and running a model of colony growth. Cells are represented by the delta-shaped "turtles", NetLogo's name for agents, and grow on a grid of square "patches", 1x1 entities that can also possess properties and behaviours of their own. The user-defined interface, including buttons, switches, sliders, graphs, and text boxes all natively provided by NetLogo, can be seen to either side of the main view.)

Introduction

Many species of bacteria, long thought to act solely as individual entities, have been shown to form complex multicellular structures called biofilms. Biofilms typically consist of closely-clustered cells of one or more cooperating species, bound and protected by an adhesive, resilient extracellular matrix. Biofilms may also be porous to allow the passage of nutrients to the interior, and usually grant the constituent organisms substantial resilience against mechanical and chemical stress (1).

Biofilms are not the only form of multicellularity that bacteria display, however. Long before the discovery of biofilms, Streptomycetes were known to form large structures in which cells do not truly divide – instead, partially-formed intercellular walls create a series of compartments along filaments called hyphae, throughout which cellular components are free to travel (1)(2). Other species form chains of cells through incomplete cell division. Besides the potential benefits of resistance to mechanical and chemical attack, multicellularity also enables bacterial populations to cooperate in seeking nutrient sources, and maximise the efficiency with which they can exploit their growth environment (1).

Forming a greater understanding of how multicellular bacterial behaviour can be controlled carries a number of potential benefits. Biofilms carry a socioeconomic cost, for example through pathogenic colonies surviving within medical equipment and infecting vulnerable patients (3), or through forming onto food-processing equipment and causing spoiling (4). As a result, there is great demand for methods by which biofilm formation can be discouraged or prevented, or biofilms made to form in such a way as to be vulnerable to standard sterilisation techniques.

While the genetic and metabolic drivers of multicellularity can be studied on the cellular level in the laboratory, the benefits, effects and overall emergent behaviour of biofilms can be effectively explored through computational modelling. Traditional spatial models are mathematically complex, but bacterial cells do operate as individuals that respond to one another's signals and their environment to determine their individual behaviour. As a result, they are a good fit for agent-based modelling.

Agent-based modelling consists, as the name suggests, as a group of individual "agents" occupying the experimental space. Agents' behaviour may be influenced by one another, but there is no central director – each agent possesses its own decision-making criteria and makes its decisions as an individual. Agent-based models typically also create an environment that the agents exist in and can alter through their behaviour, changes that may in turn affect the behaviour of the same or other agents(5).

Multiple software packages are available for agent-based modelling, but the one that was chosen was NetLogo. As a dedicated program for creating agent-based models, its bespoke scripting language contains a wide array of useful native functions, and is more intuitive to a novice programmer than general-purpose programming languages. It also includes a built-in display output as well as the ability to easily create a user interface by dragging-and-dropping interface elements.

Computing speed was a concern when choosing NetLogo, and early models remained very basic in order to slowly test its limits. It is, however, capable of interfacing with C and Java, and some experimenters have achieved large gains in performance by offloading particularly demanding routines to true, faster programming languages. Other experimenters have documented a range of ways in which native NetLogo performance can be improved through efficient code architecture(6).

Given that the models planned were not expected to be very computationally heavy, NetLogo was chosen as the most likely option with which interesting results could be obtained over the course of the project. Several models were thus created over the course of the project.

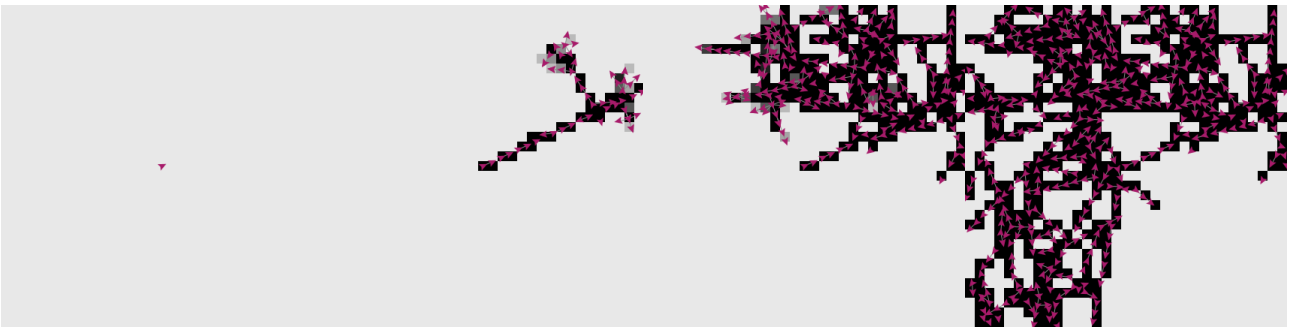
Results and Discussion

Basic Growth and Branching

The first model was intended to mimic a generalised view of bacterial colony formation, without reference to a particular species or growth mode. In particular, the model aims to address the effects of the colony's energy supply on the speed and shape of its growth – nutrients are spatially distributed and cells are only able to take up nutrients from their immediate vicinity. Later iterations of the model include nutrient diffusion mechanics, both outside and inside cells.

