Towards the Existence of Two Distinct Types of Emergent Behavior Relevant to Biological Systems

Citation

Permanent link
https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37364745

Terms of Use
This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story
The Harvard community has made this article openly available. Please share how this access benefits you. Submit a story.

Accessibility
Towards the Existence of Two Distinct Types of Emergent Behavior Relevant to Biological Systems

A thesis presented by
Nicholas Wong

to
the Faculty of the
Harvard John A. Paulson School of Engineering and Applied Sciences
in partial fulfillment of the requirements for
the Bachelor of Arts degree in
Biomedical Engineering and Computer Science

Faculty Advisor: Prof. Stratos Idreos

Harvard University
Cambridge, MA
April 3, 2020
Honor Code

In submitting this thesis to the Harvard John A. Paulson School of Engineering and Applied Sciences in partial fulfillment of the requirements for the joint degree of Bachelor of Arts, I affirm my awareness of the standards of the Harvard College Honor Code.

Nicholas F. Wong
Acknowledgments

Professor Idreos, thank you for taking me on as your advisee, even though I had slept through a handful of your 9am lectures. I am extremely grateful for your class, and I credit your analysis of the fundamental components of data structures with formally getting me started in examining complex systems.

Professor Parker, your dedication to your students inspires me, and I am grateful to you for forcing me to become a better hypothesis-writer and scientific thinker. Without your instruction, I would not have been able to perform the laboratory experiments in this thesis.

Professor Mickens, thank you for reminding me that scientific research and thought is an evolving entity characterized by discussion and clarity. I am incredibly grateful for your systems security course, which helped me mature as a researcher.

Dr. Hancock – I don’t know what I would have done without your help. From allowing me to use the Active Learning Labs to helping me procure materials and supplies to providing me with data when I felt COVID-19 had removed any chance at quantitative analysis. Even more importantly, thank you for your kindness and discussions.

Professor Moyer, it has been a long 4 years, and you’ve been advising me and listening to my concerns the whole way. Thank you so much for being my introduction to Biomedical Engineering as well as for being my primary source of guidance as I learned to become a better scientist and student.

To all of the staff of the Active Learning Labs, thank you for keeping an incredible part of Harvard alive and available to all undergraduates.
Mary, thank you for patiently listening to all of my complaining and for helping me revise this thesis for hours on end. Without your support and help, I’m positive I would not have made it through writing this document.

Daniel, it’s been 4 years, from LS50 to ES123, and I’m 100% sure I would not be where I am now without your unending support, guidance, and humor. Thank you so incredibly much for all of the late night problem set sessions, the rants, and the advice.

Jungyeon, thank you for pointing out the incredibly obvious fact that I should make use of both human and bacterial cells in writing this thesis, and thank you for your support through this process.

To my roommates, Audrey, Bagatur, Maddie, and Sam, I have no idea how I would have made it through this process and college without you. This thesis would not have been possible without your unending support and optimism, your discussions, and the snacks when I forgot to eat. I’m incredibly grateful to each of you for being a home I could always return to, whether from lab, class, or lab (I was always in lab).
List of Contributions

In light of the COVID-19 disruption to the academic year and laboratory work, Dr. Melissa Hancock of the Harvard Active Learning Labs graciously shared her previous work with *Bacillus Subtilis* with me for the purposes of writing this thesis. All images of *Bacillus Subtilis* used herein are with permission from Dr. Hancock and are the result of her hard work.

Additionally, all laboratory work was performed in the biology laboratories of the Harvard Active Learning Labs with the permission of Dr. Hancock.
Abstract

Many organisms in the world exhibit strange behaviors when found in groups; fish, birds, and ants all act differently when in groups than when separate. In this thesis, I define these types of behaviors as emergent behaviors and subdivide them into two types: primary and secondary emergence. The former indicates emergent behavior that arises due to constant behaviors or properties in the individuals. The latter refers to emergent behavior which arises due to behaviors which evolve as reactions to the environment. To support the relevance of this subdivision, I performed experiments with Human Dermal Fibroblasts, *Bacillus Subtilis*, and computer simulations. The images collected were quantified via image analysis, and statistically compared in order to make claims about whether the observed behaviors were primary or secondary emergence. The *B. Subtilis* and all four categories of simulation resulted in random walks. Comparisons between categories of simulation and *B. Subtilis* revealed that certain results were further from random walks than expected, namely “Threshold – Dense,” “Towards Neighbors,” and *B. Subtilis*. Due to the values of the r-squared coefficients obtained, I argue that the “Threshold – Dense,” “Towards Neighbors,” represent secondary and primary emergence, respectively. I claim that these deviations are evidence of emergent behaviors being detected by statistical analysis, a key step towards using emergent behaviors to predict and control biology.
# Table of Contents

**HONOR CODE** ......................................................................................................................................................... 2

**ACKNOWLEDGMENTS** ................................................................................................................................................. 3

**LIST OF CONTRIBUTIONS** ............................................................................................................................................. 5

**ABSTRACT** ................................................................................................................................................................. 6

**INTRODUCTION** ........................................................................................................................................................... 10

**HISTORICAL BACKGROUND** ......................................................................................................................................... 10

  - Philosophy ........................................................................................................................................................................ 10
  - Mathematics and Computation ........................................................................................................................................... 11
  - Biology ................................................................................................................................................................................ 12

**EMERGENT BEHAVIOR** ..................................................................................................................................................... 14

**CELLULAR AUTOMATA** ..................................................................................................................................................... 15

**RANDOM WALKS** ............................................................................................................................................................. 16

**HUMAN DERMAL FIBROBLASTS AND BACILLUS SUBTILIS** ............................................................................................. 17

**CONTRIBUTION AND HYPOTHESIS** ............................................................................................................................. 18

**MATERIALS AND METHODS** ........................................................................................................................................... 21

**HUMAN DERMAL FIBROBLAST EXPERIMENTS** ........................................................................................................... 21

  - Cell Culture ........................................................................................................................................................................ 21
  - High, Medium, and Low Density Setup ............................................................................................................................... 21

  - Plate Setup .......................................................................................................................................................................... 21
  - Imaging Strategy .................................................................................................................................................................... 22
  - Control .................................................................................................................................................................................. 22

**BACILLUS SUBTILIS EXPERIMENTS** .............................................................................................................................. 23

  - Cell Culture ........................................................................................................................................................................ 23
  - High, Medium, and Low Density Setup ............................................................................................................................... 23
Introduction

In everyday life, when one observes a flock of birds or a school of fish, it is easy to reflect that these phenomena are beautiful and intriguing – they elude simple explanations as to their exact origins. Are the birds controlled by a leader? Do the fish communicate and vote on which direction they should swim next?

In human biology, these phenomena take on less obvious forms, from the metastasis of cancerous tumors [2] to the migration of tissues to close a wound [3]. In fact, it is not clear whether the ways in which birds come together to form a flock moving as one unit is in any way related to the way in which human fibroblasts come together to form a tissue that can heal a wound.

These phenomena are known as “emergent behavior,” a term whose use has been observed in numerous disciplines and with almost equally numerous definitions. Thus, the following sections aim to provide historical context and coherent terminology.

Historical Background

Philosophy

In the field of philosophy, it is common to subdivide emergence into categories based on the properties of individuals involved and whether the resulting emergence is deducible from the properties of the individuals [4-7]. According to a canonical argument by Chalmers, emergent behaviors fall into two categories: strong and weak [5]. Strong emergence is when a group’s properties cannot be logically deduced from those of the individuals within it [5]. In contrast, weak emergence is an “unexpected” consequence of the rules and principles which govern the
individuals [5]. Fromm argues this duality does not sufficiently explain various manifestations of emergence and argues that emergence should be categorized based on the types of causality and feedback mechanisms involved [6]. Throughout the history of emergence, some have argued that that the “strong-weak” duality is not complex enough to capture the significant consequences of emergent behavior [4]. This results in varying philosophical subdivisions of emergence, but as Wilson argues, these various definitions actually fit within the strong versus weak emergence framework, and thus the other distinctions are merely “superficial” in nature [7].

Most relevant to this paper, Bedau argues that while strong and weak emergence both exist, only weak emergence is relevant to the sciences [8]. Bedau finds that strong emergence cannot be studied in the real world, stating, “strong emergence is logically possible, [but] it is uncomfortably like magic” since the emergent property of the group cannot be explained by the properties of the individuals [8]. Instead, he focuses on Chalmers’ definition of weak emergence, adding that the relation between the emergent property of the group and the properties of its individuals can only be discovered through simulation [8]. Moreover, Bedau finds that weak emergence is not simply an abstraction human minds make, but rather an actual scientific phenomenon [9].

In summary, Chalmers distinguishes between strong and weak emergence, and Bedau argues that only weak emergence is relevant to the sciences [5, 8].

