4-17-2018

Lattice-Gas Cellular Automata In Modeling Biological Pattern Formation

Gizem Yuce
Illinois State University, gzm.yuce@gmail.com

LATTICE-GAS CELLULAR AUTOMATA IN MODELING BIOLOGICAL PATTERN FORMATION

GIZEM YUCE

65 Pages

Cellular automaton models (the most famous of which is arguably the Game of Life) are particularly suited for studying systems in the physical/biological world that evolve in discrete units and discrete time steps, depending on no master plan but instead on local interactions with simple rules. As a result of such interactions, patterns of behavior or structure with unexpected complexity may emerge. We can count termite nests, animal coat patterns, and bird flocking among such patterns. Lattice-gas cellular automaton (LGCA) models take the Game-of-Life type models one step further and introduce additional features, where several interacting units with interior structure (such as velocity) may be present in the same cell. Under familiar physical rules, which may be chosen from conservation laws, statistical mechanics, and quantum mechanics, the units (particles) of one or more species interact with each other, often according to a stochastic model, before propagating. In this study, we make use of an "adhesive" LGCA model on a hexagonal lattice and the agent-based NetLogo software to simulate the conditions under which a biological pattern, similar to a mammal coat pattern but more frequently observed in real life in the developmental stage of smaller organisms, emerges. By varying two parameters, namely the "adhesive strength" of the pigment cells and the particle concentration, we observe a line of bifurcation points where patterns change from stable (homogeneously distributed and relatively unchanging in time) to unstable (increasingly growing patches with wide spaces in between). The dependence on these parameters is very similar to the pattern displayed in [4] and [6] for the same model on a square lattice.

KEYWORDS: Cellular Automata, Lattice-Gas Cellular Automata, Agent-Based Modeling, NetLogo

LATTICE-GAS CELLULAR AUTOMATA IN MODELING BIOLOGICAL PATTERN FORMATION

GIZEM YUCE

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Mathematics

ILLINOIS STATE UNIVERSITY

2018

LATTICE-GAS CELLULAR AUTOMATA IN MODELING BIOLOGICAL PATTERN FORMATION

GIZEM YUCE

ACKNOWLEDGMENTS

First of all, I am deeply grateful to my thesis advisors Prof. Fusun Akman and Prof. Olcay Akman for introducing me to Mathematical Biology and hence making this thesis possible. I would like to thank them not only for their vast knowledge and expertise in the field, but also for their determined support, kindness, and constant guidance through every step of this thesis. I would also like to thank my thesis committee members, Prof. Unal Ufuktepe and Prof. Nathan Mortimer, for their time and effort in reviewing this thesis. Next, my thanks go to Geoffrey Ower for his substantial contributions in writing the NetLogo code; I could not have learned NetLogo in such a short time without his help. Finally, my deepest gratitude goes to my family and friends, for their unflinching support, love and blessings. When things look hopeless, it is to them I turn and find my strength.

G. Y.

CONTENTS

FIGURES

CHAPTER I: INTRODUCTION

1.1 Background of Biological Pattern Formation

Biological pattern formation is practically synonymous with development of organisms. “In the beginning of embryological development, all cells are identical [6].” Hence, the genome itself is not sufficient to explain differentiation and pattern formation. Interactions among cells, and between cells and their environment, are responsible for the developmental stage. A complex pattern emerges from a collection of similar cells and bottom-up, mindless interactions at the mesoscopic level selected by nature.

Morphogenesis (“beginning of the shape”) is the biological process that shapes an organism by differentiation of cells, tissues, and organs, and the eventual development of organ systems. We can also apply this term to the development of unicellular organisms. Let us briefly go over the evolution of various explanations of the mechanisms responsible for morphogenesis, following Deutsch and Dormann [6]. The authors emphasize that both the space and time components of pattern formation need to be accounted for. Plato and his followers, for example, regarded all forms (biological or not) as unchanging and preformed, uncoiling and following their destiny in a predetermined path. No new form was allowed to come to existence in this ideology. By contrast, the Aristotelian school accepted the necessity of epigenetic changes (those that involve heritable gene expression without any change to the genome, as the modern biologists would say) in new pattern formation, and adopted a dynamic world view. The modern understanding of morphogenesis involves ontogenetic changes (development from embryo to adult form) in the small time-scale as well as phylogenetic changes (evolution of species) in the long time-scale. Although Darwin’s “pangenesis” theory of heredity had many incorrect components, *The Origin of Species* (1859) created the impetus that challenged and changed the dogmas.

Today, there are several accepted mechanisms that explain different types of pattern formation, including preformation, optimization, and self-organization. In fact, it is often necessary to combine more than one characteristic to model a particular example. Just like physical theories, it is important to choose a model that best approximates the kind and scale of system under consideration. In this paper, we will examine cellular automata in general and adhesive lattice-gas cellular automata in particular. The latter is an example of an agent-based, bottom-up, discrete model that relies on stochastic cell-cell interactions and Boltzmann-type physics to explain self-organization of patterns at cellular level. One particular application of this model involves aggregation via cadherin expression, where motile cells seem to follow a gradient field of cadherin expressed by their neighbors.

1.2 Mathematical Modeling of Pattern Formation: Famous Models

As Deutsch and Dormann [6] admit, there is no unified theory of morphogenesis today, certainly not on a par with Darwin's theory of natural selection. We understand the existing models to be a patchwork of ideas that (individually or in groups) work well for particular organisms, which makes morphogenesis one of the most exciting subfields of biology. Is there an organizing principle, and are we going to know in our lifetime? As summarized in [6], there are only so many things cells do in morphogenesis: they change shape, grow, differentiate, die, and migrate. Most importantly, cells do not act independent of each other. It has been long known that cells interact with each other over short and long distances; they also react to their chemical environment, which is partially formed by activities of nearby cells. Direct (local) cell interactions include adhesion, alignment, contact guidance/contact inhibition, and haptotaxis (directional motility or outgrowth of cells). Indirect (long-distance) cell interactions are understood to be guided by mechanical forces such as bending as well as chemical signals. For example, chemotaxis explains how cells orient themselves towards the local maxima of a chemical gradient field [6]. Although we think of the particles in our simulations as pigment cells, the model certainly applies to any system of cells that express certain levels of chemicals to interact with each other and migrate as a result. It would be interesting to generalize this model to include more than one species of cells (like the oocyte and nurse cells [13] in Section 1.4), and to asymmetric adhesion rules.

On the whole, a morphogenetic system is one that exhibits self-organization. It is the job of mathematical modeling to describe what assumptions and simplifications to make when choosing the essential players (cells and interactions) so that real biological systems can be re-created in such a way that the macroscopic patterns that they form can be plausibly explained.

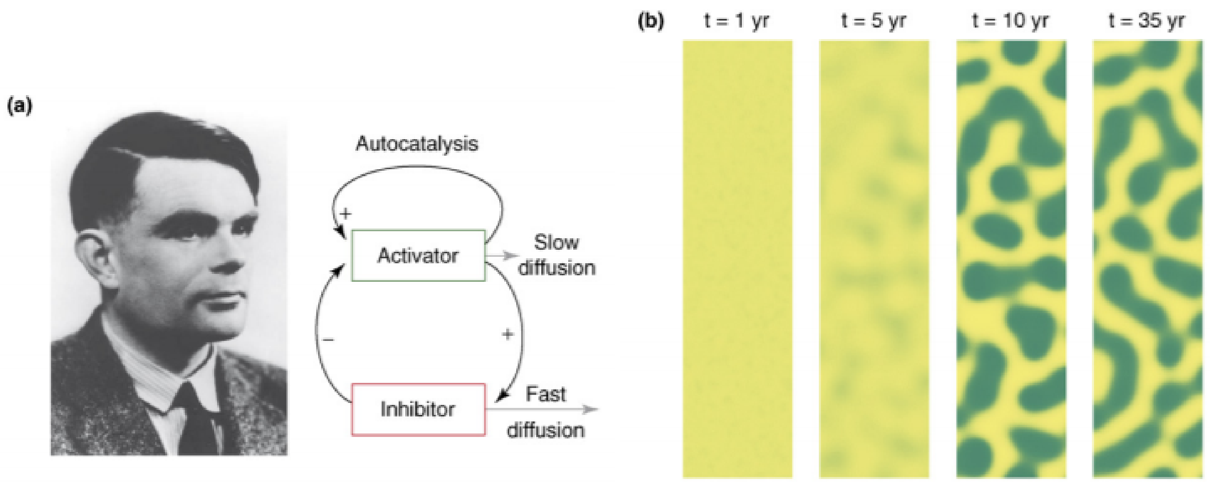


Figure 1: Alan Turing & the system of activator and inhibitor, and pattern production for vegetation biomass by the arid ecosystem model

A detailed history of mathematical modeling of pattern formation is given in [6]. Since we will explore cellular automata in detail later, let us just mention one of the pioneering works, by Alan Turing, that is widely recognized by biologists. In addition to his ground-breaking work in the theory of computation and his cryptological work during World War II, Turing was also responsible for the revolutionary reaction-diffusion model for pattern formation [29]. The novel idea behind his work was that diffusion, which was accepted to be a stabilizing process, could actually destabilize a homogeneous initial condition and lead to a heterogeneous pattern formation in a reactive system. As explained by Rietkerk and Van de Koppel [24], short-distance positive feedback (amorphous aggregation) and long-distance negative feedback, or, "scale-dependent feedback," has been shown to be responsible for a variety of pattern formations. Turing's system consists of two chemical substances, an activator that creates more of itself, and an inhibitor that is also created by the activator. These two chemicals provide positive and negative feedback respectively, and an unexpected "Turing pattern" emerges as they diffuse through tissue at different rates. Fig. 1a [24] provides a sketch of his system, whereas Fig. 1b [24] shows pattern formation as a result of the Turing mechanism in an arid ecosystem, first published in Rietkerk et al. [23]

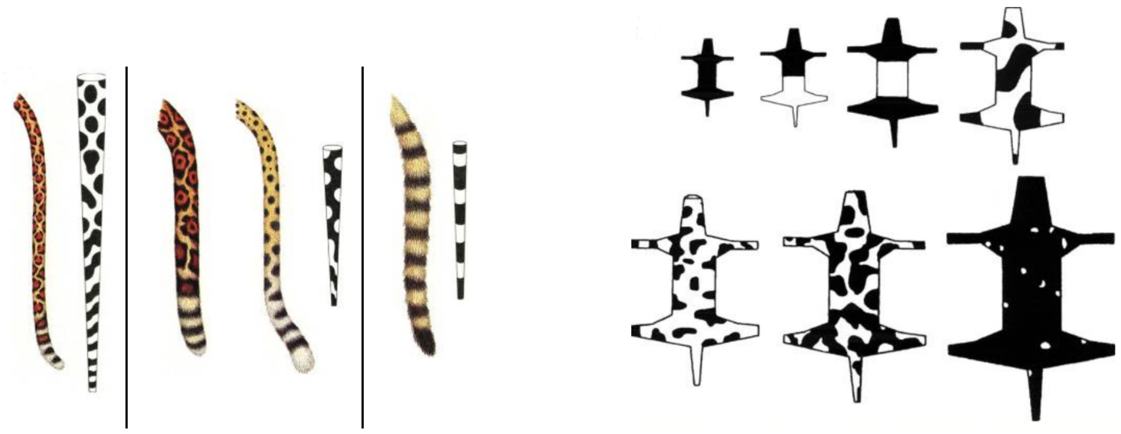


Figure 2: Examples of melanin concentration and numerical simulations of the reaction-diffusion model

(Amazingly, Turing's paper [28] included hand-drawn figures, as the Father of Computing did not have a modern computer.)

Turing not only published one of the most important papers in theoretical biology (according to Murray [20]), but also inspired generations of biologists to conduct real-life experiments as well as simulations based on his ideas. He also coined the term "morphogen" for agents of morphological change. In one of the better-known later articles (How the Leopard Gets Its Spots [20]), Murray explains how his work improved Turing's reaction-diffusion idea and shows how mammalian coat patterns could be generated. On the left of Fig. 2 [20], tapering cylinders of different widths successfully imitate tails of the leopard (left), the jaguar and the cheetah (middle), and the genet (right) respectively. On the right, his simulations show that very small and very large animals like the mouse and the elephant are likely to have uniformly colored coats, whereas middle-sized animals like the leopard are predisposed to have patterned coats. Murray showed, remarkably, that his mechanism allows a spotted animal to have a striped tail but a striped animal may not have a spotted tail. In reaction-diffusion models, indeed, in all positive-negative feedback models, the importance of scale cannot be overemphasized. Cellular-size interactions lead to patterns that are much larger than the individual cells.

1.3 General Introduction to Cellular Automata

Cellular automata are the best known models of *self organization* in physical, chemical, and biological systems [32]. True self organization happens when individuals in a large group work simultaneously without an external "boss," just employing a few local rules by observing the "states" of themselves and their neighbors. The individuals may be independent organisms such as fish, ants, or birds, or subunits of organisms, such as cells [5]. Eventually, complex behavior (such as flocking) or structures (such as termite mounds, sea shell patterns, or coat patterns) emerge on the macroscopic level.

John von Neumann (1903-1957) and Stanislaw Ulam (1909-1984) planted the seeds of the cellular automaton (CA) mechanism by proposing an abstract model of self-reproduction that creates copies of itself [6]. Further generalizations as well as simplifications of their model led to the CA mechanisms we use today. With powerful computing support, there are several modern applications for CAs in biology, chemistry, physics, and sociology studies [18]. Once CAs became better known in the scientific community, the British mathematician John Horton Conway introduced the Game of Life in 1970 as a two-dimensional deterministic cellular automaton based on a square grid. An initial array of white and black "cells" represent dead and alive individuals respectively [3]. In synchronous steps, the fate of each cell is sealed by its own state and its neighbors that simplify von Neumann's machine. In essence, the idea is a cellular automaton in which a finite number of deterministic rules are applied according to the states of the cells and of their eight neighbors': an empty cell comes to life when exactly three of its neighbors are alive; a live cell dies of crowding (more than three neighbors) or loneliness (less than two neighbors). The emergent patterns of populations have fascinated observers ever since. Populations may die out, arrive at some steady -unchanging- state, settle into periodic behavior (e.g., "blinkers"), or march across the screen in hordes (e.g., "gliders"). Fig. 3 [12], shows the several patterns generated by the

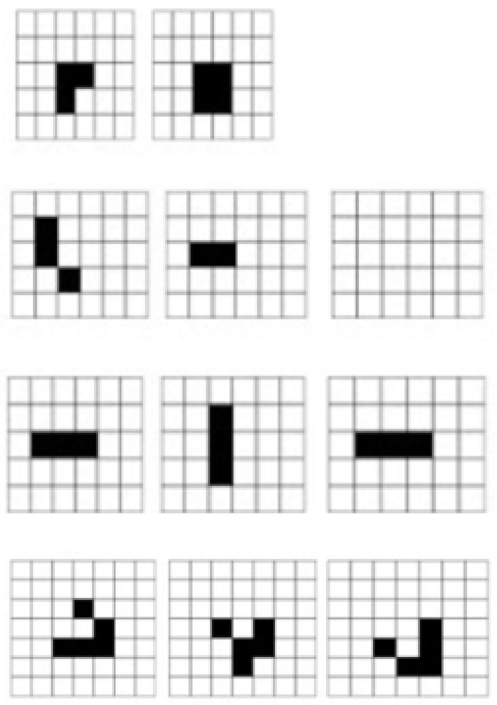


Figure 3: Examples for possible patterns in the Game of Life

Game of Life. The first pattern from the top is a stable block; the second one shows an extinction; the third is a 2-cycle (called a blinker); and the fourth is the famous glider that moves across the grid in an undulating motion.