In this model, a single agent ("turtle" in NetLogo parlance, here representing a single cell) is created in the centre of the model area. Cells possess an energy quantity that determines their ability to survive and reproduce, and die if their energy needs are not met. Cells reproduce by replicating

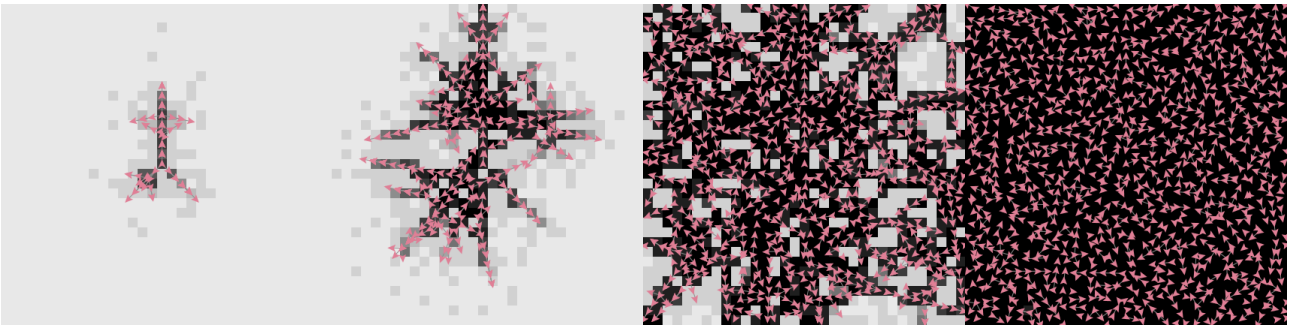
themselves and pushing the replicate forwards, mimicking cell division. Reproducing costs the cell as much energy as a turn of survival (in other words, the cell's energy upkeep requirement is doubled on a turn in which it reproduces) and a cell will reproduce whenever it can do so without immediately exhausting itself.



(Figure 1 – Screenshot series from the 2D basic growth model. Cells grow outwards in a branching pattern, consuming nutrients until no new branches can be formed.)

Under the default settings, each 1x1 grid square ("patch") can effectively support a single cell, producing one unit of energy per turn which is the same amount as the cell requires. As each cell is placed one grid length in front of its parent cell, branches usually cannot grow across one another – the first depletes the nutrients of the crossover area, inhibiting the growth of the other. This creates an effect similar to steric hindrance, whereby the colony is often blocked from exploring certain areas of the growth space, leading to an incomplete exploitation of the growth area and resources, as can be observed in figure 1.

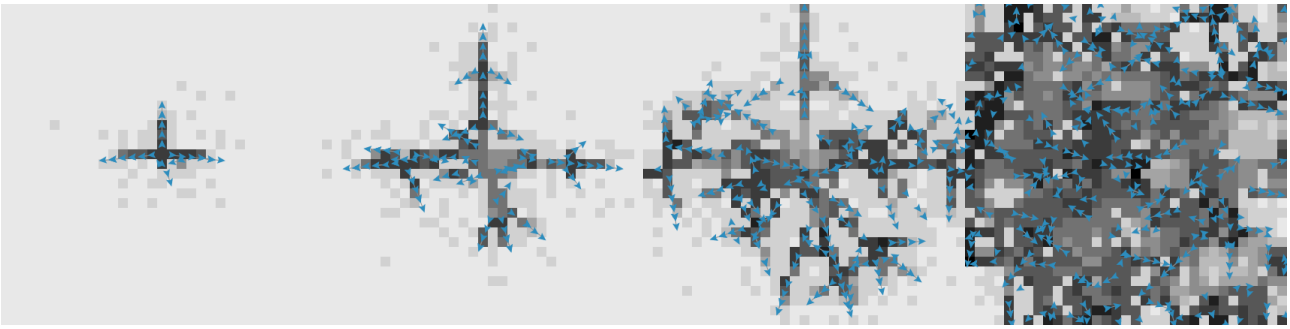
It is readily apparent why this occurs. In essence, a cell's reproduction has two outcomes – success, in which the new cell moves onto an unoccupied square, and is able to exploit a new source of energy without competition, or a failure, in which the new cell moves onto an already-occupied space, and the biofilm's overall access to resources is not increased. Since a patch supports one cell perpetually, but only a certain number of attempts at reproduction, it is not guaranteed that any one patch will be occupied by a cell before the colony as a whole runs out of energy for further growth.



(Figure 2 – Version of the model with nutrient diffusion. Cells branch out as before, but growth continues until every tile is occupied.)

This limitation is overcome when nutrient diffusion across the growth space is introduced to the model. More than once cell can survive on a grid space and a once cell can accumulate energy indefinitely if any surrounding patch stays empty, allowing it to continually reproduce until all adjacent squares are occupied. As seen in Figure 2, this ultimately, and invariably, results in complete growth over the surface with one cell per grid square, as this puts the energy flow exactly in equilibrium. However, the last few squares are only occupied following a long trial and error process.

The previous two models produce a colony that is largely static, as would typically be seen under a microscope. However, an interesting variation can be seen when each cell's energy requirements are greater than a single tile can provide. Static cells quickly exhaust resources on their own tile, and thereafter die. As long as a cell is able to take up resources faster than it uses them, however, it is still able to reproduce before dying.