Mathematics and Computation

As alluded to in the philosophical background, the concept of emergence has been applied to many fields, including computer simulation. Math has often been used as the tool for modeling emergent behaviors (or more accurately, the individuals and their interactions which
result in emergent behaviors) [10]. The mathematical and computational ties to emergent behavior are often phrased in terms of either modeling a system or in terms of using emergent behavior to compute an outcome. An example of the former is the equation \[ V_i \frac{d S^i}{dt} = U \left\{ \sum_j \omega_{ij} S^j - S^i \right\} + h^i(\epsilon) \], which is used by Millonas to model the flow of swarming ants through a lattice network [11]. An example of the latter is Forrest and Miller’s discussion of using the principles of emergent behavior in machine learning classifier systems to determine whether a given object is or is not food [12].

Both mathematicians and philosophers working from first principles arrive at the conclusion that both strong emergence and weak emergence must exist, and that the existence of strong emergence is only hindered by the practical concerns of observation and measurement [5, 13]. Bar-Yam presents an argument for the scientific-relevance of strong emergence based in micro- and macrostates of systems [13]; however, this thesis will focus on weak emergence, which benefits from a clearer applicability to real-world systems.

Biology

As opposed to the theoretical approach to emergent behavior from philosophy and mathematics, biology studies emergent behaviors concretely and arrives at apparent contradictions in the ways in which they occur [1, 2, 11, 14-19].
Bacteria have often been used as a starting point for the study of emergent behaviors in living systems, as in the example of Chen et al. analyzing *B. subtilis* in order to demonstrate correlations in the movement of the bacteria [14]. Chen et al. shows experimentally how bacteria correlate direction of movement regardless of colony densities, and they hypothesize that this may provide an evolutionary advantage since the group experiences the ability to detect nutrients or predators as an emergent property [14]. A key finding of the work of Chen et al. is that the *B. subtilis* collective response appears to occur in a scale-invariant way, independent of colony density [14].

In contrast to the above scale-invariant findings, some work with mammalian cells has reported scale-dependent emergence. Sun et al. explores the emergence of chemosensing – the ability to detect and respond to the presence of specific molecules in the environment – in mammalian cells and similarly takes a correlation approach to demonstrating that the cells are acting as a group when exposed to chemical stimuli [1]. A practical consideration pointed out is that with complex mammalian cells, it is difficult if not impossible to completely isolate each of the routes of intercellular communication.

![Figure 1](image.png)

*Figure 1. A figure from Sun et al. A shows the cross-correlations of relative fluorescent indexes in fibroblasts averaged across nearest neighbors, and B shows the same cross-correlations in dense colonies (1,060 cells/mm²) for different stimulating ATP concentrations [1]*
communication, which makes experimental validation of the cause of the emergent behavior difficult [1]. Sun et al. uses fibroblasts and their reaction to ATP to demonstrate that lower densities of fibroblasts had almost no detectable response while increasing density resulted in increasingly pronounced responses, as summarized in Figure 1 [1].

In the bacterial example above, the findings are that the emergent behavior is independent of the density of the colony [14], whereas the emergence of the target behavior in fibroblasts appears to be dependent on the density of the colony [1]. It is from this difference that this thesis draws inspiration.

Emergent Behavior

Emergent behavior is canonically separated into strong and weak subdivisions [5], and while there are arguments for the relevance of strong emergence [13], the remainder of this thesis will deal solely with weak emergence.

Papers regarding emergent behavior often focus on different aspects of emergent behavior even within the category of weak emergence [15, 20-22]. Many definitions place an emphasis on the ability of individuals in a group to achieve a unified action or decision without the existence of a central authority [15, 21]. Others focus on how individuals are simple and how individuals can only communicate with nearby individuals [22]. Still other definitions emphasize that individuals coordinate their behaviors [20]. Despite their different focuses, each of these definitions is still within the category of weak emergence provided by Chalmers: weak emergence is behavior which arises as an unexpected result of the rules and properties governing the individuals in a group [5].
In order to preserve the flexibility of meaning in Chalmers’ argument, this thesis implicitly refers to Chalmers’ definition of weak emergence in all mentions of emergence and emergent behavior below. However, in order to explain some of the choices made in the design of the simulation aspect of this thesis, some aspects of the above, narrower definitions are used.

Cellular Automata

As argued by Bedau, weak emergence must be demonstrated via computer simulation [8]. This thesis uses cellular automata inspired by Conway’s Game of Life [23], which is a well-known cellular automaton with extremely simple rules.

Cellular automata are frequently used to investigate the emergence of behaviors in groups of individuals [24, 25]. These individuals are referred to as “agents” in the terminology of cellular automata. Cellular automata can take on a form known as “agent-based simulation,” where the creator of the simulation creates agents which follow creator-defined rules in order to interact at each time step of the simulation [24]. Agent-based simulation has the following benefits: it uses simple agents as the individuals in the group, it provides clear insight into the limited number of ways in which these agents may interact, and it has been shown to still exhibit complex emergent behaviors [24]. Cellular automata additionally have the ability to be either synchronous or asynchronous, which determines whether each individual performs its action simultaneously (synchronous) or not (asynchronous) [24].

One of the key limitations that arises when using cellular automata to approximate biology is that the individuals within the cellular automaton are restricted to discrete states (e.g. alive or dead). Thus, it is difficult to approximate continuous real-valued quantities such as angle of orientation. However, it should be noted that even Conway’s Game of Life can create
arbitrarily complex phenomena and that it is possible to build a Turing-complete machine using only cellular automata [25]. The specifics of these proofs are beyond the scope and purpose of this thesis; however, they are worth noting for the sake of arguing that cellular automata are sufficient to model the biology herein.

Random Walks

![Random Walk Diagram](image)

**Figure 2.** Two ways of depicting two dimensional random walks are shown. A shows an agent (dark circle) at time $t_0$ which can move to position 1 or -1 at time $t_1$ with probability 0.5 of each move. B shows a Markov depiction of the same process, where the probability of transitioning to either adjacent position at each time step is 0.5.

Random walk models for motion are common mathematical models used to describe biology [26-29]. As depicted in Figure 2, the simplest form of a random walk is an agent which takes a step in any of two available directions with equal probability at each time step. For example, in an infinite single dimensional random walk, the agent takes a step either to the left or to the right with probability $\frac{1}{2}$ at each time step. However, the applicability and complexity of random walks can be greatly increased by introducing the concepts of persistence (continuing in the same direction as prior motion), bias (preferring some subset of directions to others), and waiting (allowing no step to be taken at a given time step) [28]. The diffusion and Fokker-Planck equations are readily derivable from even simple random walk models [28], demonstrating that random walks are sufficient for describing cellular motion. Dickinson and Codling both employ
two dimensional random walk models to explore and predict cell motion [16, 28]. As a result, random walks are selected as the primary basis model for the simulation work in this thesis.

Human Dermal Fibroblasts and *Bacillus Subtilis*

The laboratory component of this thesis consists of two parts: the first is the culture of Human Dermal Fibroblasts (HDFs) which are depicted in Figure 3 panel A, and the second is the culture of *Bacillus Subtilis* (*B. Subtilis*), depicted in Figure 3 panel B.

![Figure 3. A depicts an image at 20x magnification of Human Dermal Fibroblasts cultured in the Harvard Active Learning Labs. B is an image taken by Dr. Hancock of B. Subtilis colonies expressing green fluorescent protein.](image)

HDFs are used due to their well-studied role in the process of wound healing [30]. Fibroblasts play a major role in the proliferation phase of wound healing, which is characterized by rapid cell growth and movement, as well as remodeling of the extracellular matrix [30]. Additionally, HDFs are relatively easy to culture in the laboratory and have been studied frequently in the context of tissue migration and cell motility [30].

*B. Subtilis* is used due to its prevalence in the study of biofilms [31, 32]. Biofilms are created when bacterial colonies are closely associated and linked by an extracellular matrix. Biofilms are considered the dominant manifestation of bacterial growth in nature and often consist of many species of bacteria together [31]. Thus, biofilms are associated with the cell-cell
communication which is required in order to organize the individual cells into the biofilm [31, 32]. It is this characteristic that makes \textit{B. Subtilis} an appropriate candidate for work concerning emergent behavior at the cellular level.

Contribution and Hypothesis

Emergent behaviors abound in biology, and many phenomena concerning the human body can also be considered emergent behaviors, e.g. cell migration during wound healing and the metastasis of cancer cells [18, 20]. Non-healing wounds alone are estimated to cost the United States around $3 billion annually [33], and cancers are estimated at placing a burden of roughly $60 billion annually in the United States [34]. However, neither of these metrics fully accounts for the intangible costs due to suffering, trauma, and other psychosocial costs [34]. Thus, the study and understanding of the emergent behaviors of cells is well-motivated by existing medical concerns.

Much of the study of emergent behavior up until this point has been the work of philosophers and computational scientists, with less literature available regarding its applications to biology. Specifically, Millonas’ modeling of swarming ants [11], Cucker and Smale’s analysis of flocking birds [15], and Campos et al.’s analysis of modeling human cells [26] all produce mathematical models. However, these mathematical models are not then considered in terms of how they can be used to engineer the underlying biology.