Although it is possible to set cellular automata in a space of arbitrary dimension, studying them in one- or two-dimensional space is mathematically easier. In the one-dimensional case (See Fig. 4) [6], there is a large number of adjacent boxes in an array with specific boundary conditions such as periodic, reflecting, or fixed. To see how the CA evolves, we need discrete moments in time, $t = 0, 1, 2, 3, \ldots$, where $t = 0$ denotes the initial time, and a local transition function for the change of state for each cell and its neighbors according to the set of rules defined. In Fig.4 [6], there are three different boundary types for a one-dimensional CA. The cells on the left and right of the dashed lines define the left and right closest neighbor cells for the boundaries. For fixed values of boundaries, gray cells are used. After deciding what to do at the boundary, we consider

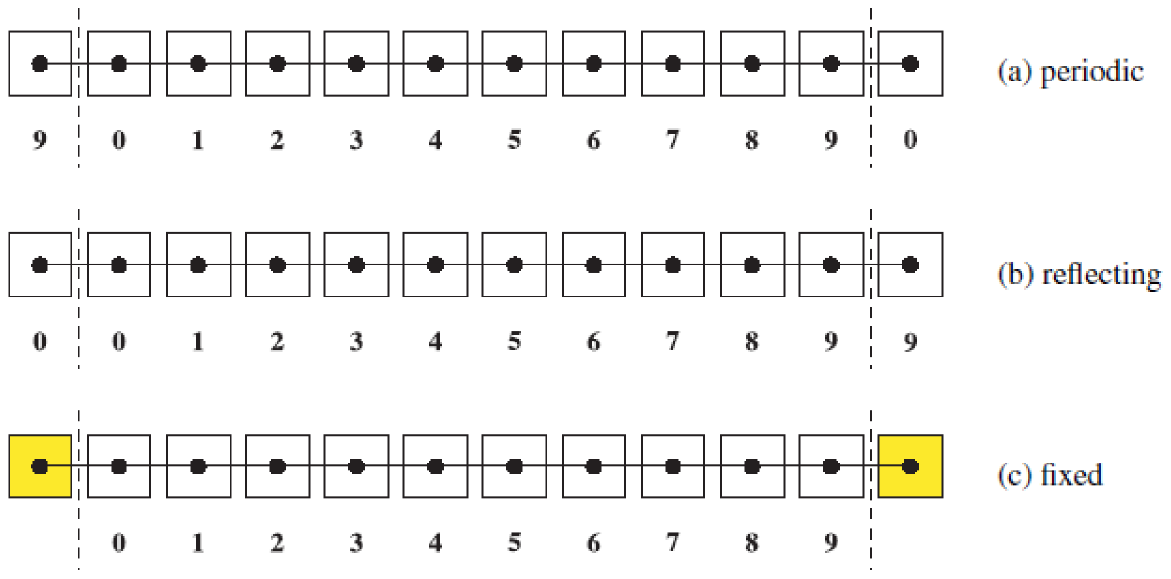


Figure 4: One-dimensional lattice configurations

the local transition function, which may be probabilistic or deterministic.

If the same set of rules are applied to all cell states at the same time, then we say that CA is synchronous; if cells are allowed to change their states at different times, then we call the CA asynchronous. All of these notions readily generalize to higher dimensions [1].

In this project, our goal is to understand the so-called *lattice-gas cellular automata (LGCA)* mechanisms and analyze their contributions to biological pattern formation. This line of investigation of lattice-gas cellular automata methods was proposed in paper of Hardy et al. [15]. What is the distinction between a plain CA and a lattice-gas CA? In a CA, there is only one set of rules to be applied at each time step. By contrast, LGCA have separate "interaction" and "propagation" rules, based more on physical rather than biological processes. In the first, static, phase of time step, particles "collide" and change direction, or are otherwise affected, in the cell they are already located in. The second, dynamic, phase of the time step consists of "propagating" these affected particles in the direction they are facing. The complicated but more realistic rules allow LGCAs to have many more interesting emergent properties than ordinary

CAs.

In the first lattice gas cellular automata models, gas was modeled as a group of particles with continuous position moving in continuous time, with a discrete set of velocities. Later, LGCAs were set in discrete space and time, in addition to having discrete velocity values. Hardy et al. [15] studied the first real LGCA in its simplest form, which was a cellular automaton on a two-dimensional square lattice [9]. This automaton is known as HPP. This model contains a fixed set of four vectors that connect each cell to its neighbors, so that at each node, there are four cells that are designated as "closest neighbors." One of the crucial points here is that each cell can have at most one particle within the same "velocity channel," which is valid for all modern LGCA. This is simply the "Pauli exclusion principle" that says matter particles cannot occupy the same state. Also, all particles are identical (another idea from quantum physics) and the development in time is deterministic [31].

In a possible regular two-dimensional lattice, "square" is not the only choice for the shape of a cell. For a polygon that can tile the plane, its interior angle should be a factor of 360 degrees. According to this fact, there are only three regular polygons with this condition: equilateral triangles, squares, and regular hexagons (Fig. 5) [6]. In this figure, cells and nodes (dots at the center of each cell) in the three possible two-dimensional lattices are shown. One of the important advantages of using a hexagonal automaton is the additional variation that it accommodates compared to the square or triangle ones. Due to the larger number of edges, hexagonal lattices are better approximations to the continuous plane than the others. There are "velocity channels" that serve as both particle holders and direction inside cells, namely north-east, east, south-east, south-west, west, and north-west [19]. In this set-up, we randomly assign zero to six particles modeling biological pigment cells to each cell, no more than one occupying the same channel. As we have mentioned before, the static and dynamic transition rules then take over and move the particles around, conserving the total

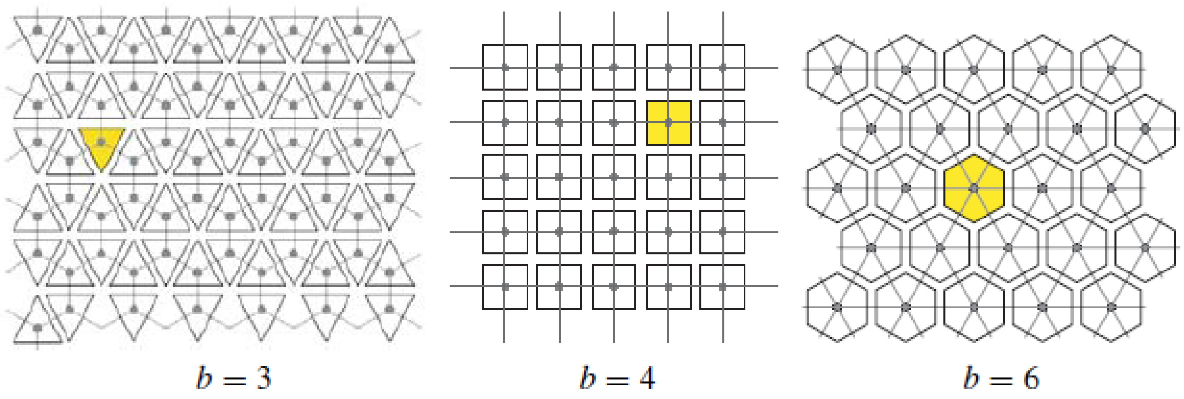


Figure 5: Two-dimensional lattice configurations

number (yet another physical concept).

We will specifically study local interaction rules that create adhesive patterns. Therefore, the full name of our model is “Adhesive lattice-gas cellular automaton” (ALGCA). The overall pattern, if any, is developed by a combination of two factors, namely diffusion as the dynamic propagation rule, and sticking as the static local interaction rule. The adhesion/sticking strength is controlled by a positive parameter α. In our model, particles (pigment cells) directly and locally interact with each other without any intermediaries such as chemicals, outer signals, etc., during the local interaction rule I, then get diffused by the propagation rule P. Each time step contains a back-to-back application of operators, $P \circ I$, synchronously at all hexagons.

CA are particular examples of “agent-based models” (ABMs), where agents are the images of real-world individuals, and they have certain characteristics like being active, identifiable, autonomous, and capable of making independent decisions. This technique is useful to simulate complex systems by deterministic, stochastic, and adaptive rules, and may lead a disordered system to organize itself into an ordered configuration. ABMs allow either synchronous or non-synchronous interactions of agents with other agents and with their environments. We will talk about ABMs in depth later.

Overall, our goal is to examine the large-scale behavior of an adhesive lattice-gas cellular automaton that simulates pattern formation due to cell-cell interactions. Here, the fundamental point is that every single cell responds to the signals from its immediate environment. The static, local interaction rule I depends on two parameters, α (the adhesive strength) and $\bar{\rho}$ (fixed particle density). Depending on the values of these parameters, we will observe either homogeneous, stable behavior (no discernible pattern), or unstable behavior, resulting in aggregating and evolving patterns. The bifurcation line will be clearly visible.

1.4 Biological Examples of Adhesive Aggregation in the Literature

From [6]: "Stable cell interactions are needed to maintain the structural integrity of tissues, and dynamic changes in cell adhesion are required in the morphogenesis of developing tissues. Stable interactions actually depend on active adhesion mechanisms that are very similar to those involved in tissue dynamics. Adhesion mechanisms are highly regulated during tissue morphogenesis and are intimately coupled to cell migration processes. In particular, molecules of the cadherin and integrin families are involved in the control of cell movement. Cadherin-mediated cell compaction and cellular rearrangements may be analogous to integrin-mediated cell spreading and motility on the extracellular matrix."

The first *in vivo* example about cell sorting in *Drosophila* that depends on affinity is provided by Godt and Tepass [13]. In follicles of the fruitfly ovary, there are two types of germline cells (which are evolved from earlier cells and show continuity of consecutive generations): the oocyte (egg cell) and its attendant nurse cells (helper cells for food and stability to their neighboring cells). As a whole, it consists of these two germline cells and a surrounding layer of follicle cells. The oocyte is settled next to the follicle cells at the posterior pole of the follicle. Here, the point is that even though all germline and follicle cells are expressing E-cadherin, the majority comes from posterior follicle cells and the oocyte. Therefore, it is possible that for this specific positioning, the oocyte considers cells with higher levels of E-cadherin. By removing E-cadherin from the oocyte and nurse cells, it is shown that the oocyte adapts to random position. This indicates that the adhesion between germline and follicle cells is fundamental for positioning.

In another recent and similar study, Gonzalez and Johnston [14] studied anterior-posterior polarity in $Drosophila$ by the oocyte and the nurse cells and the surrounding layer of follicle cells. The main focus was finding the mechanism of oocyte

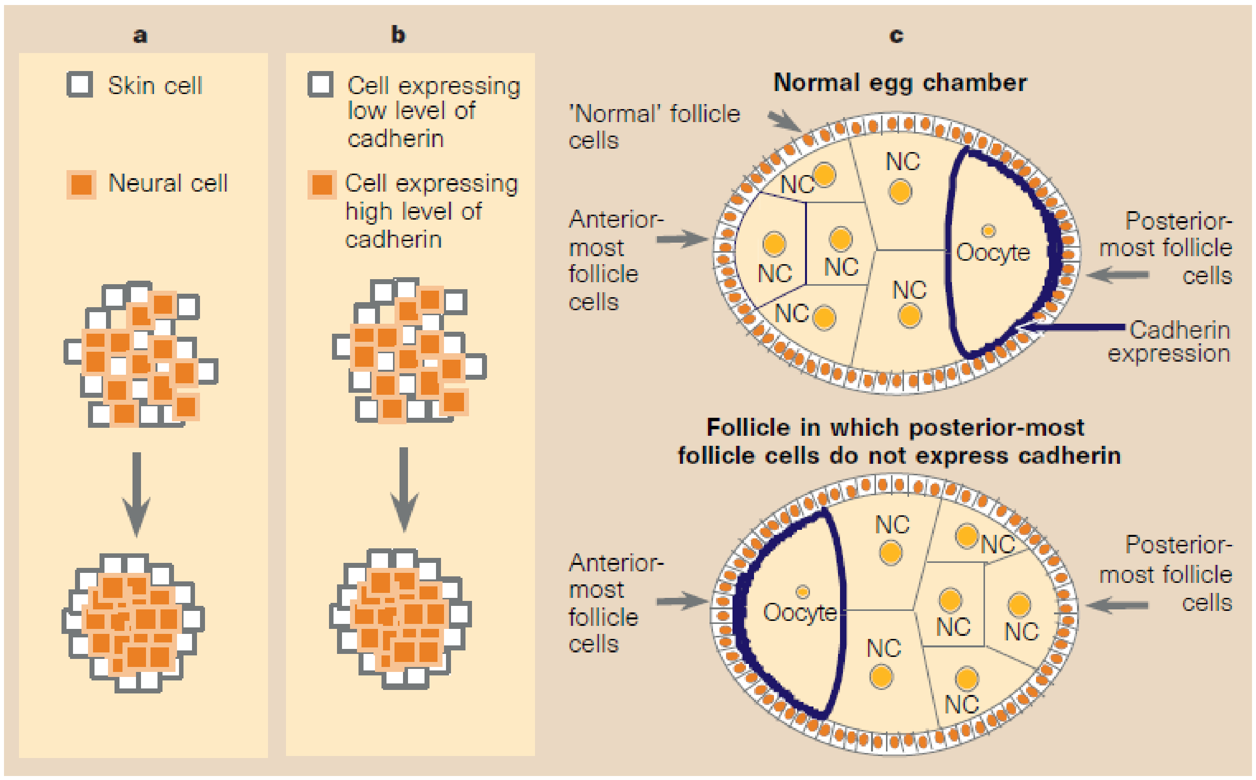

a
Skin cell
Neural cell
b
Cell expressing low level of cadherin
Cell expressing high level of cadherin
c
Normal egg chamber
'Normal' follicle cells
Anterior-most follicle cells
NC
Oocyte
Posterior-most follicle cells
Cadherin expression
Follicle in which posterior-most follicle cells do not express cadherin
Anterior-most follicle cells
Posterior-most follicle cells

Figure 6: Cell sorting into separate aggregates during developmental stage

that helps it to reach the posterior of germline cyst and the role of cadherin-dependent adhesion for this movement. Different from [13], they showed that the anterior-posterior polarity occurs in two adhesive steps; homotypic adhesion between the germ cells for the placement of the two pro-oocytes (cells that have four connection canals that are two more than regular cell in the cyst) where they can contact with the posterior follicle cells, and heterotypic adhesion between oocyte (the one is selected from two pro-oocytes) and those follicle cells. Their results as consistent with [13] revealed that the incident of polarity is based on cadherin-dependent adhesion.

