***Simulation of Self-organizing Behaviors in Cells***

**Jie Lian, Lu Yang, and Geyang Zhang**

**Introduction**

Self-organization is a collection of individual behaviors, where each unit in the set reacts by the feedback from the local environment and form an overall pattern (Witzany, 2014). Self-organization is also an important feature in cells. Therefore, the idea of simulating the cell growth based on self-organization of basic behaviors naturally arises (Xie, 2018).

In this project, we designed a simulator to mimic cell growth pattern forming by incorporating behaviors of cell-cell and cell-environment interactions. In the simulation, we use the combination of four basic behaviors of growth, death, attraction, and repulsion to mimic the dynamics of cell growth. Each individual cell determines its behavior by the surrounding cells and environment. The cell-cell interaction closely relates to its surrounding density. A cell with low surrounding density attracts other cells and starts growing while a cell with high surrounding density repels other cells and dies. The cell-environment interaction is achieved by introducing zone that can either stimulate or inhibit cell growth.

Besides mimicking the behaviors of single type of cells, we expand the project further in simulating a “Source and Sink” model (Rudge *et al.*, 2012). There are two types of cells in the model, source and sink. Source cells, expressing blue fluorescence, can secrete signal that induces the sink cell to express yellow fluorescence. When two clusters of cells approaches to each other, the transmission of signals starts.