Even in cases where a complex system approach is taken to understand biology, the nature of the emergent phenomena is not closely analyzed or categorized. For example, Deisboeck and Couzin explore how swarm behaviors (a type of emergent behavior) appear in cancer cell populations, with a particular focus on how the cells come to a consensus on which
direction to move [2]. It is at this intersection of modelling emergent behavior and biology where this thesis makes a key contribution.

Although a given emergent behavior might be observed to be the same in two different populations, I argue that there may be at least two distinct underlying types of emergent behavior. Concretely, this means that although *B. Subtilis* and HDFs may experience 2D collective migration when cultured *in vitro*, the types of that emergent behavior are distinct. I do not mean in the philosophical sense – i.e. strong or weak emergence – but rather that the collective migration emerges in one of the following two ways:

1. **Primary**: An existing property of the cells results directly in the observed emergent phenomenon.
2. **Secondary**: Upon certain environmental conditions being met, the cells evolve a new behavior which results in the observed emergent phenomenon.

These are both forms of weak emergence as defined previously. Primary emergent behavior is caused by underlying behaviors or characteristics which are constant for the individuals under study while secondary emergent behavior is caused by behaviors which are reactions to changes in the environment. The hypothesis of this thesis is that *B. Subtilis* experiences collective migration as a result of a primary emergent behavior and that HDFs experience collective migration as a secondary emergent behavior.

A common counterargument to making this distinction is that from the perspective of understanding the complex system, this distinction is largely irrelevant – many of the same predictions can be made so long as the corresponding mathematical model captures the resulting emergent behavior. However, this distinction is relevant for guiding the kinds of questions that should be asked upon observing a cellular emergent behavior. Primary emergent behaviors are
the results of always-present characteristics which can be observed in the cells. Secondary emergent behaviors are not observable until a certain threshold is reached. For instance, it may not be possible to predict that HDFs will collectively migrate simply from observing a low-density culture of the cells. However, once some threshold is reached, which corresponds to individual cells detecting the presence of their neighbors and reacting, collective migration may result.

Though this thesis utilizes HDFs and *B. Subtilis* to further this idea that there exist two biologically-relevant types of emergent behavior, it makes few claims as to which mechanisms result in the observed behaviors in the cells. The use of laboratory experiments is motivated primarily by the intuition that the underlying differences in sophistication between these types of cells will result in the observation of primary and secondary emergent phenomena. The purpose of the cellular automata and modeling aspect is to provide *in silico* evidence for the existence of this distinction.

Primarily, this thesis seeks to explore that primary and secondary emergent behaviors exist and are distinct from one another. Secondarily, the work herein should demonstrate that with respect to emergent behavior, these two model organisms exhibit the same emergent behavior from different causes. As a tertiary point, the cellular automata created for the purposes of this thesis are available in Appendix A and are created with the intention of providing a general tool for modeling emergent behaviors.
Materials and Methods

Each experiment is designed to image clusters of cells and quantitatively measure their collective migration from their starting point via image analysis.

Human Dermal Fibroblast Experiments

Cell Culture

Freshly mailed HDFs were cultured in a T75 flask in Fisher Scientific Medium 106 with Fisher Scientific Low Serum Growth Supplement (LSGS) up to approximately 90% confluence. At this point, cells were passaged and split 1:2 in T75 flasks, then allowed to grow up to approximately 80% confluency before being used.

High, Medium, and Low Density Setup

Since the goal of the experiments is to isolate primary and secondary emergent behaviors, three different densities of cells were prepared: high, medium, and low. The high density selected was $33.3 \cdot 10^4 \text{cells mL}^{-1}$, for a target of 333 cells per 20μL. The medium density was a 10x dilution of $3.3 \cdot 10^4 \text{cells mL}^{-1}$, and the low density was a 100x dilution of $0.3 \cdot 10^4 \text{cells mL}^{-1}$.

Plate Setup

A 24-well plate was then subdivided into three density sectors – high, medium, and low – with 8 wells each. Each density sector was subdivided into two additional size sectors – large and small – for a total of 6 sectors. In each large sector, one aliquot of 10 μL of the appropriate density solution were dropped carefully onto one marked dot in each well. In each small sector,
one aliquot of 5 μL of the appropriate density solution was similarly placed in each well. Figure 4 illustrates this layout. In order to preserve the humidity within the plate, the space between wells was filled with ultrapure water. After seeding, the 24-well plate was placed in the incubator at 37°C with 5% CO₂.

Imaging Strategy

6 hours after the initial seeding, each well in the 24-well plate was washed with warm, sterile phosphate buffer solution (PBS), and filled with LSGS-containing medium 106. The cells were then imaged once every hour for 3 hours, then once every 12 hours for the remaining 72 hours. Unfortunately, due to COVID-19 and the closing of the Harvard Active Learning Labs, this imaging was not completed (as reflected below).

Control

As a control experiment, a separate 6-well plate was set up with one well containing a single 100 μL high density cluster of cells and another well containing four 20 μL high density clusters of cells. The expected result of the single cluster well was to observe a large cell cluster’s collective motion. The expected result of the well with multiple clusters was that...
clusters would grow and migrate relative to each other as the fibroblasts produced paracrine signaling molecules and thus concentration gradients of these molecules in the well. The single cluster well thus would control for collective migration’s dependence on cluster size, and the multi-cluster well would control for the symmetry-breaking introduced by the presence of other cells in the well.

*Bacillus Subtilis* Experiments

Cell Culture

Initial colonies of *B. Subtilis* expressing Green Fluorescent Protein (GFP) were supplied by Dr. Melissa Hancock, and were to be grown to an optical density at 600 nm of approximately 2 (OD$_{600}$ $\approx$ 2) overnight in a shaking incubator before performing the following experiments. However, the initial agar plates streaked with *B. Subtilis* stock anomalously did not grow. In the interest of time, coupled with growing concerns about COVID-19, these experiments were *not* performed. The following is the experimental setup that would have been used without the interruption by COVID-19.

High, Medium, and Low Density Setup

Using the cultured flask of *B. Subtilis* at OD$_{600}$ $\approx$ 2, three aliquots were to be made, representing high, medium, and low densities. The high density was to be at the original OD$_{600}$ $\approx$ 2, the medium density was to be a 10x dilution at OD$_{600}$ $\approx$ 0.2, and the low density was to be a 100x dilution at OD$_{600}$ $\approx$ 0.02.
Plate Setup

Three agar plates were to be prepared and labeled with their corresponding density, as well as the intended initial locations of colonies. Note that these initial locations were to be no closer than 2 cm, allowing for approximately 8 well-spaced colonies per plate.

Imaging Strategy

Each plate was to be seeded by placing a drop of 3 μL of the appropriate density aliquot of *B. Subtilis* onto each marked initial location. The plates were to be incubated in a non-shaking incubator while being imaged once every 10 minutes for 72 hours by a GFP-fluorescent live-imaging camera.

Control

In parallel with the HDF experiments, a control plate was to be setup with a large colony and another control plate with several colonies in proximity to one another. The large colony would reveal whether the size of the colony (not its density) was related to its ability to migrate, and the multiple colonies would reveal the effects of proximal colonies on each other’s migration.

Simulation

All coded aspects to this thesis were written for Python 3.7.3, and they make use of the following libraries: NumPy v1.16.1, MatPlotLib v3.0.2, Pandas v0.24.1, SciPy v1.2.1, and Seaborn v0.10.0, which are all under the BSD license. All analysis and simulation were
performed on a 2015-Macbook Pro with a 2.5 GHz Intel i7 and 16 Gb of memory. All code used can be found in Appendix A.

*Cell.py, CellDefinitions.py, and Simulation.py*

These three files define the main components of the cellular automata for simulation. Primarily, *Cell* objects represent the agents of the simulation, and *Simulation* objects maintain the required state to run the simulation.

The *SimpleCell* class defined in *CellDefinitions.py* extends the base *Cell* object from *Cell.py* and is the object of interest for this thesis. This object has three modes, represented by methods, by which it can move on a given timestep of the simulation.

The first method of movement is *randomWalk*, which performs a random walk with waiting. Upon a call to *randomWalk*, the cell chooses a new space between adjacent available spaces and its current space with equal probability then moves or stays in place accordingly. Available spaces are defined as those which are within the edges of the environment, immediately to the left, right, above, or below, and not occupied by other cells. I base the following brief derivation of the relationship between expected distance and time on the work of Codling et al. [18].