Peifer [21] summarized the cell sorting mechanism, which can be based on different levels of cadherins. The relevant studies of recent times which suggest that differential expression of cadherins is the determinant mechanism for certain morphogenetic events in the whole animal world, are mentioned in [21]. Figure 6a [21], shows an *in vitro* example, sorting of cells due to their types. It is clear that neural cells

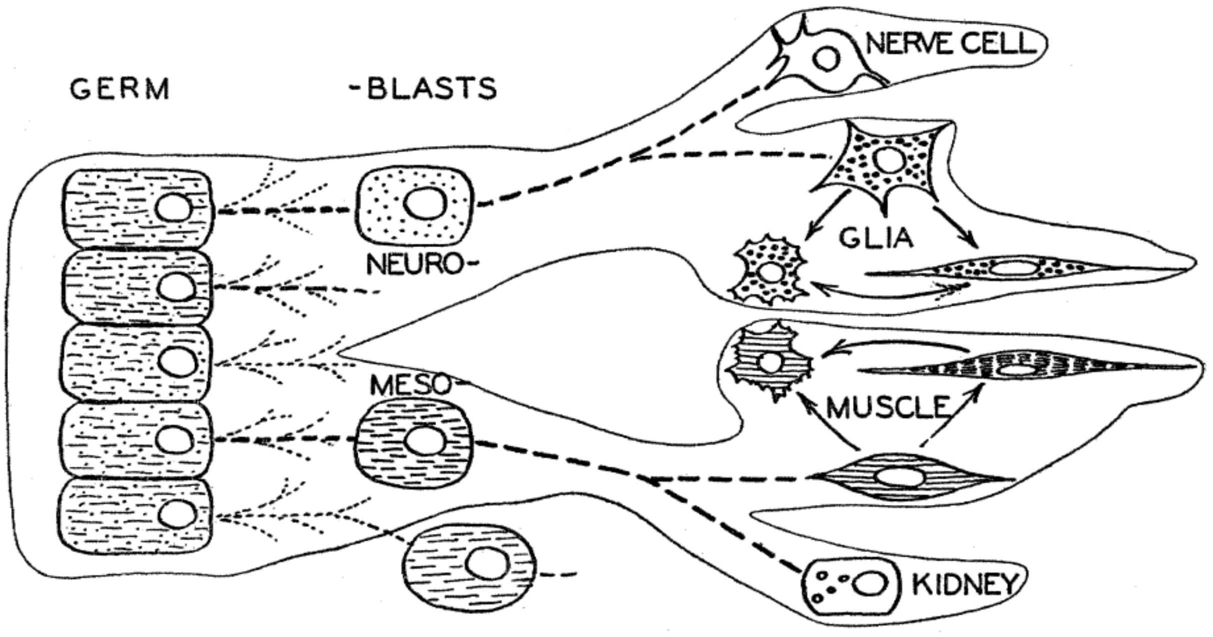


Figure 7: Schematic drawing of four cell strains

are organizing in such a way that they can adhere to other neural cells by expression of different cadherins. Fig. 6b [21], shows that cells with higher level of cadherin are sorting to stay together. In Fig. 6c [21], the expression of higher levels of E-cadherin in the oocyte and posterior follicle cells than nurse cells (NC) is shown. This is demonstrated also in [13] and [14], and it appears that expression of E-cadherin ensures the positioning of the oocyte at the posterior pole.

Weiss [29] discussed main topics for morphogenesis with their current problems in 1950. In particular, differentiation and its criteria examined in detail. According to this study, differentiation is defined as “the gradual elaboration of new chemical systems and compounds not previously present as such, presumably by gradual transformation of the patterns according to which synthesis of the protoplasmic (the cytoplasm and nucleus of a cell.) compounds occurs. This transformation of basic protoplasm takes divergent courses in different cell strains, producing lines which become increasingly dissimilar in their biochemical and morphological constitution as development proceeds.” As end products, secretion bodies, pigment granules, etc., are signs of cellular

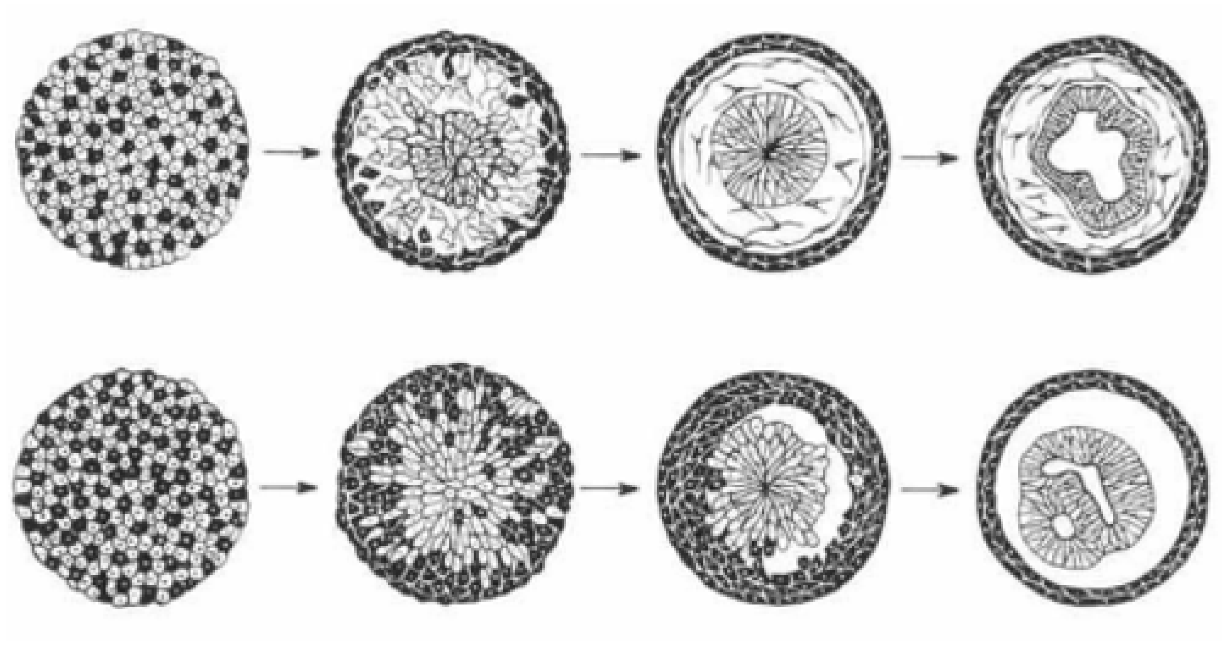


Figure 8: Diagrams for stages of composite reaggregates

manufacturing processes which differ due to their respective formative mechanisms. Figure 7 [29], shows how cells of the same constitution that originate from the same germ layer can advance to different cell strains. These strains keep splitting until all of the specialized cell forms evolve.

In 1955, Townes and Holtfreter [27] performed an experiment by using eggs of several species of Amphibia, which is considered as the beginning of modern morphogenesis analysis. The different cell types of the embryo showed tissue-specific tendencies -for every time- by moving either to the center of cell or to the opposite direction when cell types were mixed. Then, different adhesion levels were monitored in the moving cells. Overall, as a result of adhesive effects, segregations and recombinations of tissues and individual cells occurred. With this study, the concepts of cell recognition and selective affinity of cells and tissues were firmly established. Figure 8 [27], denotes the successive stages for reaggregation of cells from the neural fold with the scattered epidermal (black-cells that are forming the outermost portion of the skin) and mesodermal (white- the middle layer among three primary germ layers) cells (first

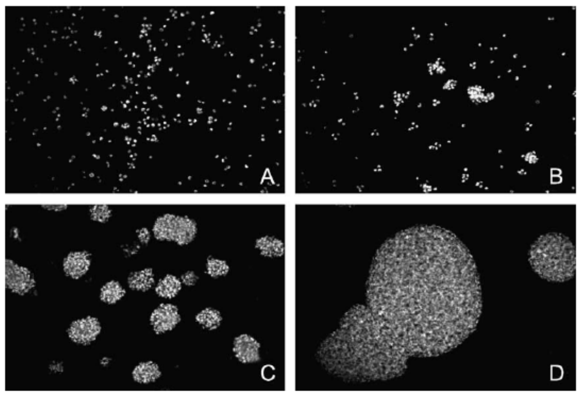


Figure 9: Aggregation of cell lines with increasing cadherin expression levels

diagram), and only epidermal and mesodermal cells (second diagram). When there are only mesodermal and epidermal cells, the latter tend to regroup at the outer edge, while the mesoderm cells move toward the center, where they merge into a larger and more homogeneous cell mass. Moreover, if there are cells from the neural fold in the composition, the mesodermal cells tend to move to between neural and epidermal cells. That is external epidermal layer, then layer of mesodermal tissue and inner neural tissue. In this study, it is shown that selective affinity is altered during development, which helps cells in their interactions with different types of cells in the processes of morphogenesis.

As an alternative explanation to Townes and Holtfreter [27], Steinberg and Takeichi [25] analyzed two populations of cultured cells with different amounts of adhesion molecules to see the underlying sorting-out mechanism. These cell lines differ from each other only by their P-cadherin synthesis amounts. This fact is especially important when the two are mixed, because the ones with higher cadherin levels surround those with lower-expressions of cadherin. It is thus once again shown that morphogenetic movements and the specific anatomical arrangements are determined by differences in the intensity of cell-cell adhesions.

After [25], Foty and Steinberg [11] evaluated the hypothesis of tissue surface tensions, which describes the path and final order of tissue arrangements that are driven by adhesion between heterotypic and homotypic cells. They examined whether the tissue surface tensions that control mutual tissue segregation, spreading, and cell sorting are produced by the intensities of adhesion between the cells containing these tissues. Different from previous studies, various amount of N-, P- and E-cadherins were measured to quantify the tendency of cells for maximization of their mutual bindings. It was found that different types of chicken embryonic cells were sorted into homotypic arrays, and then organized into segregated tissues. Figure 9 [11], shows the cell lines with lowest cadherin expression, which aggregated slowly and were shaped into small clumps (A), and then the lines with higher cadherin expression that aggregated much more rapidly (B-C-D).

CHAPTER II: AGENT-BASED MODELS AND NETLOGO

2.1 What is an Agent-Based Model?

Agent-based modeling (ABM) makes it possible for us to simulate complicated behavior, from traffic patterns to spread of diseases, by employing individual "agents" that behave according to well-defined but usually stochastic local rules [30]. With the enormous advances in computational capacity, ABMs can simulate events that would have been impossible to do with top-down (differential or difference equation) models. Agents are discrete, autonomous, detectable and traceable units that do not answer to a central authority [17]. They may be mobile, as opposed to static (confined to cells), and they may learn how to change their behavior to survive, as well as be able to reproduce. Agent-based modeling is now so advanced that credible physical theories have been put forward that suggest we are all living in a simulation!

ABMs are immeasurably different from equation-based models (EBMs) as we are able to track individual agents, observe their interactions with other agents and their environment, and visually observe emergent macroscopic complex behavior. The agent, in other words, is the object of the model [22]. The heterogeneity that emerges is worlds apart from the top-down EBM models, which can only describe the average behavior of individuals and consider them identical.

EBMs would be more suited to modeling physical and chemical phenomena, where "particles" are truly identical and interchangeable, and "laws of nature" are much better understood and verified to high precision. In contrast, ABMs are perfect for the messier and much more complex structures and interactions in biology (in addition to those coming from physics and chemistry). A sample of gold is homogeneous, but a sample of bacteria is heterogeneous. Perfect mixing is an unattainable ideal. The more heterogeneous a biological system is, the more useful an ABM becomes [10].

The downside of using ABMs vs. EBMs is mainly that (i) a lot more computational power is required, and (ii) it takes more effort to understand the relevant details of how the system works, and to equip the agents with sufficiently simple decision-making powers.

2.2 Why Agent-Based Modeling with LGCAs ?

The point of origin for this study is how biological pattern formation can be described by lattice-gas cellular automata. The idea of visualizing possible patterns, such as those produced in coats of mammals by pigment cells, requires highly movable agents instead of stationary ones. Randomizing the initial states of "pigment cells" and using a simple stochastic process that has its roots in physics, it is possible to run a large number of simulation experiments with fixed parameter values to see whether the system will be stable and settle in a more or less homogeneous state, or it will be unstable and promote the formation of clumps. A LGCA model makes it possible for us to make educated guesses about the mechanism that moves the particles. When analytical methods are either not available or not practical, LGCA shows us the actual result of the simulation, and makes it unnecessary to examine an entire parameter space. In our study, we will focus on two parameters, α and $\bar{\rho}$ (adhesion strength and particle density respectively), to see which region of the $\alpha - \bar{\rho}$ space gives rise to patterns. A mosaic picture (Fig. 16a-d) made up of actual simulation results shows the bifurcation line as clearly as if it has materialized out of an equation. We should add that the dimensionality and the near-circular symmetry of the hexagonal lattice provides a very manageable and reasonable approximation to the matrix in which pigment cells move. Moving and stochastically-decision-making agents in a symmetric environment makes a LGCA model suitable for this particular problem. Could we have achieved the same goal by employing an ordinary CA model with stochastic transition rules? No, because the LGCA model accommodates extra structure, namely, several particles in the same cell, hence a "shading" of the pigment patterns.

In this study, two types of agents are used: "turtles" and "patches" for the goal of modeling biological patterns by using a lattice-gas cellular automaton. The detailed explanation about their behaviors in the model will be described in the next section.

2.3 NetLogo

NetLogo is a multi-agent programming language that simulates natural and social complex phenomena, authored by Uri Wilensky in 1999 [26]. To investigate the interactions of individuals and their environment, the model is populated with independent agents called "turtles." These agents are moving over a grid of "patches," which form another class of programmable agents. Turtles may be taken to be bees, bacteria, ants, or shepherds, etc., whereas patches may be trees, walls, cancer cells, etc. Several modeling tools, such as cellular automata, genetic algorithms, evolution, and artificial life are enabled by NetLogo. Due to its simplicity, NetLogo allows researchers to build their own models in the authoring environment without much advanced programming skills. It is easy to explore the behaviors of agents under certain conditions and alter the environment to see its effects on simulations. Additionally, NetLogo can be downloaded for free from [33] (Appendix A).

CHAPTER III: OUR MODEL

3.1 Pattern Formation with Adhesive Lattice-Gas Cellular Automata

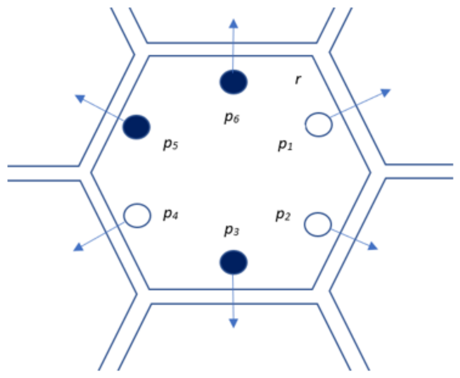
The model in this study was examined by Alexander et al. [2] initially, then improved by Bussemaker [4] and featured in Deutsch and Dormann [6]. It characterizes the motion of particles (which we think of as pigment cells) in a two-dimensional, L x L hexagonal lattice with $L = 75$ in our case. The overall dynamics of our model is created by two operations, repeated for every time step. These are the stochastic adhesive interaction rule I, which defines a reorientation of a fixed number of particles in a hexagon towards more populous neighbor cells with highest probability, and then the deterministic propagation rule P that ensures the movement of each oriented particle in a certain direction to the neighbor cell in that direction with probability 1.

We analyze a discrete *adhesive Boltzmann (LGCA) model* with the assumption that all particles move at the same time and at the same speed ($m = 1$) on a hexagonal lattice. Because of its maximum rotational symmetry, we chose the hexagonal lattice to study among all three regular tilings as mentioned before. The model is specifically suited for examination of pattern formation because it increases the effects of little fluctuations that are present in a nearly homogeneous initial state [4]. According to Deutsch [7], it is also biologically favorable since the local rules are designed to randomly minimize the work done against nearby cells, instead of the usual physics requirement of lowering energy deterministically. Using microscopic rules of statistical mechanics for macroscopic predictions makes this model simplistic yet useful to work in the mesoscopic (about cell-size) scale. An important point is that all interactions are between cells and not between cells and their environment.