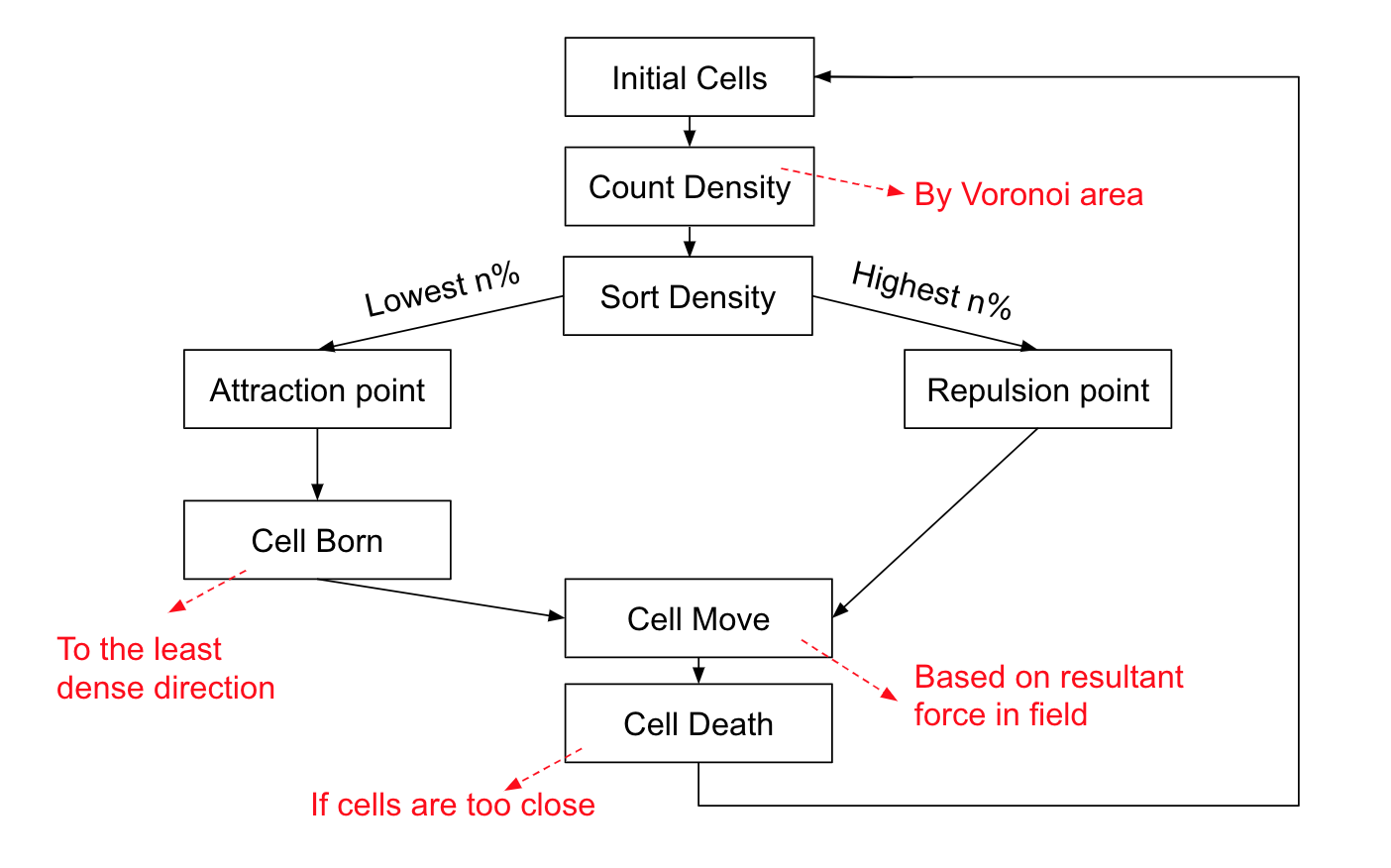
**Installation**

Download thefile and unzip it to the **go/src** directory. To run the simulator, first change the directory of the simulator, and compile all the packages (**go build**) in the directory. On Mac, the command for the simulator follows the format:

**./cgsimu COMMAND Option**

There are three options for the **COMMAND**. The first option is **OneCluster**. Taking the command, the simulator reads the stored inputs in the *OneClusterOutputs.txt*, and generates a gif, named *OneCluster.gif*, that records the simulation of one cluster of cells according to the inputs. **OPTION** available for OneCluster are **Voronoi** and **CountDensity**, which are two ways of counting the surrounding density of a cell. The second option is **AutoGenerate**. Under this command, the simulator randomly generates inputs, stores as *input.txt*, and simulates the growth pattern of one cluster of cells according to the generated inputs, and the output gif is named *AutoGenerate.gif*. **OPTION** available for AutoGenerate are **Voronoi** and **CountDensity**. The third option is **TwoCluster**. The simulator reads the stored inputs in the *TwoClusterOutputs.txt*, and generates a gif, named *TwoClusterSS.gif*, recording the simulation of the source and sink model according to the inputs. **NO OPTION** available for TwoCluster, and the default counting method is CountDensity.

**Method**



**Figure 1. Workflow of the Basic Model**. The workflow reflects functions that are called during one generation.

**Basic Model**

The basic model (Figure 1) takes in initial cells and generate a new slice of cells based on the previous one in the new board. For each cell, we first sorting them by voronoi area from largest to smallest, or simply count number of cells in each cell’s search radius. Then, the first n% cell (defined by “birthrate”) become attraction point, the last n’% cell (defined by “deathrate”) become repultion point. The attraction point also randomly generate ten cells in its “birthradius”, and select the one with least density to be the final new born cell. Next, all cells move based on the resultant force it senses in its “searchraidus”. The repulsion force is defined as:

and the attraction force is defined as:

If two cells are moved too close to each other, the death function will eliminate the one with larger density. The new slice of cells is now ready to simulate the next generation.

**Voronoi**

A Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane and can be used to determine the density of every cell in the our board. To achieve our purpose to “evenly” dividing the area of the whole board to each cell, we implement Fortune’s algorithm to create the dividing edges to create such diagram so that the Voronoi area of each cell existing on the current board can be calculated.

The main idea of Fortune’s algorithm is to compute the Voronoi diagram by orthogonally moving a sweeping line along one direction of the plane. During this “sweeping process”, the site event where a new parabola will be created and added into the beachline when a site is scanned by the sweeping line. Generally, two neighbour parabolas will squeeze each other to start an edge we want. A parabola will entirely disappear when sweeping line proceeds a certain position called circle event. By using a queue storing these site events and circle events, we modify the tree storing the growing parabola and edges and put generated complete edges into a slice which will later be used to calculate the density of each cell on the current board.