First, consider the x-axis of motion. A cell may take a step to the right or a step to the left along the x-axis, and the three remaining choices (above, below, and stay) have no impact on the cell’s position on the x-axis. Thus with probability $l = r = \frac{1}{5}$, the cell takes a step in either direction, and with probability $s = 1 - l - r = \frac{3}{5}$, the cell’s position on the x-axis does not change. Then after $t$ timesteps, the probability that the cell has taken $n_l$ steps to the left, $n_r$ steps
to the right, and \( n_s \) “steps” which do not affect the \( x \)-axis position is

\[
P(t = n_l + n_r + n_s) = \binom{n_l+n_r+n_s}{n_r}
\]

The number of ways in which the cell may take \( n_r \) steps to the right is

\[
\binom{n_l+n_r+n_s}{n_r}
\]

Thus, the distribution of steps to the right is then

\[
P(n_r) = \frac{(n_l+n_r+n_s)!}{(n_l+n_s)!(n_r)!} \cdot l^{n_l} r^{n_r} s^{n_s}
\]

which is a multinomial distribution with mean \( \langle n_r \rangle = r \cdot t \). I then claim that the distribution of the \( x \) and \( y \) positions are independently multinomial via the argument made by Codling et al.; the distribution of the \( x \) and \( y \) positions can be considered independent for sufficiently large \( t \) [18].

Codling et al. presents the expected distance in two dimensions for unit length step size and discrete timesteps taken, \( t \):

\[
E(d) = \frac{\sqrt{\pi t}}{2}
\]

For the purposes of this thesis, I am interested in simply the result that the expected distance after \( t \) timesteps is

\[
E(d) \propto \sqrt{t}.
\]

The second method of movement for a SimpleCell object is biasedRandomWalk, which searches for cells within a radius provided by the command line flag “--radius.” Upon locating a cell within its target search radius, this cell increases the probability that it will move in a direction which brings it closer to that “neighboring” cell. If there are no nearby cells, then the cell switches to the randomWalk method.

The third and final method of movement for a SimpleCell object is thresholdBiasedRandomWalk. In this method, the cell calculates a local cell density. The radius of cells counted in this calculation is determined by the command line parameter “--density-radius.” If the density is above a threshold determined by “--density-threshold,” then the cell switches to the biasedRandomWalk method, otherwise, the cell will simply default to the randomWalk method.
run.py, generate_island.py, and analyze.py

These files use the main functionalities provided by the above objects. Of particular interest is run.py; running ./run.py -h reveals all command line arguments. The simulation can be run using any one of the provided SimpleCell object methods, and it can display output to console for a visual, as in Figure 5. Additionally, the simulator can be run in “stats” mode where relevant statistics are output to a selected file for later analysis.

![Figure 5. The console output that can be displayed by the simulation software. A shows output from the command ./run.py --radius 4 --density-radius 4 --density-threshold 0.2 --walk-type rw --init newisland.txt -d 1 --cell o B shows output from the same command, but with --walk-type brw instead.](image)

generate_island.py allows for the generation of an initial conditions input to run.py, representing a cell cluster, or island, within the simulation. This file was used to generate all inputs to the simulator and should be used to replicate the results of this thesis. An important assumption made by this generator is that a cell cluster has no gaps – i.e. that each cell in the cluster begins in physical contact with its neighbors.

analyze.py is the least general file included and is provided mainly for reproduction of the statistical methods and figures included in this thesis. This file’s contents are not intended to be
used for general analysis, and this file should only be considered as a way to replicate the exact results in this thesis.

COVID-19 Notes

While much of the experimental setup would have remained the same, the following are notes about how each experimental setup was impacted by COVID-19.

Human Dermal Fibroblast Experiments

After beginning the HDF experiments, I came to the realization that imaging the cells by hand was not sufficient for collecting quantifiable data regarding the potential collective motion of the cells. In particular, the hand imaging introduces discrepancies in orientation (are two images at the same angle), position (do the origins of two images coincide), and quality (are some images better focused than others). While these discrepancies do not pose a threat to the ability to perform some qualitative analysis, a full quantitative analysis requires that these discrepancies be minimized. Otherwise, image analysis techniques may reveal collective migration where there was none or negate migration when it did occur. Furthermore, in the case of fibroblasts, it is important to be able to mark individual cells within a cell cluster and track their motion through the tissue, as is done by Vedula et al. [3], since even if the whole cluster of cells does not migrate, subsets of the cluster may. A lack of continuity from image to image makes tracking this impossible with hand imaged cells.

In order to collect the more consistent and fine-grained data suitable for image analysis, the experimental setup is the following.
The culture, and density setups would have remained as above. The plates chosen would be 96-well plates instead of 24-well plates in order to place each cell cluster in its own well. This difference is due to the practical consideration of attempting to hand image 96 separate wells. Thus, there would be 32 wells per density sector, with 16 wells per large sector and 16 wells per small sector. Hand imaging would be replaced by live microscopy through the Harvard Center for Biological Imaging (HCBI). Imaging would take place over 72 hours, with one image per cell cluster every 15 minutes. Additionally, in a machine like HCBI’s Celldiscoverer7, up to 14 96-well plates would be used to provide population-level data about the root mean square position of cell clusters in each group (density and size sector).

_Bacillus Subtilis_ Experiments

Due to the failure to produce a viable plate of _B. Subtilis_ colonies and a lack of time due to COVID-19, the _B. Subtilis_ experiments were not carried out. In order to still provide some data for the purposes of producing quantitative results, images of _B. Subtilis_ colonies were kindly provided by Dr. Melissa Hancock from her previous work.

The planned experiments above would have been carried out with imaging identical to that carried out by Dr. Hancock. However, the setup of the agar plates and colony layout would have been as I originally described in Materials and Methods. In particular, the colonies would not have been as close together as observed in the results below from Dr. Hancock.
Results

Due to the interruption by COVID-19, these results are incomplete. All conclusions drawn from these results should therefore be interpreted as tentative.

Human Dermal Fibroblast Experimental Results

The HDF experimental results are qualitative due to the inability to analyze them quantitatively.

Figure 6 displays the different initial densities of clusters of fibroblasts, where the large black mass in the center of the images is the marking used to indicate the origin.

The main results of these experiments are split into three sections: morphological changes, deterioration, and migration.

Morphological Changes

All three density sectors showed significant morphological changes in individual fibroblasts. These changes were expected, since they are the fibroblasts extending filopodia and lamellipodia in order to migrate around on the plate surface. Unexpected, however, is that the extension of these filopodia and lamellipodia did not appear to be consistently along a particular
axis of motion. A representative example of these types of changes appears in the top half of Figure 7.

Figure 7. All images shown were taken at 4x magnification. A is the initial seeding of a well in a small, low density sector. B is 1 hour after the initial seeding. C is 2 hours after the initial seeding. D is 14 hours after the initial seeding. E is an image from the large, single cluster control 24 hours after the initial seeding. In all images, the black, out of focus masses are the pen markings used to indicate the initial positions of the cell clusters. Panels A-D show the progression of changes within a single small, low density sector well.
The large cluster control experiment, shown in the bottom half of Figure 7, was imaged at a substantially later time point than were the experiments (around 48 hours after seeding as opposed to 14 hours after seeding) and shows the expected lanky morphology of HDFs, with several sub-clusters appearing to be oriented in the same direction.

Deterioration

In several of the high density sector wells, it was observed that the fibroblasts did not seem to change from their roughly circular shape. Instead, they appeared to have undergone apoptosis, as highlighted by Figure 8.

In order to determine the reason for this apparent breakdown, further experiments would have been performed with larger cell clusters, as the control experiment did not appear to suffer from this issue.

Migration

Across different density clusters, it is apparent that small groups of fibroblasts moved significantly between images, resulting in them not being in frame. Due to the discontinuity of the imaging, it is uncertain where these fibroblasts went as well as how quickly they migrated.
Bacillus Subtilis Experimental Results

In order to process the data provided by Dr. Hancock, I used Fiji (ImageJ v2.0.0) from the NIH [35, 36]. Specifically, all images were converted to grayscale, then an intensity threshold was applied to filter out noise, as is standard in image analysis. ImageJ’s built-in MTrack2 [37] was used to locate and track objects between frames, producing an output of the coordinates of the center of each object which was subsequently analyzed and plotted via analyze.py.

Figure 9. Different colored line segments correspond to the positions of the centers of different colonies within the images provided by Dr. Hancock. Colony center position tracking was performed via ImageJ’s MTrack2 plugin.
As is visible in Figure 9, the positions of the bacterial colonies did not appear to shift substantially over the course of the experiment. In order to obtain a meaningful measure of distance, I use a standard formula of root mean square distance (RMSD). The position at each timestep is centered on the starting point by subtracting the starting point. Then, the squared distance of the centered points is computed, \( SquaredDistance(t) = x(t)^2 + y(t)^2 \). Finally, the root of the mean of the squared distances at each timestep is taken across the colonies to compute the final RMSD for \( n \) colonies, \( RMSD(t) = \sqrt{\frac{\Sigma_{t=1}^{n} x_i(t)^2 + y_i(t)^2}{n}} \). To quantify how close to a random walk the motion is, I performed a linear regression on both RMSD versus timestep and RMSD versus the square root of timestep. These regressions yielded r-squared coefficients, which are standard measures of fit, of \( r_{\text{linear}} \) and \( r_{\text{random walk}} \), respectively. Whichever coefficient is closer to unity better captures the nature of the motion. As visualized in Figure 10, the motion of

![A] RMSD - B. Subtilis

![B] RMSD - B. Subtilis

Figure 10. A shows RMSD of B. Subtilis colonies versus time in black. B shows RMSD of the colonies versus the square root of time in black. Both plots are fitted with a linear regression, which is shown in red.
the *B. Subtilis* colonies is slightly closer to a random walk, with $r_{\text{randomWalk}} = 0.992$ and $r_{\text{linear}} = 0.951$.