The centers of the hexagonal cells, called *nodes*, are indexed by vectors $\mathbf{r}$. The six *velocity channels* that serve both as velocity and direction vectors are given by

$$\mathbf{c}_i = \left(\cos\left(\frac{2\pi(i-1)}{6}\right), \sin\left(\frac{2\pi(i-1)}{6}\right)\right), \quad 1 \le i \le 6.$$

Hence, the node of the neighboring hexagonal cell in direction $\mathbf{c}_i$ is given by $(\mathbf{r} + \mathbf{c}_i)$. As we mentioned previously, every channel can be occupied by at most one particle at any given time step, and the total number of particles in every cell is conserved during the stochastic phase. Additionally, the total number of particles is preserved during the propagation step. Consequently, the *average number $\bar{\rho}$ of particles per cell* remains constant during each simulation. This density is one of the determining factors of pattern formation in this model.



$$b = 6$$

$$velocity\ channels: (r, p_1), (r, p_2), (r, p_3), (r, p_4), (r, p_5), (r, p_6)$$

$$\boldsymbol{\eta}(r) = (\eta_1(r), \eta_2(r), \eta_3(r), \eta_4(r), \eta_5(r), \eta_6(r))$$

$$= (0,0,1,0,1,1)$$

$$n(r) = 3$$

Figure 10: Two-dimensional hexagonal lattice configuration

Fig. 10 gives an example of a configuration of a cell in a two-dimensional hexagonal lattice, accommodating up to 6 particles. The presence of particles are shown with filled dots in the corresponding channels.

The *occupation number* of channel i of the cell at node $\mathbf{r}$ is denoted by $\eta_i(\mathbf{r}, t) \in \{0, 1\}$. In other words, it is the number of particles in the channel facing direction $\mathbf{c}_i$ at the beginning of time step t in the pre-collision state. Then $n(\mathbf{r}, t) = \sum \eta_i(\mathbf{r}, t)$ is the occupation number of the whole cell. In order to describe the probabilistic adhesive interaction rule, we examine each cell and its immediate neighbors in the lattice, by computing the *gradient* vector

$$\mathbf{G}(\mathbf{r}, t) = \sum_{i=1}^{6} n(\mathbf{r} + \mathbf{c}_i, t)\mathbf{c}_i,$$

which defines a weighted sum of direction vectors according to the populations of neighbor cells only, and serves as the direction that the particles in the middle are most likely to “gravitate” towards. For the transition into the post-collision state $\eta_i^I(\mathbf{r}, t)$, we also need to know the total number $n = n(\mathbf{r}, t)$ of particles in the center cell. However, it is not necessary to know the channels that they are currently located at, since they will be re-positioned into n channels of the same cell with certain probabilities. All particles are identical, so there are $N(n) = \binom{6}{n}$ ways of placing them into channels. Each possible

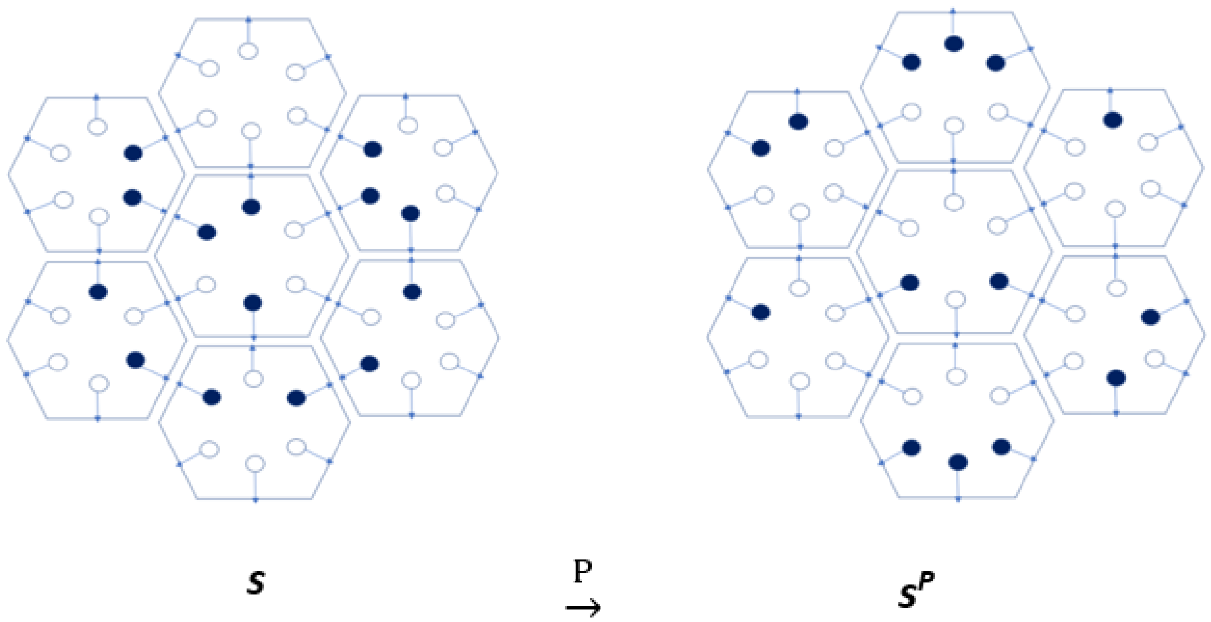


Figure 11: Two-dimensional hexagonal lattice configuration with propagation step

configuration is given by some collection $\{\eta_i\}$ of occupation numbers, which denotes the near future -but not the present- state. For each $0 \leq n \leq 6$, we have $N(n)$ pre-computed and stored $flux$ vectors

$$\mathbf{J}(\{\eta_i\}) = \sum_{i=1}^{6} \eta_i \mathbf{c}_i$$

that represent the sum of velocities of the particles occupying the center cell, once they are placed in the relevant channels. That is, the particles will be posed to move in this weighted direction and with this velocity (on the average) in the propagation step. Since there will be no change in the configuration when the cell is either completely empty or completely full, it is enough to compute the flux vectors for $1 \leq n \leq 5$ only.

In Figure 11, we can see the propagation step realized in the two-dimensional hexagonal lattice with speed $m = 1$, where the lattice configurations before $(\mathbf{s})$ and after $(\mathbf{s}^P)$ the propagation step are shown. The presence of a particle is shown with a filled dot in the corresponding channel.

In order to choose the configuration $\{\eta_i\}$ that will re-distribute the particles in the

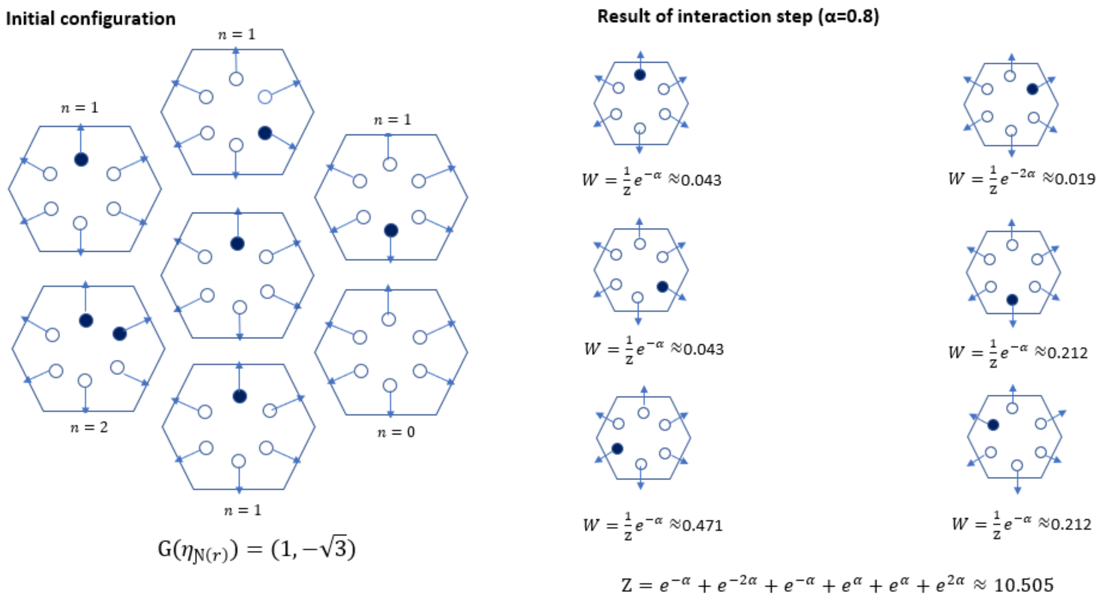


Figure 12: Example for adhesive interaction in the hexagonal lattice

center cell, we take the dot product $\mathbf{G}(\mathbf{r},t)\cdot\mathbf{J}(\{\eta_i\})$ for each configuration. This result will be at its maximum when both $\mathbf{G}(\mathbf{r},t)$ and $\mathbf{J}(\{\eta_i\})$ are facing the same direction and at its minimum when they are facing opposite directions. The code is instructed to choose $\mathbf{J}$ such that $\mathbf{G}(\mathbf{r},t)$ and $\mathbf{J}$ are closest to each other with the highest probability. Therefore, we define the probability that $\{\eta_i(\mathbf{r},t)\}$ will become $\{\eta_i\}$, given $\mathbf{G}(\mathbf{r},t)$, to be

$$W\big(\{\eta_i(\mathbf{r},t)\}\to\{\eta_i^I(\mathbf{r},t)\}=\{\eta_i\}|\mathbf{G}(\mathbf{r},t)\big)=\frac{e^{\alpha\mathbf{G}(\mathbf{r},t)\cdot\mathbf{J}(\{\eta_i\})}}{Z},$$

where Z is the normalization factor

$$Z=\sum_{\{\eta_i'\}}e^{\alpha\mathbf{G}(\mathbf{r},t)\cdot\mathbf{J}(\{\eta_i'\})}.$$

Figure 12 illustrates an example of the transition probability. The right side (η^I) denotes all possible results of the interaction step that is applied to the center cell of initial configuration (η) with probability W. According to the results, the particle will be poised to move towards the neighbor on the south-west most probably, but there is small nonzero probability that the particle may go to any other neighbor due to our stochastic rule. Here, $\alpha > 0$ is the second parameter in the model, which determines the *sensitivity* or the *adhesion strength* of particles. With negative values of α, we have a repulsive model of particles. The idea of exponentiation is useful because it

- creates a nonzero probability for each configuration to be realized, and
- preserves the order of the values of the dot products as real numbers.

For small positive values of α, particles will be mostly reshuffled into channel configurations with equal probability, and for large values of α, they will be more likely to get into a configuration that matches the gradient. In the latter case, homogeneous patterns (stability) will be broken, and a pattern formation through adhesion will become possible. Hence, for each fixed value of the particle density $\bar{\rho}$, there exists a critical value of α at which stability turns into unstability. This is shown by a graph for a square lattice in Bussemaker [4] and Deutsch [7] (Fig. 14). We have simulated a similar graph (Fig. 13) for the hexagonal lattice.

During the propagation stage of time step t, every particle moves from its channel to the same channel in the next hexagon that lies in the direction of the velocity of the channel by the deterministic rule

(1) $$\eta_i(\mathbf{r} + \mathbf{c}_i, t+1) = \eta_i^I(\mathbf{r}, t).$$

We employ periodic boundary conditions for propagation. In our simulations with NetLogo, shadings of gray are used to denote particle concentrations in each cell. Pattern formation is indicated by dark patches that become more defined as time passes, and there are white spaces left between them as empty cells. This is a demonstration of the unstability of equilibrium states that are indicated by a uniform distribution of smaller, lighter patches. Due to the stochasticity of the model, we need to work with the expected (average) values of occupation numbers in order to study stability analytically.

3.2 Aside: Stability Analysis of Systems of First Order Nonlinear Autonomous Difference Equations

Mathematical models for discrete-time biological systems are generally more complicated than linear models, and they require nonlinear difference equations. Identification of equilibrium points and examination of their stability are standard tools [16]. If the function F in a first-order system

$$X_{t+1} = F(X_t)$$

does not depend on t explicitly, then the system is referred to as an $autonomous$ difference equation, otherwise, it is $nonautonomous$. We will only be concerned with the autonomous case. For a first-order difference equation

$$x_{t+1} = f(x_t),$$

an $equilibrium\ point$ is a constant solution $\bar{x}$ of the difference equation that satisfies

$$\bar{x} = f(\bar{x}).$$

Similarly, for the first-order system $X_{t+1} = F(X_t)$, an $equilibrium\ point$ is a solution $\bar{X}$ that satisfies

$$\bar{X} = F(\bar{X}).$$

Next, we define the local stability of an equilibrium point. An equilibrium point $\bar{x}$ of $x_{t+1} = f(x_t)$ is $locally\ stable$ if, for any $\epsilon > 0$, there exists $\delta > 0$ such that whenever $|x_0 - \bar{x}| < \delta$, we have

$$|x_t - \bar{x}| = |f^t(x_0) - \bar{x}| < \epsilon \quad \text{for every} \quad t \geq 0.$$

If $\bar{x}$ is not stable, then it is said to be *unstable*. The equilibrium point $\bar{x}$ is *locally attracting* if there exists $\gamma > 0$ such that for all $|x_0 - \bar{x}| < \gamma$, we have

$$\lim_{t\to\infty} x_t = \lim_{t\to\infty} f^t(x_0) = \bar{x}.$$

Finally, the equilibrium point $\bar{x}$ is *locally asymptotically stable* if it is stable and locally attracting.

Theorem [16]. Assume that f is continuously differentiable on an open interval I containing $\bar{x}$, and that $\bar{x}$ is a fixed point of f. Then $\bar{x}$ is a locally asymptotically stable equilibrium of $x_{t+1} = f(x_t)$ if

$$|f'(\bar{x})| < 1,$$

and is unstable if

$$|f'(\bar{x})| > 1.$$

Let us next consider a first-order system consisting of n equations, with $X(t) = (x_1(t), x_2(t), \cdots, x_n(t))^T$, and

$$X(t+1) = F(X(t)),$$

where $F = (f_1, f_2, \cdots, f_n)$ and $f_i = f_i(x_1, x_2, \cdots, x_n)$, $i = 1, 2, \cdots, n$. Now, suppose that this system has an equilibrium at $\bar{X}$. Then the corresponding Jacobian matrix J, evaluated at $\bar{X}$, is

$$\begin{bmatrix} \frac{\partial f_1(\bar{X})}{\partial x_1} & \frac{\partial f_1(\bar{X})}{\partial x_2} & \cdots & \frac{\partial f_1(\bar{X})}{\partial x_n} \\ \frac{\partial f_2(\bar{X})}{\partial x_1} & \frac{\partial f_2(\bar{X})}{\partial x_2} & \cdots & \frac{\partial f_2(\bar{X})}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_n(\bar{X})}{\partial x_1} & \frac{\partial f_n(\bar{X})}{\partial x_2} & \cdots & \frac{\partial f_n(\bar{X})}{\partial x_n} \end{bmatrix}.$$

J is the "linearization" of the system at $\bar{X}$, and is analogous to $|f'(\bar{x})|$ above. The local asymptotic stability of $\bar{X}$ is dependent on the eigenvalues of the Jacobian matrix J. (We require that all partial derivatives of f_i be continuous in an open set containing $\bar{X}$.) The eigenvalues of the Jacobian matrix are the solutions of the characteristic equation

$$\det(J - \lambda I) = 0.$$

The following well-known theorem can be generalized to $\mathbb{R}^n$. The main point is that all eigenvalues of the Jacobian matrix at an equilibrium point must have absolute values (or moduli in the complex case) smaller than 1, so that we can say the equilibrium is asymptotically stable. Let $\rho(A)$ denote the maximum modulus of all (complex) eigenvalues of a square matrix.