(Figure 3 – In this model, each cell's energy requirement is greater than that a tile can provide. Cells split into a number of migratory colonies and a steady state is achieved.)

Figure 3 shows the resulting pattern. Rather than one large, connected colony, several strings of cells emerge, travelling around the model area constantly. These leave trails of nutrient-depleted areas which gradually replenish behind them. The colonies also occasionally gather enough energy to branch and send off a splinter colony in another direction. Colonies also occasionally meet, and compete for resources, usually resulting in the death of one of the colonies.

However, all of the aforementioned models display blind behaviour on behalf of the cells – their behaviour does not change based on their surroundings. As a result, they cannot adapt to changing conditions, and this represents an archaic view of microorganisms. These cells also do not display any form of cooperation, whether by inter-cell signalling or metabolic interaction– again, an archaic model, given current knowledge of microbial communities and cooperative behaviour(1).

Future work in this area might involve giving the cells the ability to sense and reproduce or move towards the highest concentration of nutrients in their immediate area. Another basic interaction would be the ability to sense when a tile is already fully exploited by existing cells and cannot support a further occupant. The ability for cells to communicate information and influence each others' behaviour through chemical signal diffusion would also be of interest.

The availability and diffusion of nutrients also could benefit from further development. While in the current model, nutrients are available uniformly across the growth area, it would be more useful to model a nutrient gradient by allowing nutrients to regenerate only at one or more edges of the growth area. In addition, nutrient diffusion could be slower across denser regions of cells, which would be useful for modelling the effects of biofilm porousness on internal nutrient availability, and therefore the population density within the biofilm.

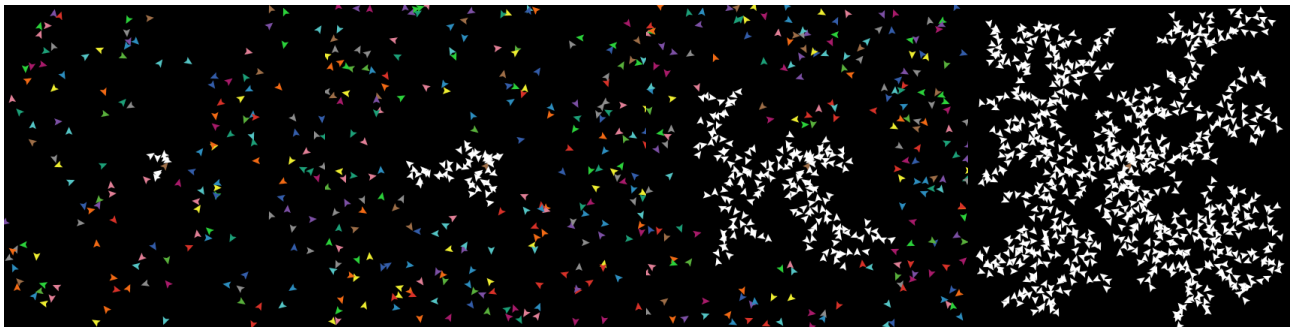
A further limitation with this type of model is that the growth and overall shape of the colony is only the result of cell division, and that all cells in the colony are descendants of those originally seeded at the beginning. As observed in Figure 1, this can produce a branched shape, but there are other, indirect methods of growth that can produce such a branching pattern, most notably a process called diffusion-limited aggregation.

Diffusion-Limited Aggregation

Diffusion-limited aggregation is a process in which dispersed particles following a random walk attach irreversibly to one another. Over time, particles accumulate into a branched, fractal shape. The mobile particles are far more likely to come into contact with the extended branches than with the core, since their random walk would have to successfully navigate past the existing branches to the centre for the latter to occur, which leads to the highly decentralised pattern **(Sander DLA)**.

This phenomenon is seen in many scientific contexts, and is commonly associated with crystal formation. However, it also has a role to play in biofilm formation. Cells in a biofilm excrete extracellular polysaccharide, a sticky matrix that allows other, free-floating cells to attach to the biofilm. In this way, biofilms may grow by aggregation as well as by cell division. This process was also illustrated using agent-based modelling in NetLogo.

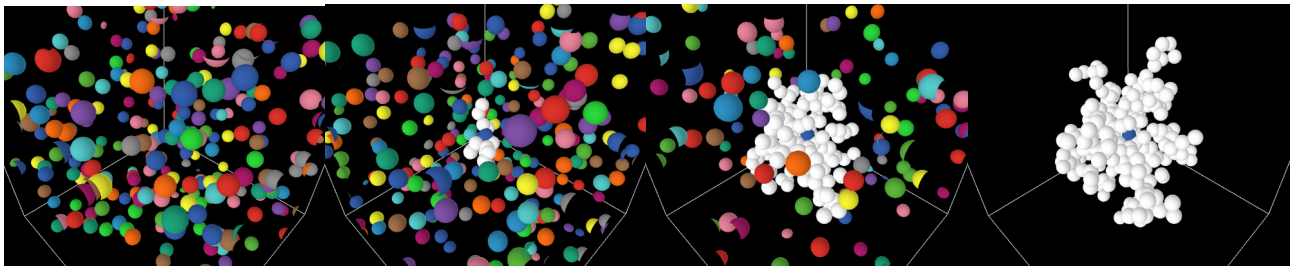
In the diffusion-limited growth model, a single, stationary particle is placed in the centre of the model area, and a large number of other, mobile particles are placed randomly. The mobile particles follow a random walk, and permanently turn stationary when they touch an existing stationary particle. The radius within which particles can be considered "touching" is changeable. The number of starting particles is customisable, and model can also be set to spawn in new mobile particles at random locations continuously in order to simulate new agents entering the simulation space.