**Environmental Interaction**

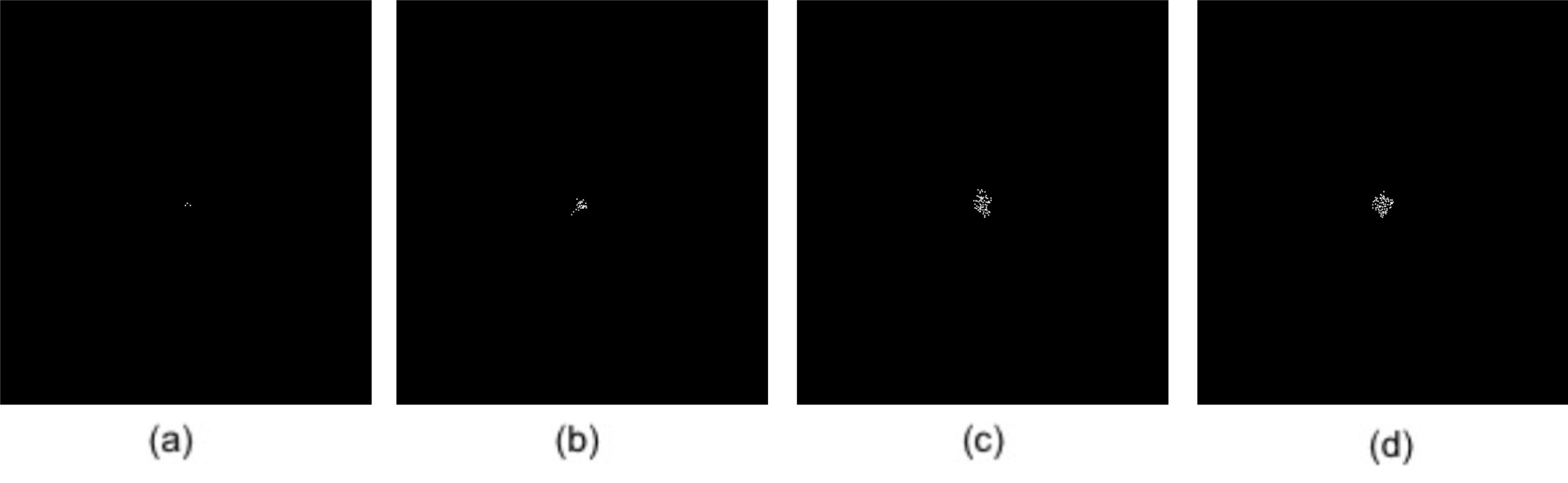
Environmental interaction was observed by two types of simulation. The first is barrier, include board boundary and “Maze”, the collection of rectangles that cells take as entities and can not grow in. The second is “Zone”, which are circle areas that has either inhibit or promote effects on cell growth. The strength of zones are decimals from 0 to 1, if inhibit, the value will be negative; if promote, the value will be positive. A new generated cell has a initial value of 1.0 to survive, this value takes the sum of zones strength it located in and based on the possibility to decide if to maintain this cell.

**Source and Sink Simulation**

The source and sink model is similar to the basic model with few changes in inputs. Initial cells are randomly assigned into either source or sink cluster. For each cluster of cells, cells grow independently, based on the inputs of birth radius, death radius, birth rate, and death rate, using the basic model. Once the distance of two clusters is less than a default value (100.0, in this case), sink cells begin to receive the signal from source cell, and the amount of signal is estimated using the number of source cells currently present. Besides, to mimic the time delay in the signal transduction, in each generation, small portion of the total signal is saved for the next generation, and the majority is used to increase the signal level in each sink cell. The signal level in sink cells is visualized by different levels of yellowness. Higher the signal level, more yellowish the cell shows in the output.

**Result**

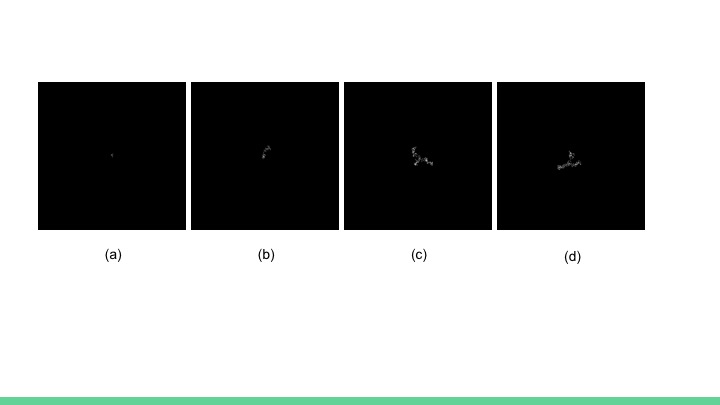
**Basic model (Voronoi)**

****

**Figure 2. Sorting density by count cell’s Voronoi area.** Search radius 20 and birth radius 5. (a) 10 generation; (b) 50 generation; (c) 70 generation; (d) 90 generation.