Theorem [8]. Let $f : I \subset \mathbb{R}^2 \to \mathbb{R}^2$ be continuously differentiable, where I is an open subset of $\mathbb{R}^2$, $\bar{x}$ be an equilibrium point of f, and $A = J(\bar{x})$, where J is the corresponding Jacobian matrix. Then the following are true:

- If $\rho(A) < 1$, then $\bar{x}$ is asymptotically stable.
- If $\rho(A) > 1$, then $\bar{x}$ is unstable.
- If $\rho(A) = 1$, then $\bar{x}$ may or may not be unstable.

3.3 Stability Analysis of the Hexagonal ALGCA

Let $f_i(\mathbf{r},t) = \langle \eta_i(\mathbf{r},t) \rangle$ denote the single-particle distribution function that is obtained by averaging Eq. (1) over all possible initial positions. Then, the average rate of change of the occupation number of ith channel can be written as

$$f_i(\mathbf{r}+\mathbf{c}_i, t+1) - f_i(\mathbf{r},t) = \langle \eta_i^I(\mathbf{r},t) - \eta_i(\mathbf{r},t) \rangle, \quad 1 \leq i \leq 6.$$

By ignoring the dependence of the micro-states on each other and neglecting fluctuations, we obtain what is called the nonlinear *mean-field (Boltzmann) approximation* in physics:

(2) $$f_i(\mathbf{r}+\mathbf{c}_i, t+1) - f_i(\mathbf{r},t) = \langle \eta_i^I(\mathbf{r},t) - \eta_i(\mathbf{r},t) \rangle_{MF}, \quad 1 \leq i \leq 6.$$

These 6 equations form a deterministic, nonlinear, autonomous (the local rules are independent of t) system of difference equations. Namely, it is possible to search for equilibrium states by setting the right-hand side equal to zero and to discuss the stability of these states by the usual linearization process. It can be shown that for the ith channel, the value $f_i(\mathbf{r},t) = \bar{\rho}/6$, where $\bar{\rho}$ is the average density of cells at a node, gives us an equilibrium point as in [4]. Since we randomly distribute $\bar{\rho}L^2$ particles to all channels at $t=0$, we always start in a nearly homogeneous state close to this equilibrium, and then analyze the evolution of the system in time under fixed parameters $\bar{\rho}$ and α.

The linearization, eigenvalues, and eigenvectors have been computed in closed form for a square lattice in [4], after the f_i have been subjected to a Fourier transform into **q** (momentum) space, and the Fourier coefficients are examined. However, even in this relatively simpler case, only writing the dominant eigenvalue as an approximation up to $O(|\mathbf{q}|^4)$ was possible. Thus, the condition that its modulus is equal to 1 (a critical value) was approximate. The critical values for α were then obtained from this condition

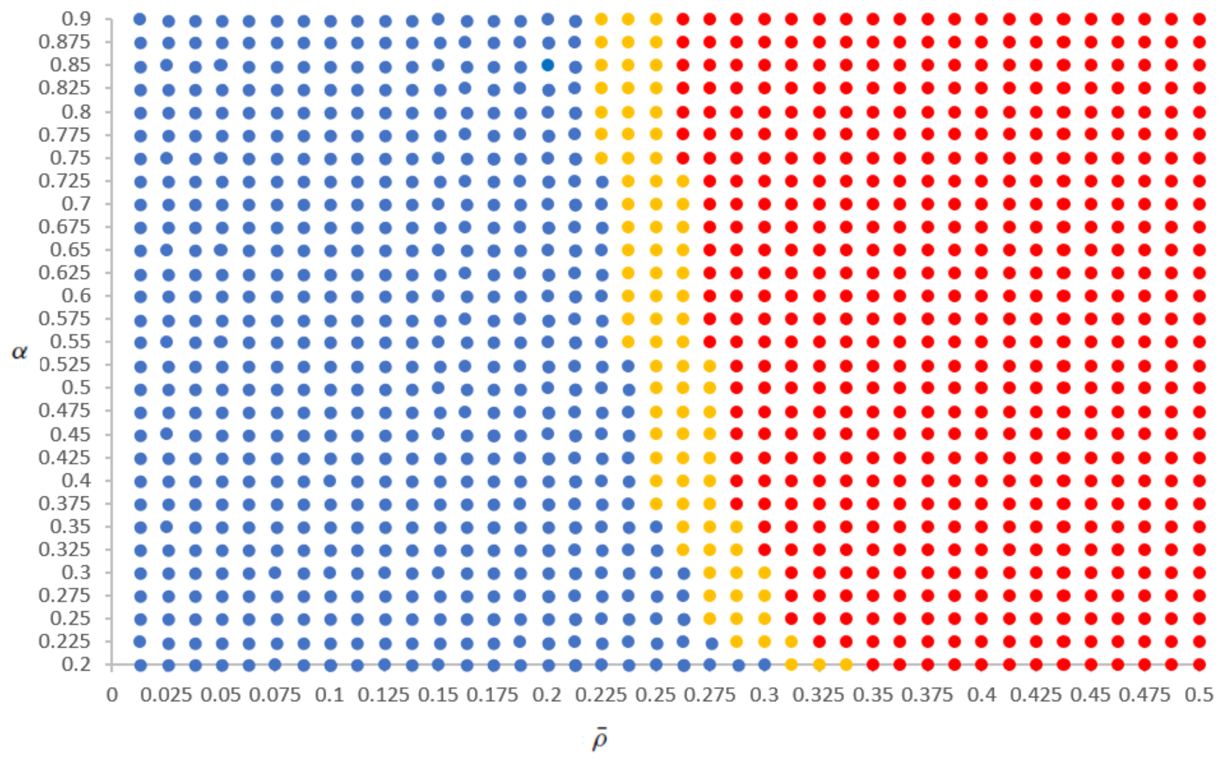


Figure 13: Phase diagram of the adhesive one-species LGCA model for the hexagonal lattice

numerically by fixing values of $\bar{\rho}$. Since the underlying equations for a hexagonal lattice are more complicated, we decided to make a similar phase space diagram showing the critical values of α as a function of $\bar{\rho}$, and separate the stable/unstable (no pattern/pattern) regions by performing simulations for various values of $(\alpha, \bar{\rho})$ in a grid (Fig. 13). Hence, we were able to draw a similar demarcation curve to the one in Fig. 14 that was not simulated either in [4] or [6].

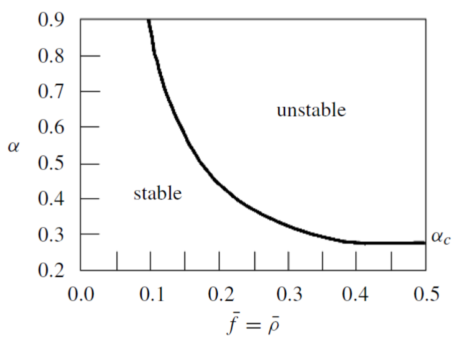


Figure 14: Diagram for adhesive pattern formation in the LGCA model on square lattice with stable and unstable cases

3.4 The Simulation

Let us now show the results of our simulations, all starting from spatially homogeneous random initial conditions. On a 75x75 hexagonal lattice, we ran our NetLogo code for 100 iterations for each pair of values $(\alpha, \bar{\rho})$ in a rectangular grid. We then visually determined whether the homogeneous equilibrium was stable (no pattern; blue points) or unstable (distinctive patterns; red points). The transitional yellow dots show patterns that display both kinds of characteristics. The resulting graph is shown in Figure 13, and is comparable to the stable/unstable regions for the square grid in [6], page 154 (Fig 14).

Two additional figures show special cases, one unstable and one stable, with intermediate stages (Figure 15). These longer runs were not possible for every point on the grid due to inadequate computing time. However, these examples clearly show the cases of increasing instability and increasing homogenization respectively, as time progresses. Shades of gray, from white to black, indicate lower to higher density of pigment particles. Finally, we have put together a mosaic of the end results of the 100-run simulations, in the same relative positions to the corresponding $(\alpha, \bar{\rho})$ pairs in the $\alpha\bar{\rho}$-plane. This blow-up of Figure 13 is indeed striking, and shows the advantage of agent-based models in visualization over EBMs.

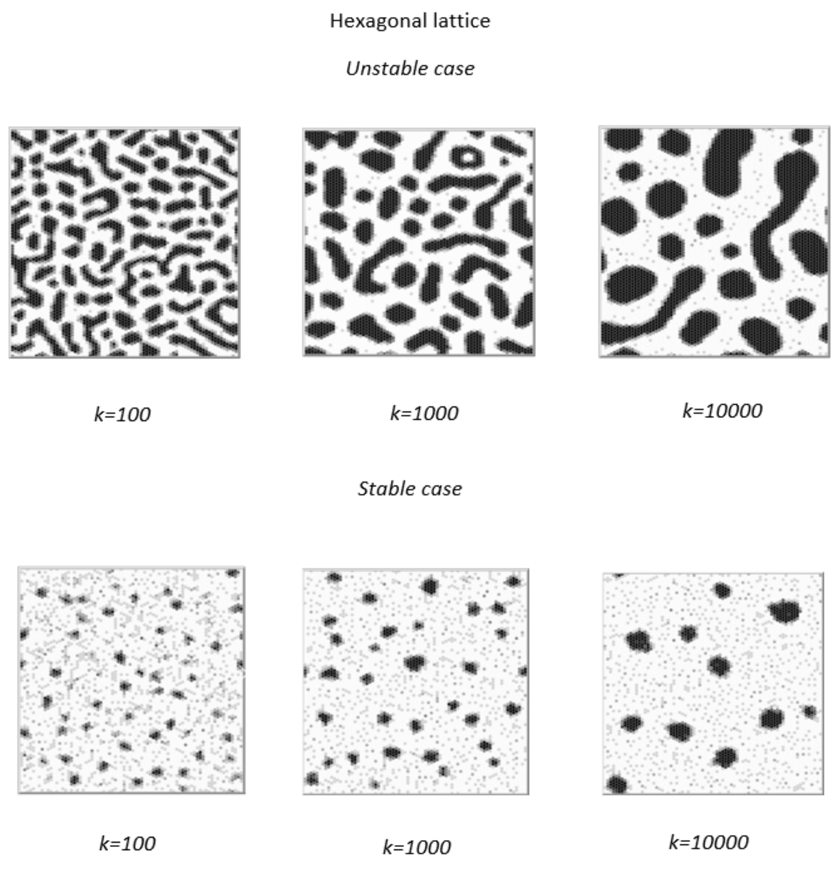


Figure 15: Adhesive pattern formation in the LGCA model on hexagonal lattice with stable and unstable cases

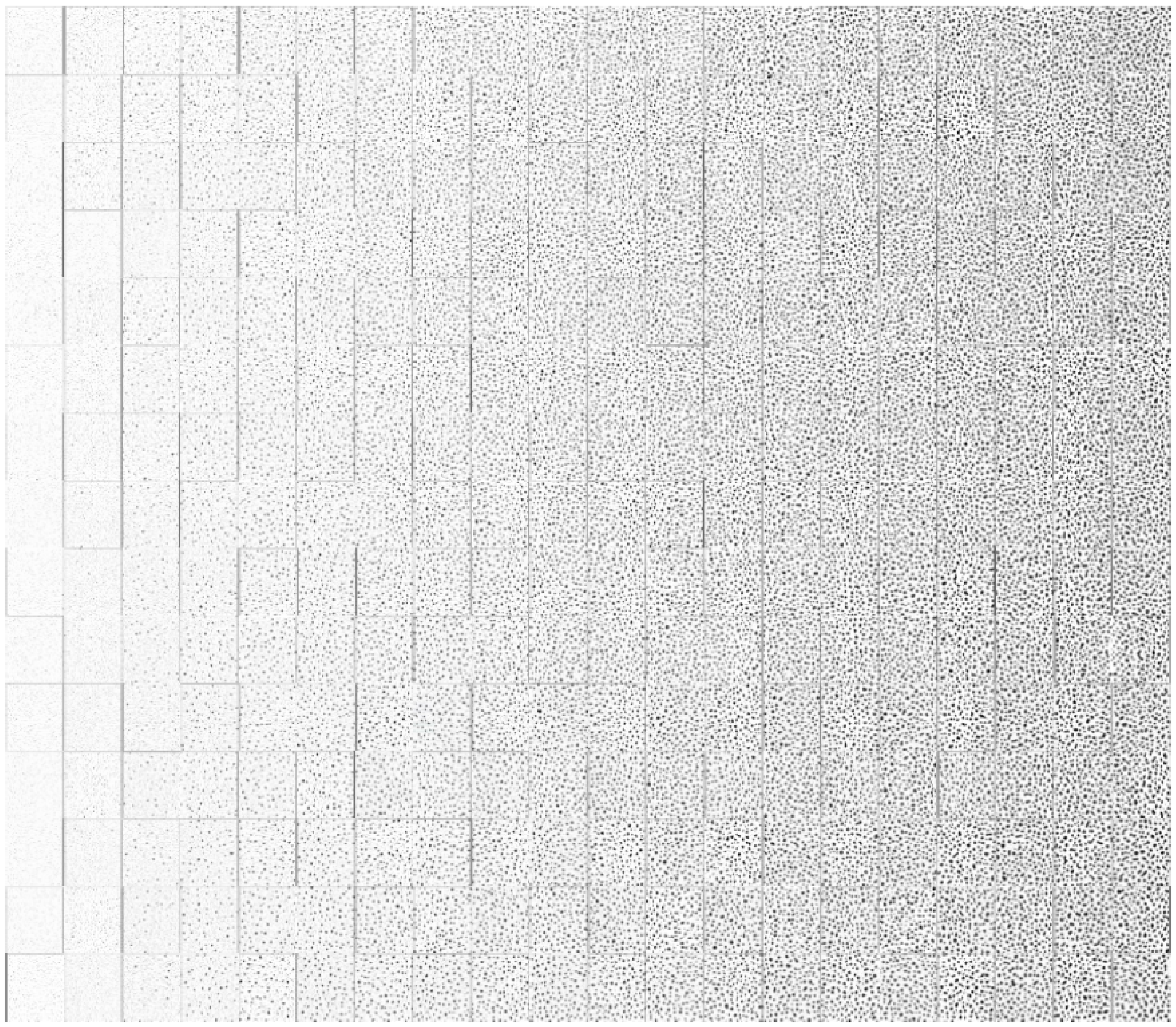


Figure 16a: The mosaic configuration of 100-run simulations in the $\alpha\bar{\rho}$-plane. The x-axis shows $\bar{\rho}$ values from 0 to 0.25 by 0.0125 increment. The y-axis denotes α values starting from 0.55 to 0.9 by 0.025