(Figure 4 – Diffusion-limited growth model, in which randomly-moving agents are gradually caught by the central amalgamation and form a stationary, branched crystalline aggregate)

The result of this is visible in figure 4 – a branched pattern extending from the centre in all directions. This is the kind of pattern that would be expected to be seen if a biofilm were formed solely by cells aggregating, but not dividing (or at least, aggregating much faster than the reproduction cycle of the cells). This provides an interesting secondary mode of growth for a biofilm for potential future work.

The original model was implemented in 2D, but was later translated to full 3D. In the 3D model, the model space remains homogenous, i.e. particles are equally free to travel in all directions and area – there is no interface between two media, nor is gravity simulated, though both of these would make interesting targets for future work.



(Figure 5 – Difusion-limited growth model in NetLogo3D, a closely-related fork of NetLogo. Although aggregation is slower, the model is fundamentally the same as the 2D iteration.)

As shown by figure 5, the end result is very similar to the 2D model, although the overall aggregation is slower given that collisions are simply less likely in a 3D space than a 2D space of the same xy dimensions. The overall number of particles was also reduced, as higher numbers interfered with the visibility of the aggregate in a 3D environment. The transition to NetLogo3D necessitated the use of a few additional commands such as pitch and roll, to account for three-dimensional movement, but the overall functionality of NetLogo3D is the same as NetLogo, and it receives the same updates as the core NetLogo.

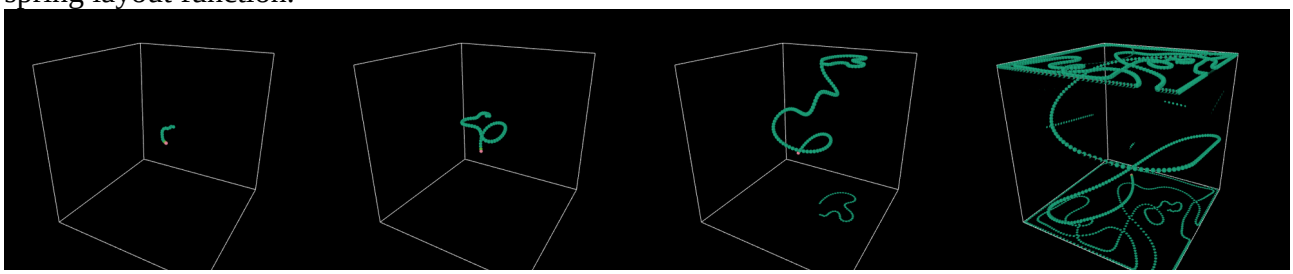
Aggregation represents one means of handling collisions between cells, but there are other possible interactions. Most obviously, being physical objects, cells should exclude one another from their own space, but NetLogo does not include true functionality for this by default. Other interactions could include being linked together in a chain, as some bacteria do, such as Streptococcus (1). To further expand upon the possibilities for future models, attempts were made to repurpose existing NetLogo functions for the purpose of modelling physical interactions between agents.

3D Chain Physics

During the switch from NetLogo 2D to 3D, attention was focused on making cells interact with one another and their environment in a physically accurate way. While NetLogo can and has been used for particle physics models, it lacks built-in support for implementing a physics engine. As a result, it was quickly determined that any sort of complex physics model would be beyond the scope of the project.

However, some experimentation was made with NetLogo's "Layout Spring" function. With this command, each "turtle" in the selected agentset moves in such a way as to maximise their distance from every other turtle in the agentset. However, they also experience a spring-based attraction to other turtles to which they are "linked" (a NetLogo function), proportional to their distance of separation in accordance with Hooke's law.

Multiple models were made to test this function. The first of these was a simple model in which a string of connected cells are subjected to NetLogo's spring physics. In this model, a new cell is created at the end of the existing chain each tick (though the creation of new cells can be switched off at any time). The entire set of cells is moved a number of times every tick according to the spring layout function.



(Figure 6 – 3D spring layout model testing. A chain of linked cells grows over time, and the chain uncoils as far as possible. However, it tends to rapidly become stuck against the area boundary.)

As can be seen in Figure 6, the result is that the string of cells extends to form a straight line where possible, curving once the string approaches the edges of the simulation area. This is broadly as

expected – a straight line configuration puts as much distance as possible between each element of the chain and every other element, whilst maintaining cohesion between the line of linked cells.

The constant recalculation of each element's position, however, is computationally intensive – with ten cycles of the spring function per tick, the simulation is orders of magnitude slower than the previous models shown. In light of this, further models were developed using this routine to test whether its use on a large scale could be feasible.