Simulation Results

Four main categories of simulation were performed with 40 trials per category. Within each category of simulation, the root mean square distance (RMSD) is computed as in the *B. Subtilis* results, $RMSD(t) = \sqrt{\frac{\sum_{i=1}^{n} x_i(t)^2 + y_i(t)^2}{n}}$. As with the *B. Subtilis* experiments, if a category yields an $r_{\text{randomWalk}}$ closer to unity than $r_{\text{linear}}$ is, it is reasonable to assume that this category is performing a random walk. If $r_{\text{linear}}$ is closer to unity than $r_{\text{randomWalk}}$ is, then it is reasonable to assume that this category is not a random walk and is instead a result of emergent behavior of the cells.

![Figure 11](image-url)  
*Figure 11. Plots of RMSD are shown. A shows RMSD versus timestep in black. B shows RMSD versus the square root of timestep in black. A linear regression fitted to each line is overlaid in red on each plot.*
Figure 11 shows linear regression lines fitted to RMSD versus time of the cell clusters in the “Random Walk” category. As expected, the line fitted to RMSD versus the square root of timestep appears to be a better fit than the line fitted to RMSD versus timestep, indicating a random walk. This is confirmed by $r_{randomWalk} = 0.927$ and $r_{linear} = 0.850$.

Next, in the threshold categories, two different simulations were performed. The first, “Threshold – Sparse,” is a simulation of a cluster of cells in a sparse environment which included 100 cells in the initial cell cluster. The second simulation, “Threshold – Dense,” included an initial cluster of 1000 cells. In both simulations, cells performed a random walk until they were in an area with a local cell density above a particular threshold, at which point they switched to moving towards nearby cells. In the sparse environment, the cell clusters appeared to perform a random walk, with $r_{randomWalk} = 0.803$ and $r_{linear} = 0.676$. In the dense environment, the cell clusters also appeared to perform a random walk, with $r_{randomWalk} = 0.973$ and $r_{linear} = 0.906$. Note the difference in $r_{linear}$ for the sparse and dense environments.

The final category, “Towards Neighbors” simulated cells which always try to go in the direction of “nearby” cells. “Nearby” was defined as being within a radius of 4 lattice squares of the cell. These cell clusters performed a random walk, with $r_{randomWalk} = 0.998$ and $r_{linear} = 0.977$.

Overall and Intersectional Results

The primary results are those from the $B. Subtilis$ and simulated experiments. Ranking by $r_{linear}$, from lowest to highest results in the following: “Threshold – Sparse,” “Random Walk,” “Threshold – Dense,” $B. Subtilis$, and “Towards Neighbors.” Ranking by $r_{randomWalk}$ from lowest to highest results in the following: “Threshold – Sparse,” “Random Walk,” “Threshold – Dense,” $B. Subtilis$, and “Towards Neighbors.” Results of note are the relative rankings of the
“Threshold” categories to each other, the ranking of “Random Walk,” and the ranking of *B. Subtilis*. 
Discussion

Experimental Analysis

In the laboratory experiments, the hypothesis of this thesis is that *B. Subtilis* experiences a primary emergent behavior, where the emergence of the behavior is not dependent on an external environmental condition (like local cell density). The hypothesis of this thesis is also that HDFs, in contrast, experience a secondary emergent behavior, where the emergence of the behavior is dependent on an external environmental condition.

The results of the *B. Subtilis* data (images provided by Dr. Hancock) as provided show that $r_{\text{random walk}} > r_{\text{linear}}$, which indicates that the *B. Subtilis* colonies perform a random walk. This does not support the hypothesis that they undergo a primary emergent behavior, where the expectation is that the RMSD would vary non-linearly with the square root of time.

Without quantitative analyses of the HDF motion, it is impossible to make quantitative claims about the nature of the motion of the clusters of fibroblasts. However, there is some evidence that different behaviors were observed which varied from density sector to density sector. For example, higher density clusters of fibroblasts experienced cellular deterioration and apoptosis at much higher qualitative rates than did lower density clusters. The control experiments seem to suggest that this may be due to another issue, such as the fibroblasts running out of ATP during the initial seeding process, since the large cluster of fibroblasts in the control did not experience this deterioration. This is similarly supported by the lack of deterioration in the lower density sectors, where fewer fibroblasts would have been consuming nutrients, thus making the possibility of a lack of nutrients less likely.
Due to the limitation of the experimental results by COVID-19, it is clear that further experimentation in both the *B. Subtilis* and HDF domains is necessary to substantiate the above preliminary observations.

Simulation Analysis

Based on the derivation of the expected distance achieved in a 2D random walk, a true random walk’s RMSD should vary linearly with the square root of time. Table 1 displays the values of $r_{\text{linear}}$ and $r_{\text{randomWalk}}$ for each category of simulation and indicates the value which is closer to unity. Given that each cell in the simulation performs a form of random walk, the expected behavior is that the cluster of cells should also perform a random walk. Deviations from this expectation indicate the emergence of a new behavior.

<table>
<thead>
<tr>
<th></th>
<th>Random Walk</th>
<th>Threshold – Sparse</th>
<th>Threshold – Dense</th>
<th>Towards Neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{\text{randomWalk}}$</td>
<td>0.927</td>
<td>0.803</td>
<td>0.973</td>
<td>0.998</td>
</tr>
<tr>
<td>$r_{\text{linear}}$</td>
<td>0.850</td>
<td>0.676</td>
<td>0.906</td>
<td>0.977</td>
</tr>
</tbody>
</table>

Table 1. Bolded values indicate which of $r_{\text{randomWalk}}$ and $r_{\text{linear}}$ (r-squared coefficients obtained by linear regression) is larger for the category. Categories with $r_{\text{randomWalk}}$ closer to unity are more similar to a random walk – they can be linearized by plotting RMSD against the square root of time. Conversely, categories with $r_{\text{linear}}$ closer to unity are more linear when RMSD is plotted against time.

The simulation categories were designed such that the “Towards-Neighbors” category should exhibit primary emergence – i.e. the emergent behavior is not dependent on some environmental condition. In contrast, the “Threshold” categories should exhibit varying degrees of secondary emergence – i.e. the emergent behavior is dependent on some environmental condition. The “Random Walk” category should exhibit no emergent behavior and simply follow a random walk.
The primary benefit to the simulated environment is that all methods of “communication” between cells are known and that cells between simulations and within a given simulation are identical except for the method by which they move. Furthermore, each simulation is run with the same parameters except for which type of random walk is performed and that the “Threshold – Dense” category was initialized with 10x the number of cells (1000 cells total).

The simulation setup begs the question of whether the simulation was designed simply to obtain the expected results. There are two main reasons this is not the case. First, the simulation does switch behaviors based on local cell density in the “Threshold” categories; however, I do not know a priori what consequence this switch in behaviors, if any, will have on the final results. Second, the simulation is designed to assist in supporting or refuting the hypothesis that B. Subtilis undergo a primary emergent behavior. Thus, it is necessary to create a simulation which should exhibit a primary emergent behavior as well as to create a simulation which should exhibit a secondary emergent behavior.

Despite each of these factors, it is clear from Table 1 that each category of simulation performed a random walk. However, it should be noted that in the cases of the “Threshold - Dense” and “Towards Neighbors” categories, the $r_{linear}$ values are high when compared with the “Threshold - Sparse” and “Random Walk” categories, respectively. This is significant in two main ways.

First, the increase in linearity from “Threshold - Sparse” to “Threshold - Dense” indicates that there was a density-dependence to the behavior of the simulated cell clusters when performing different types of walks dependent on local density. This is consistent with the simulation design, which explicitly switches between a true random walk and a “Towards Neighbors” walk upon reaching a set local-density threshold. Furthermore, this suggests that in
the instance of the emergent behavior of collective motion, the deviation of the cluster motion from a true random walk is likely a key indicator in detecting the presence of emergent behavior.