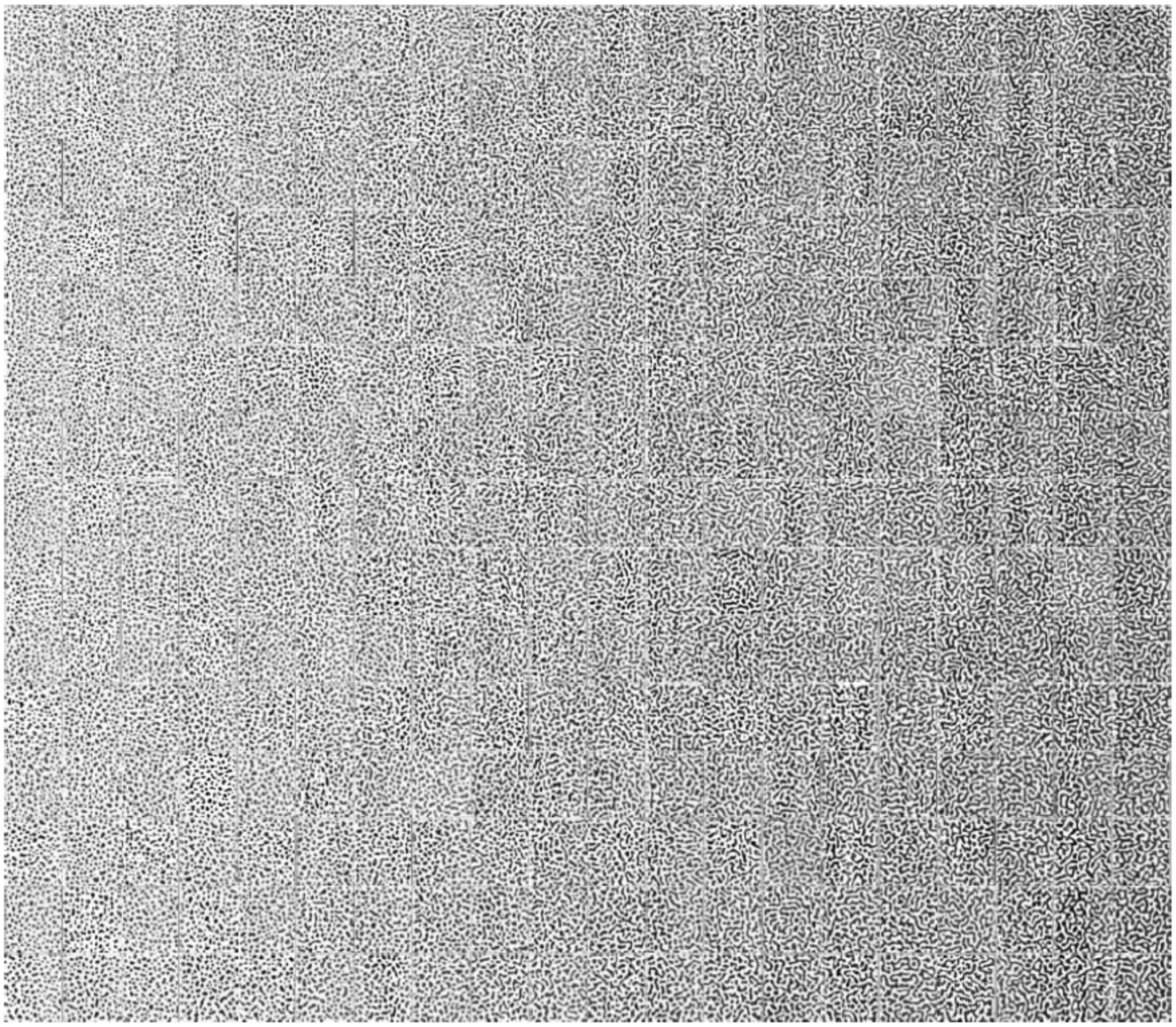


Figure 16b: The mosaic configuration of 100-run simulations in the $\alpha\bar{\rho}$-plane. The x-axis shows $\bar{\rho}$ values from 0.25 to 0.5 by 0.0125 increment. The y-axis denotes α values starting from 0.55 to 0.9 by 0.025

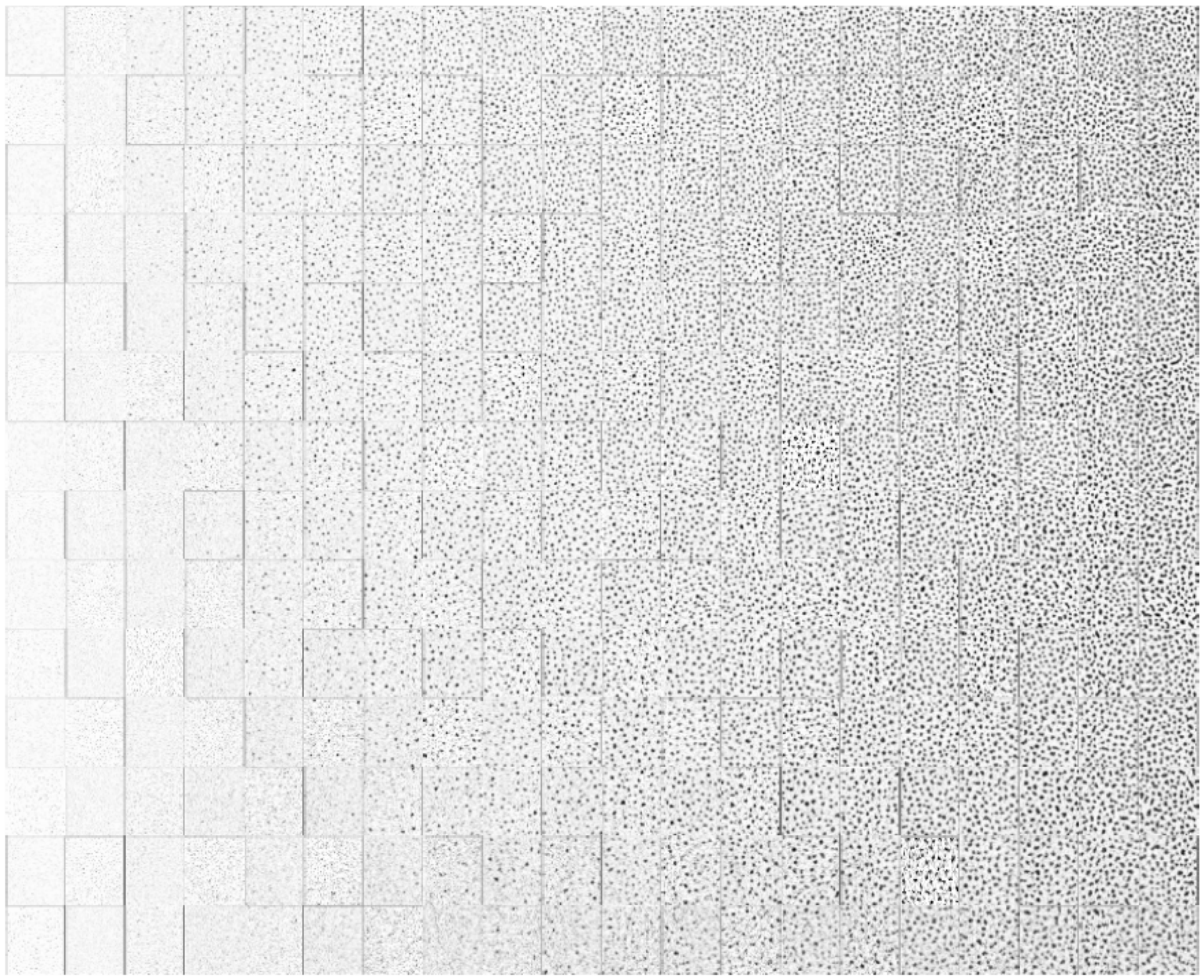


Figure 16c: The mosaic configuration of 100-run simulations in the $\alpha\bar{\rho}$-plane. The x-axis shows $\bar{\rho}$ values from 0 to 0.25 by 0.0125 increment. The y-axis denotes α values starting from 0.2 to 0.525 by 0.025

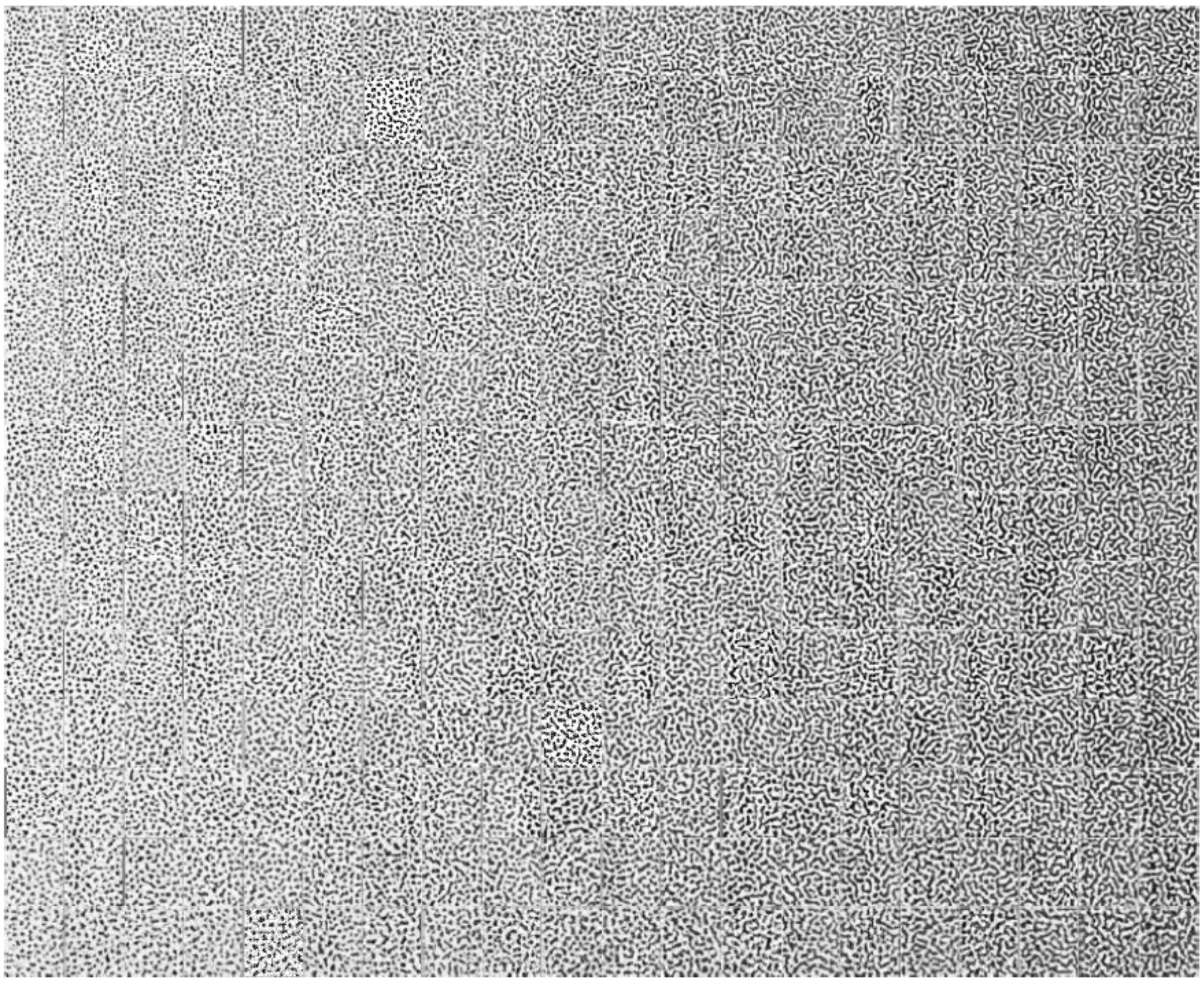


Figure 16d: The mosaic configuration of 100-run simulations in the $\alpha\bar{\rho}$-plane. The x-axis shows $\bar{\rho}$ values from 0.25 to 0.5 by 0.0125 increment. The y-axis denotes α values starting from 0.2 to 0.525 by 0.025

CHAPTER IV: DISCUSSION

4.1 Conclusions

The results of our simulation showed that in a hexagonal lattice-gas cellular automata model with a single cell type, the parameters α and $\bar{\rho}$ play a significant role in pattern formation. Due to the complicated of equations for a hexagonal lattice, we showed the critical values of α as a function of $\bar{\rho}$ on a phase space diagram by simulations. That is, the regions for unstable and stable cases were described according to pattern formation in NetLogo simulations. For higher values of α, the equilibrium states turned into unstable slightly sooner than they did for smaller α values. Smaller and lighter patches with low α value remained homogeneously distributed for a longer time due to the nature of the adhesive interaction coefficient. On the other hand, for a fixed adhesion coefficient value, the equilibrium states turned into patterns in narrow range of particle density values (yellow dots). Moreover, the results may imply that a combination of the adhesion coefficient and the particle density requires to change its configuration into pattern formation around the middle values of the particle density.

4.2 Future Work

There is surely more future work to do related with this model for the hexagonal lattice and finding an approximate value similar to the one in [4] and [6] for the dominant eigenvalue in closed form. In order to determine formation of spatial patterns, Fourier analysis can be very useful, as shown in [7]. On the other hand, in [4] a more realistic model is shown, where correlations of microstates are taken into account. Additionally, changing the local interaction rules or creating the patterns with two or more types of pigment cells might be interesting for future study. It is also possible to try asynchronous updates to see how the model evolves. What are the impacts of different boundary conditions on general pattern formation? Besides, we assumed our lattice-gas cellular automaton model under no creation or annihilation of particles, the absence of which might change the final structure completely. For real life applications, tumor invasion may be studied with cellular automata models to see the distinctions between healthy and tumor cells formation patterns. It is also possible to search for the invasive cells that might diffuse at different speeds.

REFERENCES

[1] Agapie A., Andreica A., Giuclea M. (2014). *Journal of Computational Biology*, **21** (9), 699–708.

[2] Alexander F. J.,Edrei I.,Garrido P. L.,Lebowitz J. L. (1992) Phase transitions in a probabilistic cellular automaton: growth kinetics and critical properties.*Journal of Statistical Physics*, **68**, 497-514.

[3] Belekamp E., Conway J. and Guy R. (1982). "What Is Life?" Chapter 25 in Winning Ways for Your Mathematical Plays. *Academic Press*, **2**: Games in Particular.

[4] Bussemaker H. (1996). Analysis of a pattern forming lattice gas automaton: mean-field theory and beyond.*Physical Review*, **53**, 1644-1661.

[5] Camazine S., Deneubourg J-L., Franks N., Sneyd J., Theraulaz G., and Bonabeau E. (2001). Self-Organization in Biological Systems, *Princeton University Press, Princeton, NJ.*

[6] Deutsch A., Dormann S. (2004). Cellular Automaton Modeling of Biological Pattern Formation: Characterization, Applications and Analysis. *Birkhäuser.*

[7] Deutsch, A. (2007). Lattice-gas cellular automaton modeling of developing cell systems. In Single-Cell-Based Models in Biology and Medicine. Mathematics and Biosciences in Interaction. *Birkhäuser Basel.*

[8] Elaydi, S. N. (2007). Discrete chaos: with applications in science and engineering. *CRC Press.*

[9] Fishwick, P. A. (Ed.). (2007). Handbook of dynamic system modeling. *CRC Press.*

[10] Forrester , J. W. (1968). Principles of Systems. Norwalk, CT: *Productivity Press.*

[11] Foty, R. A., Steinberg, M. S. (2005). The differential adhesion hypothesis: a direct evaluation. *Developmental biology*, **278**, 255-263.

[12] GAMES, M. The fantastic combinations of John Conway's new solitaire game "life" by Martin Gardner. *Scientific American,* 223, 120-123.

[13] Godt, D., Tepass, U. (1998). Drosophila oocyte localization is mediated by differential cadherin-based adhesion. *Nature*, **395**, 387.