The second model developed was created on similar lines, but with an important difference. Rather than have each agent represent a cell, cells are instead represented by strings of between three and five agents, hereafter termed "sections". Chains constantly grow, and splinter down the middle once they exceed a length of five sections, mimicking cell division as typically seen in rod-shaped bacteria.

This model required some more dynamic handling of turtle variables. Cell sections in this model possess two variables, "location" and "ID". Both variables are dynamically updated across entire chains using a recursive loop. "Location" records how many links a section is from the end of the chain, with the variable updating every time the chain grows. Links break when both of the sections they connect have a Location of 2 or greater, so a chain of 6 sections or more will immediately shear in the centre. Shearing also causes the locations of the daughter cells to update again.

"ID" records which cell a section is part of at any one time. This value is common to all sections of a cell, and each cell has a unique ID value. Upon shearing, one of the two daughter cells updates its ID to the lowest unclaimed value and propagates this value across all of its sections. While this variable plays no part in the existing model, the ability to call all of the sections in a specific cell would have been useful for targeting functions if the model were developed further to incorporate cell-cell or cell-environment interactions.

Agents in this model have a more refined mode of interaction. Cell sections interact only with the other sections of their own cell, and with other sections that are within a distance of 1 – roughly the size of the sections themselves, so that cells repel one another until they are no longer physically intersecting, but no further. The limitation of interactions to a certain distance does substantially improve the speed of the simulation over earlier iterations that unintentionally caused all sections to interact with all other sections regardless of distance, an issue that also caused cells to constantly drift away from one another until they simply lined the edges of the simulation space.

(Figure 7 – A single starting cell extends and splinters into two, and the daughter cells are unable to overlap one another. This process continues to create an amorphous mass of cells over time.)

As Figure 7 shows, this model results in a close jumble of cells, as there are no forces to apply motion to the cells beyond the requirement that they take as straight a shape as possible and avoid intersecting one another. This does not correspond well to real-life biofilm growth, but this most likely can be attributed to the lack of any gravity. This could be addressed by implementing a

downwards movement to the cells and a solid surface at the bottom, allowing cells to gather on the surface and accumulate into a more usual configuration.

Further work could expand upon this, but the decision was made not to continue with this angle due to worsening performance issues. NetLogo proved unable to handle a computationally useful number of cells without experiencing severe slowdown. Further exploration of this idea would most likely require offloading the physics-based computation to a faster language, such as Java or C, and would have taken too much time to develop to fall within the scope of the project.

Hyphal Growth Model

Following this exploration of the possibilities of NetLogo in modelling microbial communities, this model was specifically designed to mimic the hyphal growth modes seen in Streptomyces. In this form of growth, the organism does not form discrete cells as it grows, but instead extends to form a growing "tunnel" or hypha(1,2). Cell wall synthesis occurs at the polarisome, an organelle found at the growing tip of the hypha, and the tip extends forwards with the polarisome as the cell wall assembles behind it(7).

Tip extension is a polar behaviour – the polarisome is present at only one end of the hypha and so extension occurs only at this end. Furthermore, the polarisome grows over time and eventually splinters, leaving behind a second, much smaller polarisome within the body of the hypha. This polarisome remains stationary for some time as it accumulates material, before extending out of the hyphal wall to form a branch perpendicular to the original hypha(7).

In this way, a single streptomycete can grow into a connected network of filaments, termed a mycelium after the equivalent structure seen in fungi. Streptomyces' hyphal growth mode is extremely unusual among bacteria, whose complex multicellular structure might offer interesting opportunities for synthetic biologists that could otherwise only be attained by working with fungi(1).

Experiences from the previous model were drawn upon in the creation of the hyphal growth model, but the model was constucted largely *de novo*. The model consists of two types of "turtles" – cells and tips. The model begins with a single tip, which reproduces by creating a daughter cell and pushing it forwards. After reproducing once, the tip then changes type, becoming a functionally inert "cell", while the daughter cell remains a tip until it then reproduces in kind. In this way, there is always a single "tip" at the end of the hypha.

Tips track how long ago they last created a branch, and how long ago their own branch was formed. These values are parametrised so that the user can use three sliders to easily control how frequently a hypha branches, how long it takes that branch to begin growing and how long it takes the new branch to start forming its own branches.

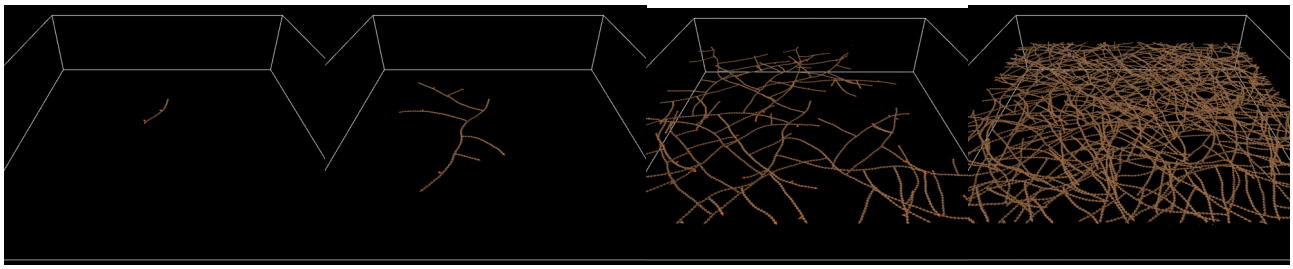
Hyphae in this model also do not necessarily grow directly forwards: the angle at which each new "turtle" forms is normally distributed with a mean of zero and a standard deviation governed by a slider in the UI. Similarly, the angle at which branches form is also normally distributed and both the mean and standard deviation can be modified, although the branch always extends from the outside wall of a curved hypha.