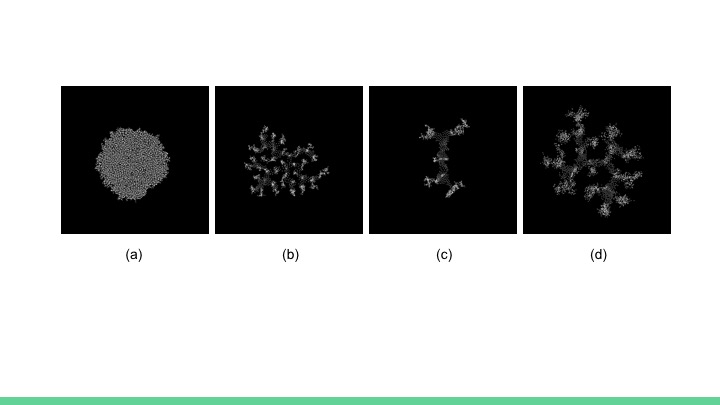
In our primary design with Voronoi diagram to calculate density, the generated patterns is quite similar to what we expected. However there are still some problems when implementing this part. 1.) the procedure will stuck when numGens (number of generations) is more than 100. This might be result from func GetArea which takes O(n3) to generate area by using Heron’s formula with input as generated Voronoi edges. So the current Voronoi part is unable handle up to 100 generations. 2) it can be answered why the cells on the pattern generated by this way tend to contract to the central point.

**Basic model (Count density)**

****

**Figure 3. Sorting density by count cells in search radius.** Search radius 20 and birth radius 5. (a) 10 generation; (b) 50 generation; (c) 70 generation; (d) 90 generation.

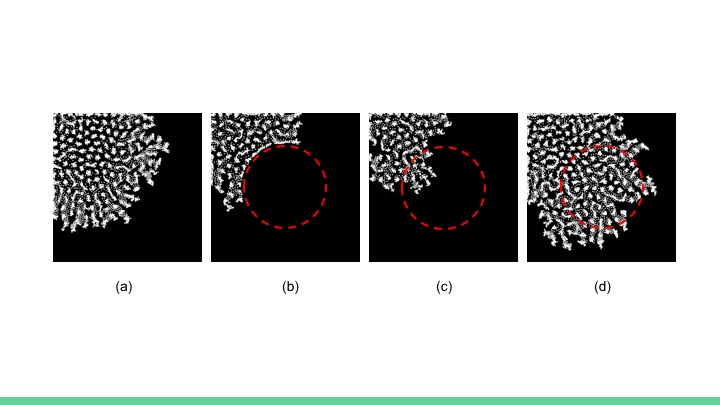
When simply sorting cells based on their density, the run time are 1.4ms, 16.4ms, 53.4ms, 65.1ms respectively for 10, 50, 70 and 90 generation.

****

**Figure 4. Different growth pattern.** (a) Search radius 5, birth radius 5; (b) Search radius 25, birth radius 5; (c) Search radius 50, birth radius 5; (d) Search radius 50, birth radius 10.

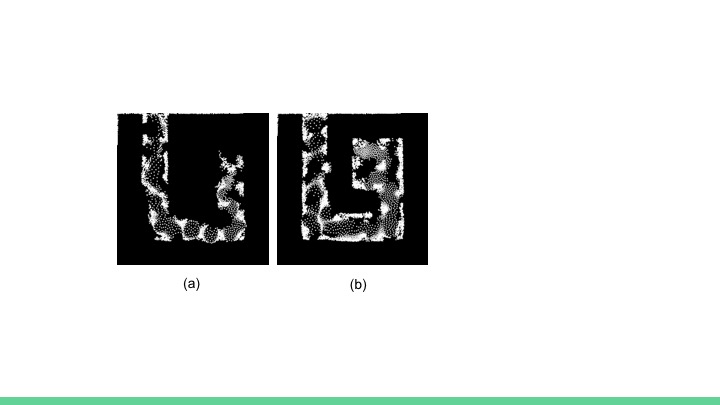
Search radius implies the maximum distance that one cell can affect another; birth radius is the surrounding area that one cell can grow a new cell in. By adjusting these two parameters, we can generate very different growth patterns. When two values are close, cells distribute evenly; but when search radius increases, cells tends to born in place that are more void, showing a strong direction preference, and the most dense part of the population concentrated on outer side that are actively producing new cells. When birth radius increases, the entity expand faster but cells connections are more loose.

**Environmental interaction**



**Figure 5. Environment interaction (resistant zone**) search radius = 20, birth radius = 5, zones indicting by red circle (a) No zone added; (b) Inhibit zone, selection strength = -1; (c) Inhibit zone, selection strength = -0.5; (d) Promote zone, selection strength = 1.