Second, the extremely linear nature of the RMSD variance with time observed in the “Towards Neighbors” category \((r_{\text{linear}} = 0.977, r_{\text{randomWalk}} = 0.998)\) seems to conflict with the deviation from the expected random walk behavior in the “Random Walk” category \((r_{\text{linear}} = 0.850, r_{\text{randomWalk}} = 0.927)\). This anomaly may be wantonly attributed to “emergent behavior,” but this is not a useful interpretation without an explanation. Instead, offer the following explanation, which should be substantiated by further experimentation. In the “Random Walk” category, the cells each individually perform a random walk, but they cannot occupy the same space as one another. Intuitively, this means that the cells should diffusively radiate away from their starting point, which is the principle described by a random walk. However, since each cell is doing this, the cluster’s overall movement is actually dampened due to some cells moving in opposite directions, which results in behavior with less random walk character – i.e. the cluster’s RMSD varies less linearly with the root of time. In contrast, the “Towards Neighbors” category cluster does not experience nearly as much diffusive effect, since all cells try to move closer to nearby cells. The result is that the cluster remains more tightly clustered together than in the “Random Walk” category and thus may be considered more accurately as a single entity. This may then explain that the “Towards Neighbors” category demonstrated more random walk character in its motion as a cluster than did the “Random Walk” category. These are forms of emergent behavior, though not the anticipated ones.
Overall Analysis

Through discussion of both the laboratory and simulated experiments, I have noted important aspects of control experiment and simulated experiment design. Although the results of these observations appear to cause a chicken-egg problem, I resolve this by recommending that both types of experimental design occur concurrently and interactively.

The control experiments for the laboratory experiments are not sufficient for constraining the experimental design to a clear set of causal relationships. Furthermore, it has become clear that the results of the simulated experiments should be used in order to inform laboratory control experiment design. For example, the simulated “Random Walk” and “Towards Neighbors” categories exhibited deviations from expected behavior. This deviation may show up in the current design of the laboratory experiments, which are further complicated by the noisiness of data collected from living systems. Thus, I should have set up control experiments that controlled for these deviations in order to make stronger claims about their nature and cause.

Similarly, the simulated experiment design should be informed by the results of the laboratory experiments. It is easy to see from qualitative observations of the fibroblasts that sparsity and shape of cell clusters are much more pronounced than in either the B. Subtilis or the simulated experimental setups. This means that this parameter should be simulated and data collected in silico to better inform the design of both sets of experiments.

Overall, this then appears to be a chicken-egg problem, where each type of experiment depends on the observations made in the other. The natural solution then is to focus on adjusting each experimental setup incrementally and in real time as new observations are made. This requires a significantly more interactive approach than was taken in this thesis. This “back-and-forth” requires being explicit about how the choices made in the simulated experiments apply to
those made in the laboratory experiments. This is important not only for consistency of results and for writing papers, but also because these observations may be easier to explain in one environment than in the other.

Significance

The primary role of this thesis in relation to the broader intersection between computer science and bioengineering is to probe computational concepts for their implications for the underlying biology. In this thesis, this is the question of what different kinds of emergent behavior mean for the HDF and *B. Subtilis* systems. In particular, this thesis explores how even without direct knowledge of the exact mechanism of emergent behavior, it is still possible to begin quantifying the behavior of the system and produce meaningful observations. The precise way in which fibroblasts or *B. Subtilis* cells individually communicate with one another in order to collectively move is unknown. However, quantifying the behavior in the form of RMSD allows for a meaningful comparison to be drawn with simulated results. Furthermore, this thesis proposes a subdivision of a concept in computation – emergent behavior – through a biological lens.

Secondarily, this thesis returns to simple model organisms (HDFs and *B. Subtilis*) to demonstrate that comparisons between well-understood organisms may offer insights at the systems level. The significance of this effort is that with careful forethought, the selection of apparently disparate organisms and subsequent comparison as systems can now be performed using computational tools. As a result, while such a comparison may have been out of scope in the past, it is now a reasonable method for learning new information about each individual organism when it is viewed as a system.
A tertiary goal is to provide software whose usability is unrelated to the success of the thesis. More broadly, scientific work which builds software for use in a particular paper should do so with future users in mind, in much the same style as software engineers build code. The domain of writing extensible code now includes every field, not just computer science.

Future Work

The next steps for the work contained herein are twofold: augmenting and fully exploring the experiments proposed and exploring the implications that the success or failure of this work may have.

With respect to augmenting and exploring the work in this thesis, there is a clear lack of data due to the disruption of COVID-19. Beyond this disruption, there are also gaps in the experimental plan, such as insufficient testing parameters on each of the laboratory experiments and incomplete control experimental designs. The augmentation of this work looks like pursuing a similar set of experiments by varying densities of populations of these organisms and quantifying the resulting behaviors. In complement to that, the simulations of this thesis should also be expanded, and the significance of more parameters to the final quantified results should be tested. Some parameters of interest are how cell division impacts the results of the simulated experiments, a sparsity parameter so that initial conditions could have gaps between cells, and the ability for cells to transition into different phenotypes during the simulation. The simulation aspect of this work begs the development of performant software for the realistic modeling of human and bacterial cells. As stated by Ruxton and Saravia, biological realism in computer models is key to improving the accuracy of conclusions drawn from these models [38]. While this appears to be an obvious point, the implementation of biological realism in computer models
requires a set of experiments in and of itself which is often costly, since key assumptions (e.g. cells not being allowed to occupy the same space in a cellular automaton) must be validated experimentally.

With respect to the implications of this work generally, I expect that future work on emergent behavior and complex systems will have far-reaching consequences for cell biology. These consequences include not only improved understanding the processes involved in wound healing and cancer systems, but also the control, tunability, and predictability of these systems. This most obviously relates to work in medicine, from therapeutics to bioelectronics. For example, if conceptual frameworks are developed around the different types of emergent behavior found in particular kinds of human cells, it becomes possible to systematically study how to target drugs to different kinds of emergent behavior. Furthermore, there is a growing need for intersectional work between bioengineers, cell biologists, computer scientists, and software engineers in order to further the human understanding of cell biology as a complex system.

In short, this thesis is a work in the pursuit of using computer software to further human understanding about how emergent behavior impacts cell biology.
References


Appendix A

Cell.py

# defines the Cell base class, which should be extended for other cell types
class Cell:
    def __init__(self, chars, cellValue, cellType):
        # ensure that chars contains the correct cell type
        if cellType not in chars:
            raise ValueError(f'{chars} does not contain "{cellType}" as a key')

        # ensure that only single character chars are used
        for k in chars:
            if len(chars[k]) != 1:
                raise ValueError(f'All values in {chars} must be single characters')

        # a dictionary mapping types to corresponding chars
        self.chars = chars

        # set this cell's value and type
        self.cellValue = cellValue
        self.cellType = cellType

        # set the character for this cell to the corresponding one for its cell type
        self.char = self.chars[self.cellType]

        # reset to False at each time step
        self.moved = False

    # this method should be overridden for different types of cells
    def nextState(self, board, x, y):
        return self

    # returns the number of neighbors of a coordinate x, y within radius
    def getNeighbors(self, board, x, y, radius=1):
        s = 0

        # set up the lower and upper limits for our "scan"
        lowYLim = max(0, y - radius)
        highYLim = min(len(board) + 1, y + radius + 1)

        for row in board[lowYLim:highYLim]:
            # analogous to above but for X direction
            lowXLim = max(0, x - radius)
            highXLim = min(len(row) + 1, x + radius + 1)

            # sum the number of living cells
            s += sum(map(lambda x: int(x) != 0, row[lowXLim:highXLim]))

        # ignore ourselves in returning the answer
        return s - int(board[y][x])
```python
def __str__(self):
    return self.char

def __int__(self):
    return self.cellValue

### Define equality and comparison operators

def __eq__(self, value):
    return self.cellValue == value.cellValue

def __ge__(self, value):
    return self.cellValue >= value.cellValue

def __gt__(self, value):
    return self.cellValue > value.cellValue

def __le__(self, value):
    return self.cellValue <= value.cellValue

def __lt__(self, value):
    return self.cellValue < value.cellValue

def __ne__(self, value):
    return self.cellValue != value.cellValue
```

```
CellDefinitions.py
from Cell import Cell

from random import choice, random
import math

""
* This file defines cells and empty cells:
* Cells have the ability to perform 3 kinds of motion:
* random walks, biased random walks, and thresholded random walks
""

# defines the completely empty cell

class EmptyCell(Cell):
    def __init__(self, chars):
        super().__init__(chars, cellValue=0, cellType='empty')

# returns the direction relative to the current position of a new position

def getDirection(newPosition, currentPosition):
    if newPosition - currentPosition > 0:
        return 1
    elif newPosition - currentPosition < 0:
        return -1
    else:
        return 0
```
# defines a basic living cell
class SimpleCell(Cell):
    def __init__(self, chars, 
                 x, y, 
                 walkType='rw', 
                 tDivision=32, 
                 radius=2, 
                 densityRadius=4, 
                 densityThreshold=0.1, 
                 divisionProbability=0.5):
        super().__init__(chars, cellValue=1, cellType='simple')

        self.walkType = walkType

        self.walks = {
            'rw': self.randomWalk,
            'brw': self.biasedRandomWalk,
            'tbrw': self.thresholdBiasedRandomWalk
        }

        self.t = 0

        # time until cell division occurs
        self.tDivision = tDivision

        # applies to biased random walks (how far to look for cells)
        self.radius = radius

        # applies to the search area for deciding density threshold
        self.densityRadius = densityRadius

        # determines the density threshold above which to switch to biased random
        walks
        self.densityThreshold = densityThreshold