Rather than simulate nutrient use and diffusion directly in this model, competition for growth space and nutrients is represented by a limit on how many cells a growing tip can be surrounded by before it "dies". The growing tip checks the number of other cells in a given radius each time it grows, and

if the number is over the maximum number of neighbours, the tip becomes a regular, quiescent cell. The radius and number of cells can be altered by the user with a slider.

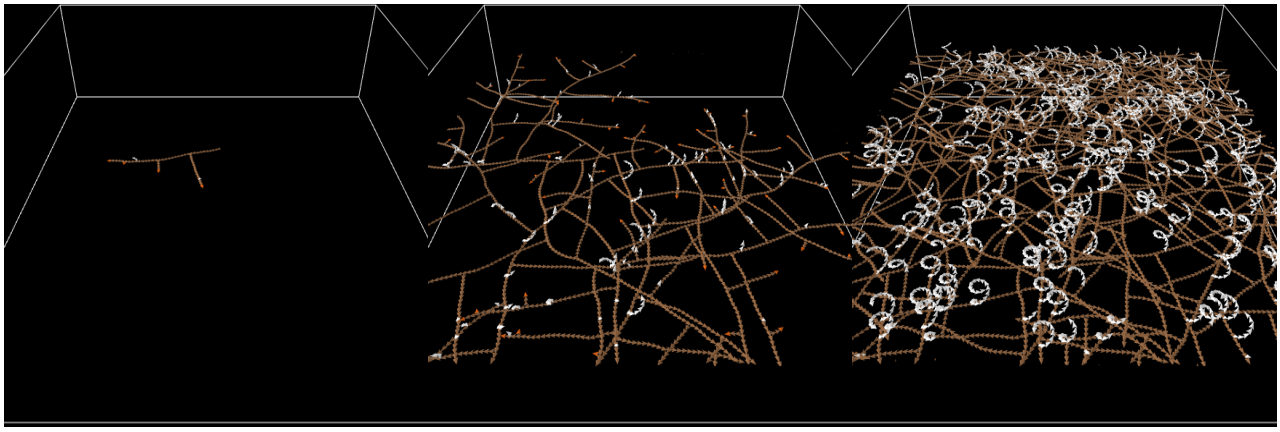
The radius of the inhibitory effect is potentially analogous to the rate at which nutrients can diffuse through the growth medium; the number of cells that can be supported in that radius is analogous to the abundance of nutrients. While the previously-seen nutrient diffusion system could be implemented in the future, the model would in that case have to implement and explore a complex system of nutrient distribution across hyphae, as unlike most bacterial colonies, the cells are not fully separated and cell contents can diffuse or be actively transported across the entire hypha.

The model was developed in NetLogo 3D to allow for vertical growth, as Streptomyces has two modes of verticality, vegetative, or "regular" hyphae may grow downwards rather than being confined solely to the horizontal plane, so long as the growth surface is penetrable by the hyphae. Aerial hyphae, meanwhile, grow upwards out of the vegetative hyphae and form spiral structures in the space above the colony. Aerial hyphal formation is part of the mycelial reproductive cycle, as they then divide into chains of spores that detach from the hyphae to seed new colonies (8,9).



(Figure 8 – Basic model of hyphal growth. The initial hypha branches at regular intervals, covering the growth area, and then growing denser over time. Branches displace one another vertically.)

The full 3D capabilities of the vegetative hyphae have yet to be implemented in the model – as seen in Figure 8, vegetative hyphae grow almost exclusively in two dimensions, although overlapping hyphae do push one another vertically rather than physically growing through one another. Downwards growth is currently limited to branches travelling diagonally in a straight line, though a more accurate interpretation could be implemented if guided by referential Streptomyces images obtained in the laboratory.