[14] González-Reyes, A., St Johnston, D. (1998). The Drosophila AP axis is polarised by the cadherin-mediated positioning of the oocyte. *Development*, **125**, 3635-3644.

[15] Hardy P., Pomeau Y., and De Pazzis O. (1973). Time evolution of a two-dimensional model system. I. Invariant states and time correlation functions. *Journal of Mathematical Physics*, **14**, 1746–1759.

[16] Linda, J. A. (2007). An introduction to mathematical biology. *PEARSON, Prentice Hall.*

[17] Macal C. M., North M. J. (2007). Agent-Based Modeling and Simulation: Desktop ABMS. *IEEE.*

[18] Mitchell M. (2002). Is the Universe Universal Computer? *Science*, **298**, 65-69.

[19] Morita K., Margenstern M., Imai K. (1999). Universality of reversible hexagonal cellular automata. *RAIRO-Theoretical Informatics and Applications*, **33** (6), 535–550.

[20] Murray, J. D. (1988). How the leopard gets its spots. *Scientific American*, **258**, 80-87.

[21] Peifer, M. (1998). Developmental Biology: Birds of a feather flock together.*Nature*, **395**, 324.

[22] Rahmandad, H., Sterman, J. (2008). Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models. *Management Science*, **54**, 998-1014.

[23] Rietkerk, M., Boerlijst, M. C., van Langevelde, F., HilleRisLambers, R., de Koppel, J. V., Kumar, L., ...& de Roos, A. M. (2002). Self-organization of vegetation in arid ecosystems. *The American Naturalist*, **160**, 524-530.

[24] Rietkerk, M., Van de Koppel, J. (2008). Regular pattern formation in real ecosystems. *Trends in ecology & evolution*, **23**, 169-175.

[25] Steinberg, M. S., Takeichi, M. (1994). Experimental specification of cell sorting, tissue spreading, and specific spatial patterning by quantitative differences in cadherin expression. *Proceedings of the National Academy of Sciences*, *91*, 206-209.

[26] Tisue S. and Wilensky U. (2004). NetLogo: A Simple Environment for Modeling Complexity.*Presented at the International Conference on Complex Systems, Boston, May 16–21.*

[27] Townes, P. L., Holtfreter, J. (1955). Directed movements and selective adhesion of embryonic amphibian cells. *Journal of Experimental Zoology Part A: Ecological Genetics and Physiology*, **128**, 53-120.

[28] Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, **237**, 37-72.

[29] Weiss, P. (1950). Perspectives in the field of morphogenesis. *The Quarterly review of biology*, **25**, 177-198.

[30] Wilensky, U., & Rand, W. (2015).*An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. MIT Press.

[31] Wolf-Gladrow, D. A. (2004). *Lattice-gas cellular automata and lattice Boltzmann models: an introduction*. Springer.

[32] Wolfram S. (1984). Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2): 1-35.

[33] https://ccl.northwestern.edu/netlogo/index.shtml (Last accessed 05/02/2018)

APPENDIX A: NETLOGO

A1. Obtaining NetLogo

To download NetLogo, please go to https://ccl.northwestern.edu/netlogo/download.shtml and click on the Download button. For the next step, we need to choose the suitable operating system and follow the required instructions through the downloading process.

A2. NetLogo Variables in Our Model

We have 5 global variables in the model:

- spawn-p
- α - adhesion coefficient
- m - velocity of particle
- G - gradient field vector
- $J_1, J_2, J_3, J_4, J_5, J_6$- the direction vectors
- hexagon size

Spawn-p determines the probability of a node sprouting one particle. High spawn-p means every node will get more particles and setting up at 1 implies every node will get 6 particles. There is a slider from 0 to 1 for it on the interface. Recall that α denotes the strength of adhesion as a positive parameter. It is defined in such a way that particles prefer to move in the direction of increasing density of cells. However, a negative value for α implies that particles tend to migrate in the direction of decreasing density, known as repulsion. The hexagon size determines the size of each cell after setting their shapes as hexagons. This number is specifically selected, since we need to see the pattern of adjacent cells without any space between them. The symbol m shows the velocity of each particle and in this study; we set $m = 1$, which means that every particle migrates to the next cell according to its direction vector. The gradient field vector G is defined to model the tendency of particles to move towards the heavier neighbors. It includes the weighted geometric distribution of particles among the neighbors but not any information about the center. This way, which ever neighbor has more particles has a greater sloped gradient field vector towards itself. Finally, the direction vectors, $J_1, J_2, J_3, J_4, J_5, J_6$, denote the direction vectors, and are given as

$$J_1 = (1,\ 0),\ \ J_2 = \left(\frac{1}{2},\ \frac{\sqrt{3}}{2}\right),\ \ J_3 = \left(-\frac{1}{2},\ \frac{\sqrt{3}}{2}\right),$$

$$J_4 = (-1, 0),\ \ J_5 = \left(-\frac{1}{2},\ -\frac{\sqrt{3}}{2}\right),\ \ J_6 = \left(\frac{1}{2},\ -\frac{\sqrt{3}}{2}\right)$$

A3. Set Up

There are two breeds of turtles: walkers and nodes. Walkers represent the particles at each cell, and nodes are the hexagonal lattices. The walkers begin in a randomly distributional form. This aims to mimic the biological pattern formation with a realistic representation of structure. We coded the direction $(0, 0)$ to be “down;” this way is more convenient due to the hexagonal shapes in NetLogo and does not affect the result. Each hexagonal cell can have at most 6 particles, and the boundaries of the lattice are wrapped around (periodic). The initial setting has shaded hexagons that determines the particle number of cells. Accordingly, heavier shades of gray denote particles with more cells, and white cells are empty.

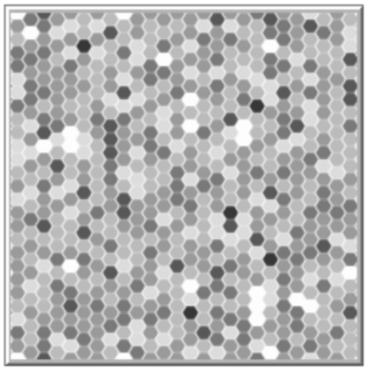
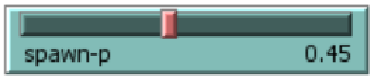


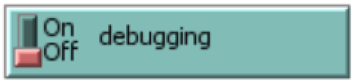
Figure A-1: Initial NetLogo Setup.

A4. Adjusting Code

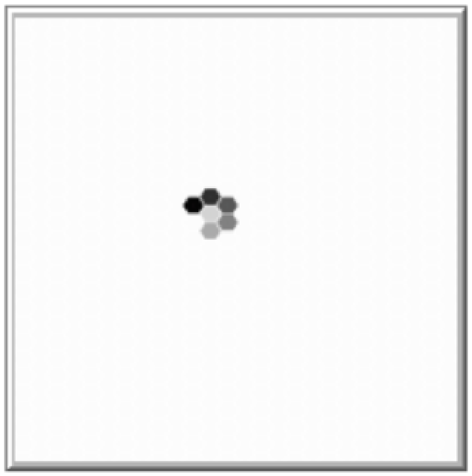
There are ten different settings in the user interface, which can be altered to match the population at hand. To customize sliders, click on the selector and move them to the left or right.

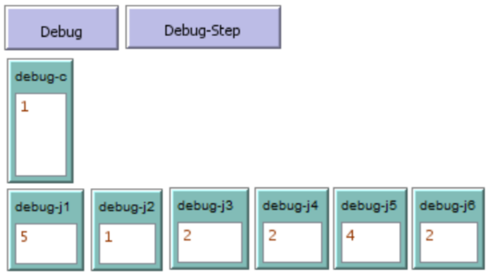

spawn-p 0.45

To turn debugging on or off, click on the switch;

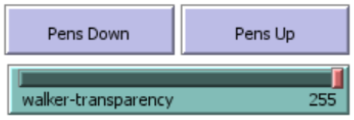


For checking calculations, there are $Debug$ as set up, $Debug - Step$ and seven input windows to put particle numbers between 0 and 6 including center cell. Once we set the values, all other cells will be empty but only seven cells at the center will be occupied according to the numbers that are entered. Then, with debug-step, it is possible to observe the behavior of particles in the center cell.





To state the walker's pen, click on pens down once we set walker-transparency to high value for clear vision. Pen is a tool to draw lines for every movement commands of turtles. To erase the lines, click on Pens Up;

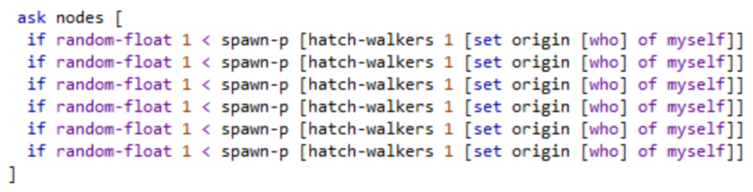


All settings that are not listed in the user interface can be adjusted directly from the code. To start with the configuration of hexagonal lattice, we write

```
to setup
  clear-all
  set-default-shape nodes "hex"
  foreach sort patches [ p ->
  ask p [
  set pcolor white sprout-nodes 1
  [let hex-color (list 100 100 100 hex-transparency)
  set color hex-color
  set size 1.2
  if pxcor mod 2 = 0
  [set ycor ycor - 0.5 ] ] ]
  ]
```

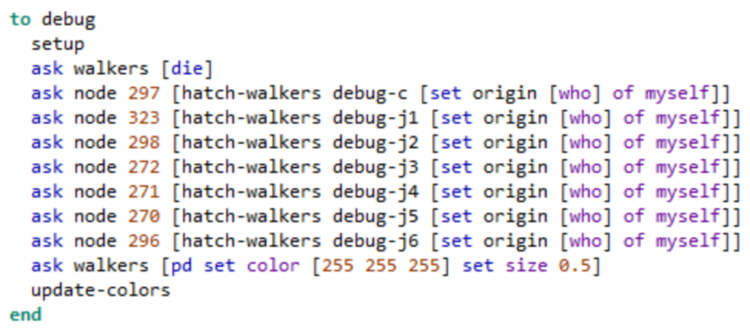
Here, the background is set in white color to have a better pattern image. Furthermore, there is a slider for hexagonal cell transparency, which adjusts the color of cells to darker or lighter.

In addition, for randomly distributed particles, $random - float$ was used with $spawn - p$ which denotes the particle density. If we increase the density, it is likely that we will get higher number of particles for each cell.



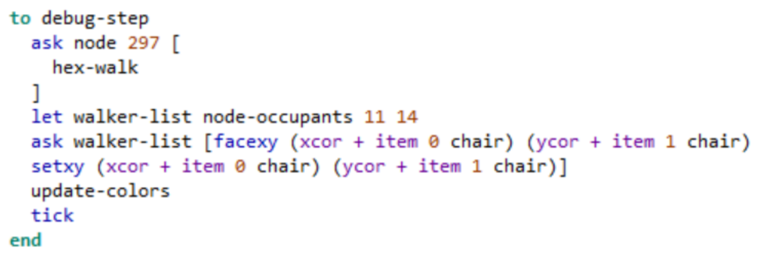
```
ask nodes [
  if random-float 1 < spawn-p [hatch-walkers 1 [set origin [who] of myself]]
  if random-float 1 < spawn-p [hatch-walkers 1 [set origin [who] of myself]]
  if random-float 1 < spawn-p [hatch-walkers 1 [set origin [who] of myself]]
  if random-float 1 < spawn-p [hatch-walkers 1 [set origin [who] of myself]]
  if random-float 1 < spawn-p [hatch-walkers 1 [set origin [who] of myself]]
  if random-float 1 < spawn-p [hatch-walkers 1 [set origin [who] of myself]]
]
```

For the computational analysis part, it is possible to check the calculations with $debug$ command. The idea is to clear all other cells in the lattice and only demonstrate seven of them for simplicity. We can enter the number of particles from the user interface directly; debug-c is center cell and the others are local neighbors. We may observe any of the neighborhood's seven cells by editing the node numbers.



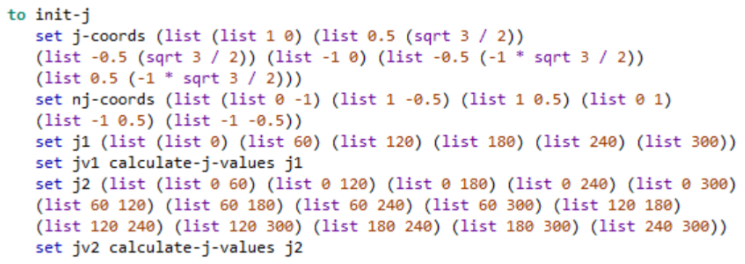
```
to debug
  setup
  ask walkers [die]
  ask node 297 [hatch-walkers debug-c [set origin [who] of myself]]
  ask node 323 [hatch-walkers debug-j1 [set origin [who] of myself]]
  ask node 298 [hatch-walkers debug-j2 [set origin [who] of myself]]
  ask node 272 [hatch-walkers debug-j3 [set origin [who] of myself]]
  ask node 271 [hatch-walkers debug-j4 [set origin [who] of myself]]
  ask node 270 [hatch-walkers debug-j5 [set origin [who] of myself]]
  ask node 296 [hatch-walkers debug-j6 [set origin [who] of myself]]
  ask walkers [pd set color [255 255 255] set size 0.5]
  update-colors
end
```

According to our set-up for debug, the following command, $debug - step$, provides the moving of particles by one step. Again, one can change the center cell with its coordinates to see related configurations.



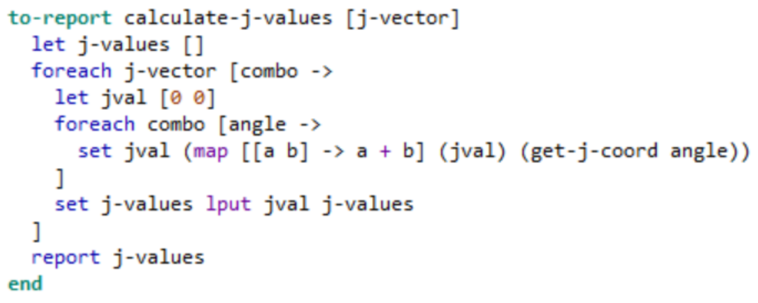
```
to debug-step
  ask node 297 [
    hex-walk
  ]
  let walker-list node-occupants 11 14
  ask walker-list [facexy (xcor + item 0 chair) (ycor + item 1 chair)
  setxy (xcor + item 0 chair) (ycor + item 1 chair)]
  update-colors
  tick
end
```

For the initial set up of direction vectors J_1, J_2, J_3, J_4, J_5 and J_6, it is essential to combine the coordinates with NetLogo coordinates, because the hexagonal turtles are not equilateral, so turtles are not moving slide away from our angle list.