        # the probability that an attempted division will succeed
        self.divisionProbability = divisionProbability

        # the x, y coordinates of this cell on the board
        self.coordinates = (x, y)

    def randomWalk(self, board, x, y):
        if not self.moved:
            # allow this cell to stay in place if no new positions are available
            newCoordinates = list()

            if x > 0:
                newCoordinates.append((x - 1, y))

            if x < len(board[0]) - 1:
newCoordinates.append((x + 1, y))

if y > 0:
    newCoordinates.append((x, y - 1))

if y < len(board) - 1:
    newCoordinates.append((x, y + 1))

# filter out non-empty cells
newCoordinates = list(filter(
    lambda pos: board[pos[1]][pos[0]] == EmptyCell(self.chars),
    newCoordinates
))

# can also stay in place
newCoordinates.append((x, y))

# select a new position randomly from those available
newX, newY = choice(newCoordinates)

# perform the actual move
board[y][x] = EmptyCell(self.chars)
board[newY][newX] = self
self.coordinates = (newX, newY)

# this cell has moved and a time step has passed (i.e. don't divide on the first time step)
self.moved = True

# perform a cell division if we have the space and it's division time
if len(newCoordinates) > 1 and self.t == 0:
    # if our division "succeeds"
    if random() < self.divisionProbability:
        # search for an empty space in which to divide
        while board[newY][newX] != EmptyCell(self.chars):
            newX, newY = choice(newCoordinates)

        board[newY][newX] = SimpleCell(
            self.chars,
            newX, newY,
            walkType=self.walkType,
            tDivision=self.tDivision,
            radius=self.radius,
            densityRadius=self.densityRadius,
            densityThreshold=self.densityThreshold,
            divisionProbability=self.divisionProbability
        )
        board[newY][newX].moved = True
        board[newY][newX].tDivision = self.tDivision
        board[newY][newX].divisionProbability = self.divisionProbability

# performs a biased random walk
def biasedRandomWalk(self, board, x, y):

# the radius in which to search for cells
r = self.radius

newCoordinates = list()
for j in range(max(0, y - r - 1), min(y + r + 2, len(board))):
    for i in range(max(0, x - r - 1), min(x + r + 2, len(board[0]))):
        # skip this cell
        if i == x and j == y:
            continue

        # if the target cell is within the searching radius
        if math.sqrt((y - j) ** 2 + (x - i) ** 2) <= r:
            if board[j][i] != EmptyCell(self.chars):
                directionX = getDirection(i, x)
                directionY = getDirection(j, y)

                newCoordinates.append((x + directionX, y))
                newCoordinates.append((x, y + directionY))

# filter out coordinates that contain cells
newCoordinates = list(filter(
    lambda pos: board[pos[1]][pos[0]] == EmptyCell(self.chars),
    newCoordinates
))

# if we don't have any coordinates found this way, perform a random walk
if len(newCoordinates) == 0:
    self.randomWalk(board, x, y)
else:
    # select a new position randomly from those available
    newX, newY = choice(newCoordinates)

    # perform the actual move
    board[y][x] = EmptyCell(self.chars)
    board[newY][newX] = self
    self.coordinates = (newX, newY)

    # this cell has moved and a time step has passed (i.e. don't divide on the
    # first time step)
    self.moved = True

    # perform a cell division if we have the space and it's division time
    if len(set(newCoordinates)) > 1 and self.t == 0:
        # if our division "succeeds"
        if random() < self.divisionProbability:
            # search for an empty space in which to divide
            while board[newY][newX] != EmptyCell(self.chars):
                newX, newY = choice(newCoordinates)
            board[newY][newX] = SimpleCell(
                self.chars,
                newX, newY,
walkType=\texttt{self.walkType},
tDivision=\texttt{self.tDivision},
radius=\texttt{self.radius},
densityRadius=\texttt{self.densityRadius},
densityThreshold=\texttt{self.densityThreshold},
divisionProbability=\texttt{self.divisionProbability}
)
board[newY][newX].moved = \texttt{True}
board[newY][newX].tDivision = \texttt{self.tDivision}
board[newY][newX].divisionProbability = \texttt{self.divisionProbability}

# performs a biased random walk when local population
# is above a certain threshold, and an unbiased one
# below that threshold
def thresholdBiasedRandomWalk(self, board, x, y):
    radius = \texttt{self.densityRadius}
    threshold = \texttt{self.densityThreshold}

    # get how many neighbors this cell has
    numNeighbors = \texttt{self.getNeighbors(board, x, y, radius=radius)}

    # compute the density of this local area
    density = \texttt{float(numNeighbors) / float((2 * radius + 1) ** 2)}

    # trigger a biased random walk at some density threshold
    if density >= threshold:
        \texttt{self.biasedRandomWalk(board, x, y)}
    else:
        \texttt{self.randomWalk(board, x, y)}

# performs the action for this cell at this time step
def nextState(self, board, x, y):
    # a time step has passed (don't divide on first time step)
    \texttt{self.t} = (\texttt{self.t} + \texttt{1}) \% \texttt{self.tDivision}

    # default to a random walk
    walk = \texttt{self.walks.get(self.walkType, self.walks['rw'])}
    \texttt{walk(board, x, y)}

\textit{Simulation.py}
from \texttt{copy} \import \texttt{deepcopy}
import \texttt{curses}
from \texttt{random} \import \texttt{randint}
from \texttt{numpy.random} \import \texttt{permutation}

\import \texttt{CellDefinitions} \import \texttt{EmptyCell, SimpleCell}

# defines the simulation which can be run to display things to screen or to simulate
results of simulation
class \textit{Simulation}():
    # render the simulation board to screen
    def render(self):
if self.stdscr is not None:
    for y, row in enumerate(self.board):
        row_str = ''.join(list(map(str, row)))
        self.stdscr.addstr(y, 0, row_str)

    self.stdscr.refresh()

# must be passed a curses stdscr
def __init__(self, stdscr, n=1000, rows=100, cols=100, initConditions=None, empty=' ', cell='.', walkType='rw', tDivision=32, radius=2, densityRadius=4, densityThreshold=0.1, divisionProbability=0):
    # initialize internal data structure
    if stdscr is not None:
        rows = curses.LINES - 1
        cols = curses.COLS

    # define the mapping of cell name to the corresponding character
    chars = {
        'empty': empty,
        'simple': cell
    }

    # initialize a board of empty cells
    self.board = [[EmptyCell(chars) for _ in range(cols)] for _ in range(rows)]

    if initConditions is None:
        initConditions = [(randint(0, rows - 1), randint(0, cols - 1)) for _ in range(n)]

    for y, x in initConditions:
        self.board[y][x] = SimpleCell(
            chars,
            x, y,
            walkType=walkType,
            tDivision=tDivision,
            radius=radius,
            densityRadius=densityRadius,
            densityThreshold=densityThreshold,
            divisionProbability=divisionProbability
        )

    # initialize display
    self.stdscr = stdscr
if self.stdscr is not None:
    self.stdscr.clear()
    self.render()

# steps to the next simulation state
def nextState(self):
    for y, row in permutation(list(enumerate(self.board))):
        for x, cell in permutation(list(enumerate(row))):
            cell.nextState(self.board, x, y)

    # reset the cells to having not moved
    for row in self.board:
        for cell in row:
            cell.moved = False

run.py
#!/usr/bin/env python3

import sys
import curses
from time import sleep
import csv
import argparse
from tqdm import tqdm
import numpy as np

from Simulation import Simulation
from helpers import parseInitialConditionsFile

class CapitalisedHelpFormatter(argparse.ArgumentDefaultsHelpFormatter):
    def add_usage(self, usage, actions, groups, prefix=None):
        if prefix is None:
            prefix = 'Usage: '
        return super(CapitalisedHelpFormatter, self).add_usage(usage, actions, groups, prefix)

ap = argparse.ArgumentParser(formatter_class=CapitalisedHelpFormatter)
ap.add_argument(
    '-n',
    type=int,
    default=1000,
    help='the number of cells to randomly add'
)
ap.add_argument(
    '-d',
    '--delay',
    type=float,
    default=None,
    help='the delay in seconds between iterations of the simulation'
ap.add_argument(  
    '--cell',  
    type=str,  
    default='.',  
    help='the ASCII character to use to represent cells'
)

ap.add_argument(  
    '--empty',  
    type=str,  
    default=' ',  
    help='the ASCII character to use to represent empty spaces'
)

ap.add_argument(  
    '--init',  
    type=str,  
    default=None,  
    help='the path of a file containing a coordinate "x y" on each line representing the starting position of a cell'
)

ap.add_argument(  
    '--walk-type',  
    type=str,  
    default='rw',  
    help='the type of walk to perform, choices are ["rw", "brw", "tbrw"]'
)

ap.add_argument(  
    '--division-time',  
    type=int,  
    default=32,  
    help='the number of time steps before cells attempt to divide'
)

ap.add_argument(  
    '--radius',  
    type=int,  
    default=2,  
    help='the radius to search when cells are biased towards more populated areas'
)