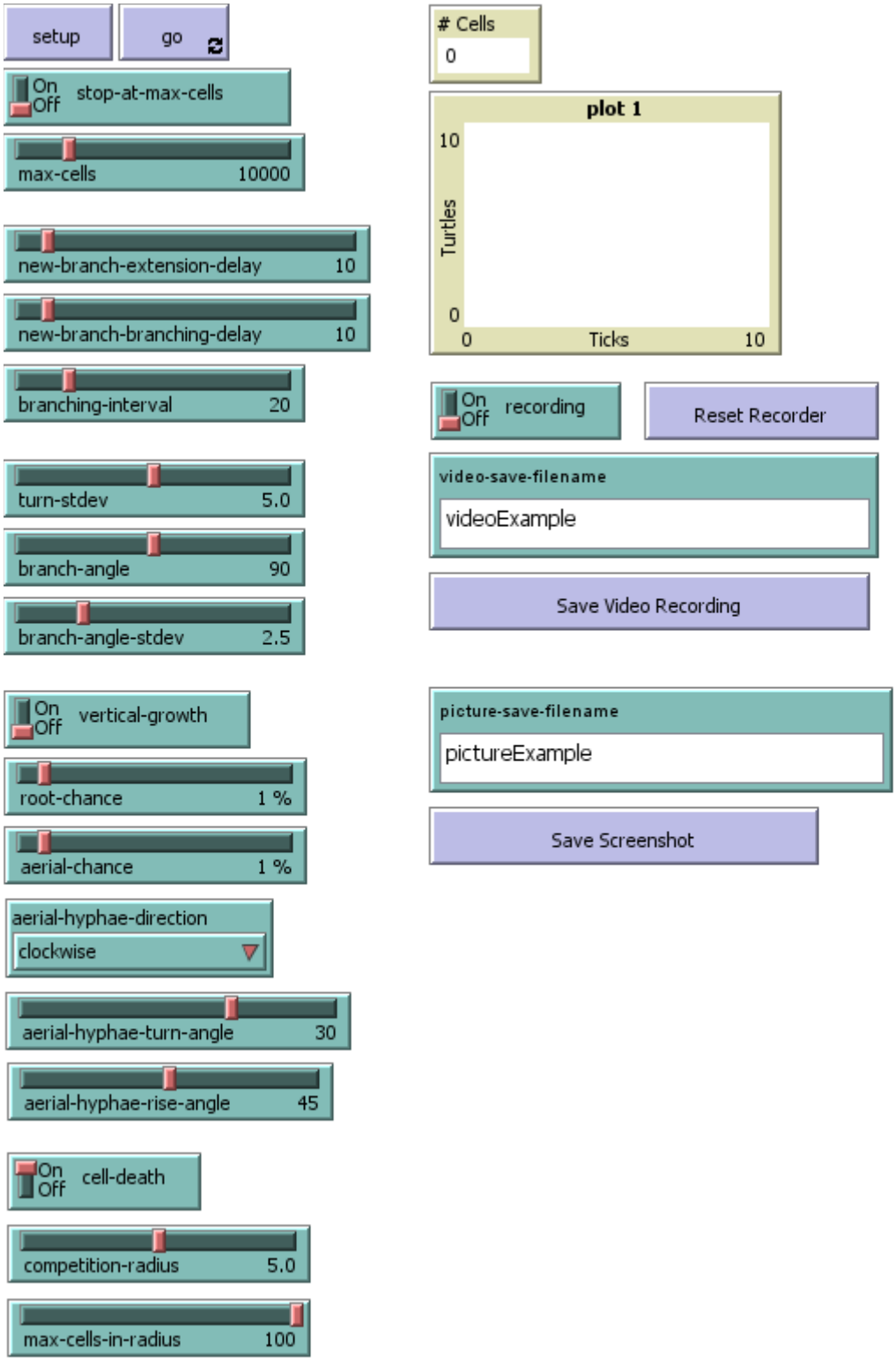


(Figure 9 – Variant of the hyphal growth model with aerial hyphae implemented. Aerial hyphae grow slowly over time, and grow in an upward spiral whose shape can be altered.)

As seen in figure 9, aerial hyphae are spawned at random along the growing tip of the hypha. Once present in the model, an aerial hypha grows steadily upwards in spiral, the angle of which can be set

by the user. However, this s not particularly true to real aerial hyphae – while they do form spiral structures, they necessarily grow by cannibalising the vegetative hyphae beneath them, and do not emerge directly from the growing tip, unlike other branches (8,9). In the model, they also do not currently form spores.

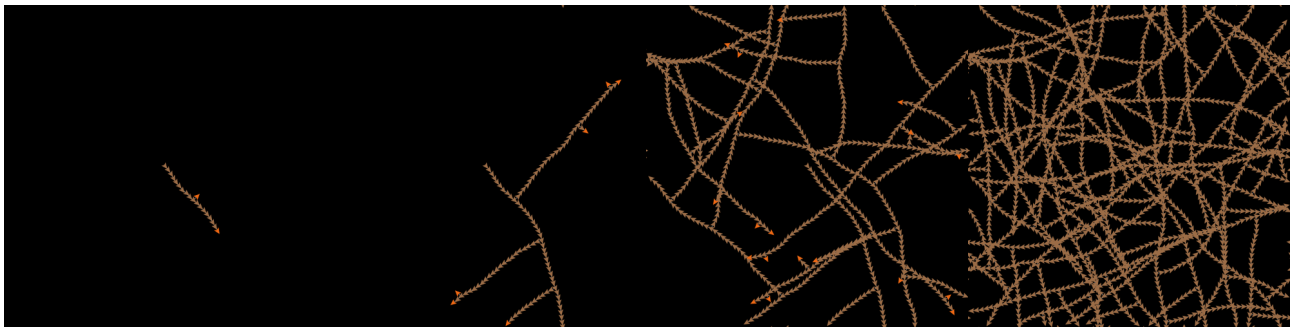
Both these issues highlight the singular importance of having experimental data in which to ground any model that is intended to be useful in academic research. More accurate modelling the growth of Streptomyces mycelia would necessitate first-hand timelapse photography, as the video evidence available online is limited. Nonetheless, the existing model does mimic the overall structure of Streptomyces mycelia to a remarkable degree.