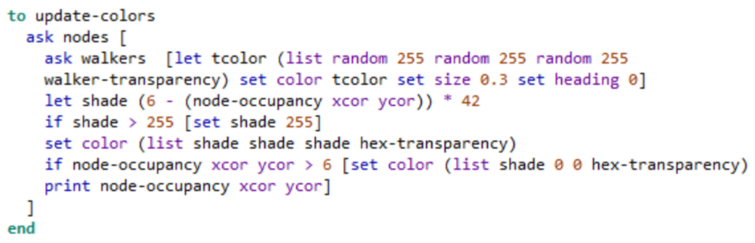
```
to init-j
   set j-coords (list (list 1 0) (list 0.5 (sqrt 3 / 2))
   (list -0.5 (sqrt 3 / 2)) (list -1 0) (list -0.5 (-1 * sqrt 3 / 2))
   (list 0.5 (-1 * sqrt 3 / 2)))
   set nj-coords (list (list 0 -1) (list 1 -0.5) (list 1 0.5) (list 0 1)
   (list -1 0.5) (list -1 -0.5))
   set j1 (list (list 0) (list 60) (list 120) (list 180) (list 240) (list 300))
   set jv1 calculate-j-values j1
   set j2 (list (list 0 60) (list 0 120) (list 0 180) (list 0 240) (list 0 300)
   (list 60 120) (list 60 180) (list 60 240) (list 60 300) (list 120 180)
   (list 120 240) (list 120 300) (list 180 240) (list 180 300) (list 240 300))
   set jv2 calculate-j-values j2
```

After defining the coordinates for directional vectors, the report for $calculate-j-values$ command gives the vectorial sum of J-vectors. Here, this command is used to start a reporter procedure. The main body of the procedure is stated with report, to report a value for the related procedure.



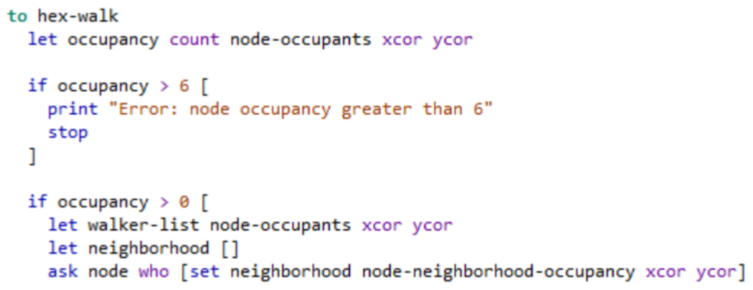
```
to-report calculate-j-values [j-vector]
  let j-values []
  foreach j-vector [combo ->
    let jval [0 0]
    foreach combo [angle ->
      set jval (map [[a b] -> a + b] (jval) (get-j-coord angle))
    ]
    set j-values lput jval j-values
  ]
  report j-values
end
```

We used the same logic and command, G-tendency vector, the raw transitional probabilities, the occupied seats for each seat configuration, the position in angles list that corresponds to index of j-coordinates and nj-coordinates. Since the raw transitional probabilities are not in normalized form, with report procedure, the weighted probability list was computed and joined vectors lists were created. Finally, for the occupancy of a cell at certain coordinates and for the turtle directions, the $to - report$ command was used. To set the node colors with their shades, we used $update - colors$. In this code, the slider for walker-transparency is installed. The corresponding shade-coding states that the higher number of particles, the darker the nodes get.



```
to update-colors
  ask nodes [
    ask walkers  [let tcolor (list random 255 random 255 random 255
    walker-transparency) set color tcolor set size 0.3 set heading 0]
    let shade (6 - (node-occupancy xcor ycor)) * 42
    if shade > 255 [set shade 255]
    set color (list shade shade shade hex-transparency)
    if node-occupancy xcor ycor > 6 [set color (list shade 0 0 hex-transparency)
    print node-occupancy xcor ycor]
  ]
end
```

The moving procedure starts with $hex - walk$, which defines occupancy of nodes at each local neighborhood and does all the calculations. Also in this part, we set up the maximum number of particle in each cell as six. The weighted probabilities for seating configurations are used here to pick one of them.

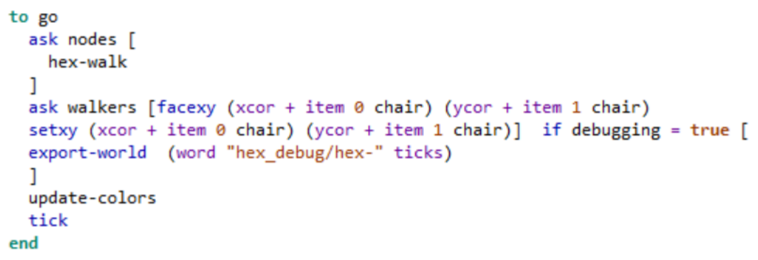


```
to hex-walk
  let occupancy count node-occupants xcor ycor

  if occupancy > 6 [
    print "Error: node occupancy greater than 6"
    stop
  ]

  if occupancy > 0 [
    let walker-list node-occupants xcor ycor
    let neighborhood []
    ask node who [set neighborhood node-neighborhood-occupancy xcor ycor]
```

After setting all configurations, the final step is that each particle moves by one step in the direction it is pointing at.



```
to go
  ask nodes [
    hex-walk
  ]
  ask walkers [facexy (xcor + item 0 chair) (ycor + item 1 chair)
  setxy (xcor + item 0 chair) (ycor + item 1 chair)]  if debugging = true [
  export-world  (word "hex_debug/hex-" ticks)
  ]
  update-colors
  tick
end
```

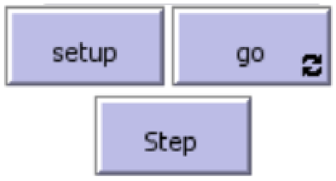
If necessary, a command can be commented out by ";" within the code. However, it is possible to have an error since the program may not run properly without certain commands.

A5. Stochasticity

The stochasticity for agent-based modeling is defined in the code with *rnd:weighted-one-of-list*. According to this command, after computing each configuration probability, the computer makes a random selection from that probability list. Due to those weighted probabilities, the highest probabilities are more likely to be chosen, however there is also a little chance for one of the others to be chosen. Additionally, in the case of having equal highest probabilities for configuration of particles, the computer is instructed to randomly pick a configuration, as defined by *rnd:weighted-one-of-list*. Finally, according to selection, every particle moves one step along the direction it is facing.

A5. Running Code

To run the code, we need to regulate the sliders. Click the *Set up* button and follow by *go*. It is possible to add *step* button for observing the change gradually. For forever running, we need to right click on *go* and edit, then select the forever option.



To stop and start the simulation, we click the *go* button. Here, time is discrete and counted by ticks, which is set above the world.

A7. The Code

```
extensions [table rnd]
breed [nodes node]
breed [walkers walker]
walkers-own [origin chair]
globals [occupied-nodes vectors v j-coords nj-coords
j1 j2 j3 j4 j5 j6 jv1 jv2 jv3 jv4 jv5 jv6]

to setup
  clear-all
  set-default-shape nodes "hex"
  foreach sort patches [ p ->
  ask p [
  set pcolor white sprout-nodes 1
  [let hex-color (list 100 100 100 hex-transparency)
  set color hex-color
  set size 1.2
  if pxcor mod 2 = 0
  [set ycor ycor - 0.5 ] ] ]
  ]
```

```
   ask nodes [
    if random-float 1 < spawn-p [hatch-walkers 1 [set origin
    [who] of myself]]
    if random-float 1 < spawn-p [hatch-walkers 1 [set origin
    [who] of myself]]
    if random-float 1 < spawn-p [hatch-walkers 1 [set origin
    [who] of myself]]
    if random-float 1 < spawn-p [hatch-walkers 1 [set origin
    [who] of myself]]
    if random-float 1 < spawn-p [hatch-walkers 1 [set origin
    [who] of myself]]
    if random-float 1 < spawn-p [hatch-walkers 1 [set origin
    [who] of myself]]
  ]

  init-j

  update-colors

  reset-ticks
end

to debug
  setup
  ask walkers [die]
  ask node 297 [hatch-walkers debug-c [set origin [who] of myself]]
  ask node 323 [hatch-walkers debug-j1 [set origin [who] of myself]]
  ask node 298 [hatch-walkers debug-j2 [set origin [who] of myself]]
  ask node 272 [hatch-walkers debug-j3 [set origin [who] of myself]]
  ask node 271 [hatch-walkers debug-j4 [set origin [who] of myself]]
  ask node 270 [hatch-walkers debug-j5 [set origin [who] of myself]]
  ask node 296 [hatch-walkers debug-j6 [set origin [who] of myself]]
  ask walkers [pd set color [255 255 255] set size 0.5]
  update-colors
end

to debug-step
  ask node 297 [
    hex-walk
  ]
  let walker-list node-occupants 11 14
  ask walker-list [facexy (xcor + item 0 chair)
  (ycor + item 1 chair) setxy (xcor + item 0 chair)
  (ycor + item 1 chair)]
  update-colors
  tick
end

to init-j
   set j-coords (list (list 1 0) (list 0.5 (sqrt 3 / 2))
   (list -0.5 (sqrt 3 / 2)) (list -1 0) (list -0.5 (-1 * sqrt 3 / 2))
(list 0.5 (-1 * sqrt 3 / 2)))
   set nj-coords (list (list 0 -1) (list 1 -0.5) (list 1 0.5)
   (list 0 1) (list -1 0.5) (list -1 -0.5))
```

```
set j1 (list (list 0) (list 60) (list 120) (list 180) (list 240)
(list 300))
set jv1 calculate-j-values j1
set j2 (list (list 0 60) (list 0 120) (list 0 180) (list 0 240)
(list 0 300) (list 60 120) (list 60 180) (list 60 240)
(list 60 300) (list 120 180) (list 120 240) (list 120 300)
(list 180 240) (list 180 300) (list 240 300))
set jv2 calculate-j-values j2
set j3 (list (list 0 60 120) (list 0 60 180) (list 0 60 240)
(list 0 60 300) (list 0 120 180) (list 0 120 240)
(list 0 120 300) (list 0 180 240) (list 0 180 300)
(list 0 240 300) (list 60 120 180) (list 60 120 240)
(list 60 120 300) (list 60 180 240) (list 60 180 300)
(list 60 240 300) (list 120 180 240) (list 120 180 300)
(list 120 240 300) (list 180 240 300))
set jv3 calculate-j-values j3
set j4 (list (list 0 60 120 180) (list 0 60 120 240)
(list 0 60 120 300) (list 0 60 180 240) (list 0 60 180 300)
(list 0 60 240 300)
(list 0 120 180 240) (list 0 120 180 300) (list 0 120 240 300)
(list 0 180 240 300) (list 60 120 180 240) (list 60 120 180 300)
(list 60 120 240 300) (list 60 180 240 300) (list 120 180 240 300))
set jv4 calculate-j-values j4
set j5 (list (list 0 60 120 180 240) (list 0 60 120 180 300)
(list 0 60 120 240 300) (list 0 60 180 240 300)
(list 0 120 180 240 300) (list 60 120 180 240 300))
```

```
  set jv5 calculate-j-values j5
  set j6 (list (list 0 60 120 180 240 300))
  set jv6 calculate-j-values j6
end

to-report calculate-j-values [j-vector]
  let j-values []
  foreach j-vector [combo ->
    let jval [0 0]
    foreach combo [angle ->
      set jval (map [[a b] -> a + b] (jval) (get-j-coord angle))
    ]
    set j-values lput jval j-values
  ]
  report j-values
end

to-report calculate-g-value [neighborhood]
  let n 0
  let gx 0
  let gy 0
  foreach j-coords [j ->
    set gx gx + item n neighborhood * item 0 j
    set gy gy + item n neighborhood * item 1 j
    set n n + 1
  ]
  report (list (gx) (gy))
```

```
end

to-report calculate-transitional-values [g jvals]
  let t []
  let gx item 0 g
  let gy item 1 g
  foreach jvals [j ->
    let jx item 0 j
    let jy item 1 j
    set t lput (exp α( * (gx * jx + gy * jy))) t
  ]
  report t
end

to-report get-seats [t]

  let i 0
  let chairs []
  let c -1
  while [i < length t] [
    set c item i t
    set chairs lput (get-nj-coord c) chairs
    set i i + 1
  ]
  report chairs
end
```

```
to-report get-j-coord [query]
  let angles [0 60 120 180 240 300]
  if query > 0 and query < 7[
     set query item (query - 1) angles
  ]
  let index position query angles
  report item index j-coords
end


to-report get-nj-coord [query]
  let angles [0 60 120 180 240 300]
  if query > 0 and query < 7[
     set query item (query - 1) angles
  ]
  let index position query angles
  report item index nj-coords
end

to update-colors
  ask nodes [
    ask walkers  [let tcolor (list random 255 random 255 random 255
    walker-transparency) set color tcolor set size 0.3 set heading 0]
    let shade (6 - (node-occupancy xcor ycor)) * 42
    if shade > 255 [set shade 255]
    set color (list shade shade shade hex–transparency)
    if node–occupancy xcor ycor > 6 [set color
    (list shade 0 0 hex–transparency)
    print node–occupancy xcor ycor]
  ]
end

to–report get–weighted–list [t]
  let sum–t sum t
  let weighted–list map [ a –> a / sum–t ] t
  report weighted–list
end

to–report join–vectors [a b]
  let joined []
  let pair []
  (foreach a b [ [i j] –> set pair []
  (set pair lput i pair) (set pair lput j pair)
  set joined lput pair joined])
  report joined
end

to hex–walk
  let occupancy count node–occupants xcor ycor

  if occupancy > 6 [
    print "Error: node occupancy greater than 6"
    stop
  ]

  if occupancy > 0 [
    let walker-list node-occupants xcor ycor
    let neighborhood []
    ask node who [set neighborhood node-neighborhood-occupancy
    xcor ycor]


    ifelse sum neighborhood > 0 [


      let g calculate-g-value neighborhood

      let t []
      let j-op []


      if occupancy = 1 [
        set t calculate-transitional-values g jv1
        set j-op j1
      ]
      if occupancy = 2 [
        set t calculate-transitional-values g jv2
        set j-op j2
      ]
```

```
if occupancy = 3 [
  set t calculate-transitional-values g jv3
  set j-op j3
]
if occupancy = 4 [
  set t calculate-transitional-values g jv4
  set j-op j4
]
if occupancy = 5 [
  set t calculate-transitional-values g jv5
  set j-op j5
]
if occupancy = 6 [
  set t calculate-transitional-values g jv6
  set j-op j6
]


let seating-configuration []
let weighted-p get-weighted-list t
let joined-p join-vectors j-op weighted-p



let draw []
if length weighted-p > 1
[set draw first rnd:weighted-one-of-list
joined-p [ [p] -> last p ]]
```

```
      ifelse length weighted-p > 1
      [set seating-configuration get-seats draw]
      [set seating-configuration get-seats item 0 j-op]



      let w []
      ask walker-list [set w lput self w]



      let i 0
      foreach w [ this-walker ->
        ask this-walker [set chair (item i seating-configuration)]
        set i i + 1
      ]



    ][
      ask walker-list [set chair [0 0]]
    ]

  ]
end
```

```
to go
  ask nodes [
    hex–walk
  ]
  ask walkers [facexy (xcor + item 0 chair) (ycor + item 1 chair)
  setxy (xcor + item 0 chair) (ycor + item 1 chair)]
  if debugging = true [
  export–world  (word "hex_debug/hex–" ticks)
  ]
  update–colors
  tick
  if ticks > 100 [stop]
end



to–report node–occupancy [centerx centery]
  report count walkers with [xcor = (centerx) and ycor = (centery)]
end

to–report node–occupants [centerx centery]
  report walkers with [xcor = (centerx) and ycor = (centery)]
end

to–report node–neighborhood–occupancy [centerx centery]
  let n1 count walkers–at  0.0 –1.0
  let n2 count walkers–at  1.0 –0.5
  let n3 count walkers–at  1.0  0.5
  let n4 count walkers-at  0.0  1.0
  let n5 count walkers-at -1.0  0.5
  let n6 count walkers-at -1.0 -0.5
  report (list n1 n2 n3 n4 n5 n6)
end
```