ap.add_argument(  
    '--density-radius',  
    type=int,  
    default=4,  
    help='the radius to search when determining the cell density around a cell'
)

ap.add_argument(  
    '--division-time',  
    type=int,  
    default=32,  
    help='the number of time steps before cells attempt to divide'
)
'--density-threshold',
    type=float,
    default=0.1,
    help='the threshold above which a biased random walk should be performed, and below which a random walk will be performed'
    )

ap.add_argument(
    '--division-probability',
    type=float,
    default=0,
    help='the probability that an attempted cell division succeeds. set to 0 for no cell division'
    )

stats_group = ap.add_argument_group(
    'Stats Arguments',
    'these only apply when the --stats flag is used and can be useful for collecting data on different conditions. Note: turns off simulation display and minimally logs to console'
    )
stats_group.add_argument(
    '--stats',
    action='store_true',
    help='if used, the simulation will not be displayed to console, and all stats options will apply'
    )
stats_group.add_argument(
    '--rows',
    type=int,
    default=100,
    help='the number of rows to simulate having'
    )
stats_group.add_argument(
    '--cols',
    type=int,
    default=100,
    help='the number of columns to simulate having'
    )
stats_group.add_argument(
    '--iters',
    type=int,
    default=1,
    help='the number of simulations to run'
    )
stats_group.add_argument(
    '--timesteps',
    type=int,
    default=1000,
    help='the number of timesteps per simulation'
    )
stats_group.add_argument(
    '--sampling',
    }
    type=int,
default=10,
    help='the number of timesteps after which to sample - higher numbers means less
    sampling'
    )
stats_group.add_argument(  
    '-o',  
    '--outfile',  
type=str,
default='statistics.txt',
    help='the output file to log collected statistics to'
    )

args = ap.parse_args()

def getCentroidPosition(board):
    positions = list()
    for row in board:
        for cell in row:
            # TODO: broken abstraction barrier, but you have no time!
            if cell.cellType == 'simple':
                positions.append(list(cell.coordinates))

    return np.mean(positions, axis=0)

def main(stdscr, stats=False):
    initialConditions = None
    if args.init is not None:
        initialConditions = parseInitialConditionsFile(args.init)

    if stats:
        keys = [
            'rows',  
            'columns',  
            'area',  
            'initialCount',  
            'iteration',  
            'timestep',  
            # 'cellCount',  
            # 'cellCoverage',  
            'centroidPositionX',  
            'centroidPositionY'
        ]

        with open(args.outfile, 'w') as datafile:
            writer = csv.DictWriter(datafile, [key for key in keys])
            writer.writeheader()

            for i in tqdm(range(args.iters)):
                # capture all of the data for this iteration
                data = {}
                simulation = Simulation(
                    None,
                    n=args.n,
initConditions=initialConditions,
rows=args.rows, cols=args.cols,
walkType=args.walk_type,
tDivision=args.division_time,
radius=args.radius,
densityRadius=args.density_radius,
densityThreshold=args.density_threshold,
divisionProbability=args.division_probability
)

for t in tqdm(range(args.timesteps)):
simulation.nextState()

if t % args.sampling == 0:
    # setup and conditions bookkeeping
    data['rows'] = args.rows
    data['columns'] = args.cols
    data['area'] = args.rows * args.cols
    data['initialCount'] = len(initialConditions) if initialConditions is not None else args.n

    # other bookkeeping
    data['iteration'] = i
    data['timestep'] = t

    # data of interest
    # data['cellCount'] = np.sum(np.array(simulation.board,
dtype=np.int32))
    data['cellCoverage'] = float(data['cellCount']) / float(data[t]['area'])

    # get positions on the centroids
    data['centroidPositionX'], data['centroidPositionY'] =
    getCentroidPosition(simulation.board)

    writer.writerow(data)

else:
simulation = Simulation(
    stdscr,
    n=args.n,
    initialConditions=initialConditions,
    empty=args.empty, cell=args.cell,
    walkType=args.walk_type,
tDivision=args.division_time,
    radius=args.radius,
densityRadius=args.density_radius,
densityThreshold=args.density_threshold,
divisionProbability=args.division_probability
)

    # run the simulation indefinitely
    while True:
if args.delay is not None:
    sleep(args.delay)

simulation.nextState()
simulation.render()

if __name__ == '__main__':
    try:
        if args.walk_type not in ['rw', 'brw', 'tbrw']:
            print(f'''{args.walk_type} not found, defaulting to random walk''',
                  file=sys.stderr)

        if args.stats:
            main(None, stats=args.stats)
        else:
            curses.wrapper(main)

    except KeyboardInterrupt:
        print('User aborted - exiting')

generate_island.py
import random
import argparse
import math

class CapitalisedHelpFormatter(argparse.ArgumentDefaultsHelpFormatter):
    def add_usage(self, usage, actions, groups, prefix=None):
        if prefix is None:
            prefix = 'Usage: '
        return super(CapitalisedHelpFormatter, self).add_usage(usage, actions, groups, prefix)

ap = argparse.ArgumentParser(formatter_class=CapitalisedHelpFormatter)
ap.add_argument(  
    'cols',  
    type=int,  
    help='the number of columns available'
)
ap.add_argument(  
    'rows',  
    type=int,  
    help='the number of rows available'
)
ap.add_argument(  
    '-n',  
    default=100,  
    type=int,
help='the number of cells to include in the island'
)
ap.add_argument(
    '-o',
    '--outfile',
    type=str,
default='island.txt',
    help='the output file, which will contain a position per line in the format "x y"
)

args = ap.parse_args()
def main():
    if (args.rows * args.cols < args.n):
        print('Insufficient space (rows * cols < n)')
        exit(1)

centerX, centerY = args.cols // 2, args.rows // 2
radius = math.sqrt(args.n / math.pi)

with open(args.outfile, 'w') as outfile:
    for x in range(round(centerX - radius), round(centerX + radius + 1)):
        for y in range(round(centerY - radius), round(centerY + radius + 1)):
            if math.sqrt((x - centerX) ** 2 + (y - centerY) ** 2) < radius:
                outfile.write(f"{x} {y}\n"

    print('Done')

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('User aborted - exiting')
return super(CapitalisedHelpFormatter, self).add_usage(usage, actions, groups, prefix)

ap = argparse.ArgumentParser(formatter_class=CapitalisedHelpFormatter)

ap.add_argument('datafile',
    type=str,
    help='the datafile to analyze - should be a csv with at least centroidPositionX, centroidPositionY, and iteration columns.'
)

ap.add_argument('--sqrt',
    action='store_true',
    help='if passed, will plot against the sqrt of timestep'
)

ap.add_argument('--save',
    action='store_true',
    help='if passed, will save figures'
)

args = ap.parse_args()

if __name__ == '__main__':
    # raw data as collected
    data = pd.read_csv(args.datafile)

    figname = os.path.basename(args.datafile).split('.')[0]

    sns.set(rc={'figure.figsize':(11.7, 11.7)})

    plt.figure()
    plt.title(f"Selected Centroid Positions - {figname}", fontsize=32)
    sns.lineplot(
        x='centroidPositionX', y='centroidPositionY',
        hue='iteration',
        data=data.where(data['iteration'] < 3),
        palette=sns.color_palette('dark', 3),
    )

    plt.xlabel('x (unit)')
    plt.ylabel('y (unit)')

    if args.save:
        plt.savefig(f'centroid_positions_{figname}.png')

    plt.show()

    # to be used to linearize random walks
    if args.sqrt:
        data['timestep'] = np.sqrt(data['timestep'])
# a time series where each row is a different trial
seriesData = data.pivot(index='iteration', columns='timestep',
values=['centroidPositionX', 'centroidPositionY'])

# center X and Y on their starting positions
centeredX = seriesData['centroidPositionX'].sub(seriesData['centroidPositionX'][0], axis=0)
centeredY = seriesData['centroidPositionY'].sub(seriesData['centroidPositionY'][0], axis=0)

distance = centeredX.pow(2).add(centeredY.pow(2))

rmsd = distance.mean().pow(0.5)

plt.figure()
plt.title(f"RMSD - {figname}", fontsize=32)
sns.lineplot(
    data=rmsd,
    label='simulated data',
    color='black'
)

slope, intercept, r_value, p_value, std_err = linregress(rmsd.index, rmsd)
print(f"Linear: {r_value}, p-value: {p_value}")

sns.lineplot(
    x=rmsd.index,
    y=[i * slope + intercept for i in rmsd.index],
    label='linear regression',
    color='red'
)

if args.sqrt:
    plt.xlabel('$\sqrt{timestep}$')
else:
    plt.xlabel('timestep')
plt.ylabel('RMSD (unit)')

slope, intercept, r_value, p_value, std_err = linregress(np.sqrt(rmsd.index), rmsd)
print(f"sqrt(t): {r_value}, p-value: {p_value}")

if args.save:
    if args.sqrt:
        plt.savefig(f'{figname}_rmsd_sqrtt_linreg.png')
    else:
        plt.savefig(f'{figname}_rmsd_linreg.png')

plt.show()