(Figure 10 – User interface for the 3D hyphal growth model, developed using NetLogo's drag-and-drop system. This user interface was developed to be used in a demonstration and allows the used to alter a wide range of variables that are used in the model.)

Figure 10 shows the user interface that was created for the mycelial model. The model has a number of sliders and switches that can be used to manipulate the angles of growth, branching frequency, verticality and level of inhibition. It also has buttons with which a run of the model can be recorded, or with which screenshots can be taken. This makes it very user-friendly to experiment with the different parameters and record the results.

In this form, the model was used to run a short demonstration for undergraduate students and proved to be an effective and engaging educational tool.



(Figure 11 – Variant of the hyphal growth model adaptedfor use in standard NetLogo. There are no aerial hyphae in this version of the model, and vertical displacement of branches does not occur.)

A 2D model was also developed by eliminating the 3D growth elements of the model, which include the aerial hyphae and the physical vertical displacement of overlapping branches. This model was found to be much faster and somewhat more user-friendly than the 3D model, although it leaves the experimenter with fewer options and does not account for any vertical movement.

Methods

All models were created and run in NetLogo 6.0.3 or NetLogo3D 6.0.3. Model files, as well as figures and a copy of this document can be found at: https://github.com/dTaylor-dcsWarwick/SynBio_EP1_Appendix.git

Conclusions

NetLogo proved to be a versatile and useful tool for agent-based modelling, able to support a wide range of different models created for different purposes and featuring a variety of different mechanics. Despite this versatility, it also proved to be straightforward to learn, with a bespoke scripting language that is well-documented and easily comprehended by a novice programmer.

The pre-existing visual output and the ability to create a user interface though simple drag-and-drop placement allowed functional programs to be created that could be easily and intuitively interacted with in order to explore model behaviour, and many output functions and formats remain as-yet untested by the experimenter.

However, more intense physics-based models, and other models with a lot of regular interactions between agents, did suffer from performance issues with relatively few agents present. Some of these issues can be avoided with intelligent, informed script design to avoid calling unnecessary checks and interactions, but NetLogo is nontheless a slow option compared to “real” programming languages. As a result, computationally intense models should look to incorporate bottleneck steps into C or Java sub-scripts, if NetLogo is to be used at all in this capacity.

Nonetheless, for modelling where massive size or exacting physical simulation is not a pressing concern, NetLogo performance is not an issue and the ease of creation more than makes up for lost time in computation. For very low-performance modelling, the ability to run NetLogo scripts via web browser allows users to experience and experiment without even having to download the software at all.

NetLogo is an ideal tool for teaching biological modelling to non- or novice programmers, and can certainly find ground as a versatile research tool as well as a teaching aid, available both to download and as a browser application. “Big data” modellers may prefer a more high-performance option, but NetLogo still offers a useful prototyping platform, or as a user-friendly front-end to a program otherwise written in a higher-performance language.

Bibliography

1. Claessen D, Rozen DE, Kuipers OP, Søgaard-Andersen L, Van Wezel GP. Bacterial solutions to multicellularity: A tale of biofilms, filaments and fruiting bodies. Nat Rev Microbiol [Internet]. 2014;12(2):115–24. Available from: http://dx.doi.org/10.1038/nrmicro3178
2. Flärdh K, Buttner MJ. Streptomyces morphogenetics: Dissecting differentiation in a filamentous bacterium. Nat Rev Microbiol. 2009;7(1):36–49.
3. Francolini I, Donelli G. Prevention and control of biofilm-based medical-device-related infections. FEMS Immunol Med Microbiol. 2010;59(3):227–38.
4. Kumar CG, Anand SK. Significance of microbial biofilms in food industry: a review. Int J Food Microbiol [Internet]. 1998;42(1–2):9–27. Available from: http://www.sciencedirect.com/science/article/pii/S0168160598000609
5. MacAl CM, North MJ. Tutorial on agent-based modelling and simulation. J Simul. 2010;4(3):151–62.
6. Railsback S, Ayllón D, Berger U, Grimm V, Lytinen S, Sheppard C, Thiele J. Improving Execution Speed of Models Implemented in NetLogo. 2016:128:2 [Internet]. 2017;20(1). Available from: http://jasss.soc.surrey.ac.uk/20/1/3.html
7. Flärdh K, Richards DM, Hempel AM, Howard M, Buttner MJ. Regulation of apical growth and hyphal branching in Streptomyces. Curr Opin Microbiol. 2012;15(6):737–43.
8. Manteca A, Fernandez M, Sanchez J. A death round affecting a young compartmentalized mycelium precedes aerial mycelium dismantling in confluent surface cultures of Streptomyces antibioticus. Microbiology. 2005;151(11):3689–97.
9. Manteca A, Claessen D, Lopez-Iglesias C, Sanchez J. Aerial hyphae in surface cultures of Streptomyces lividans and Streptomyces coelicolor originate from viable segments surviving an early programmed cell death event. FEMS Microbiol Lett. 2007;274(1):118–25.
10. Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.