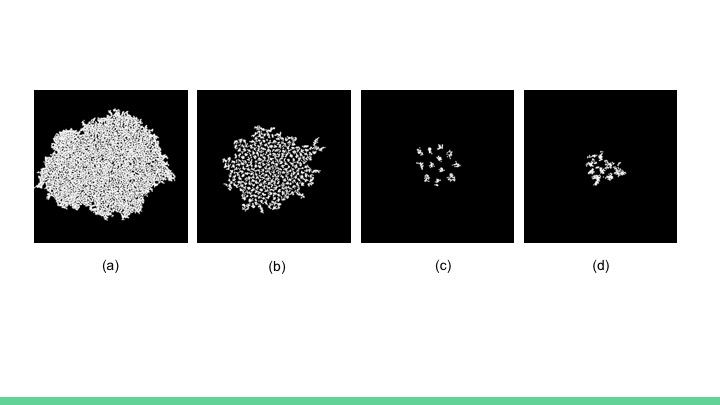
By adding zones with different selection strength on new born cells, we can have different growth results for the same parameter. For a zone of entirely negative selection effect, no cells can grow on the zone; however when the negative selection strength is only 0.5, cell can grow on the zone but the expanding speed is inhibited; when the zone have a positive selection effect, cells grow faster on the zone and the population has a direction preference for the promotion area.



**Figure 6. Environment interaction (maze)** Search radius = 50, birth radius = 10**,** 300 generation(a) Without any zone; (b) With four promote zones one each corner;

Since we have seen that the higher the search radius is, cells growing faster alone one direction, we build up a maze for cells and see how they perform. With a larger birth radius of 10, the cells grow to the maze destination without necessarily fill in every space they cover. Further, adding promotion zones at corner helps cells find their way faster.

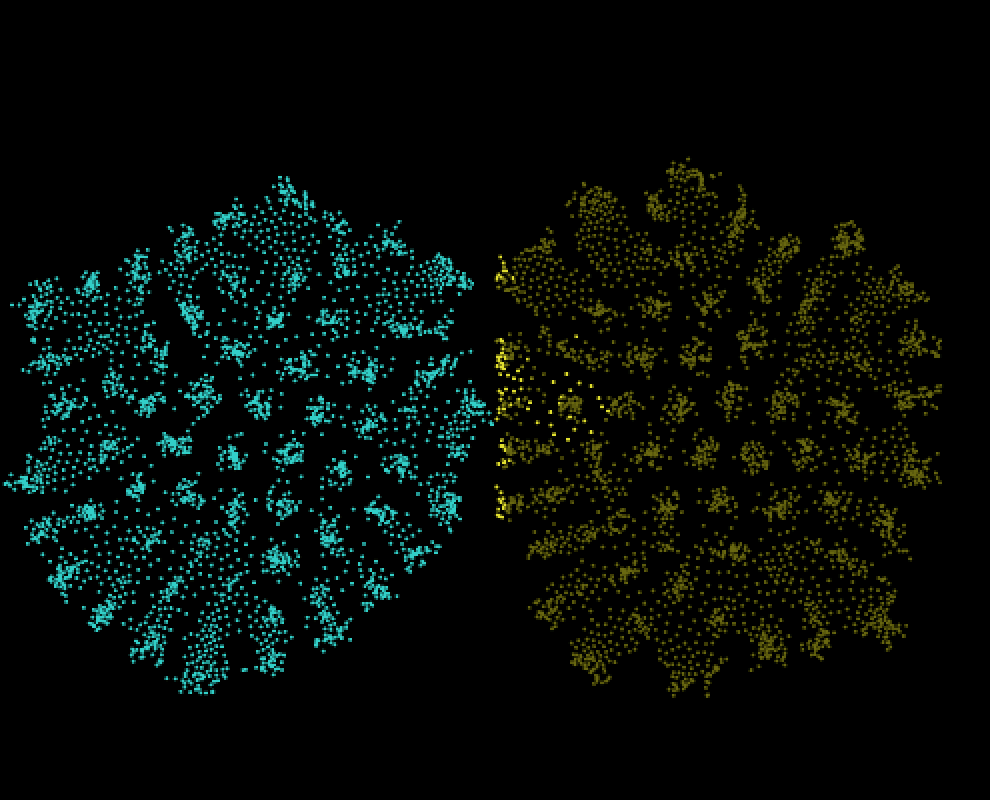
**Death and Birth Curve**



**Figure 7. Death curve and birth curve** (a) Search radius 5, birth radius 5, with death curve; (b) Search radius 10, birth radius 5, with death curve; (c) Search radius 25, birth radius 5, with death curve; (d) Search radius 25, birth radius 5, with growth curve + death curve.

In above, or death function only eliminate cells that are too close. Now we add a property of lifespan and let the death curve has a 0.5 possibility of cell death at around lifespan = 20. When Search radius close to birth radius, this property didn’t change much about the final configuration. However when search radius is larger than birth radius, the death curve break the connection between cell clusters and the cells automatically divided into several clusters. To fix the lacking of cells in between each clusters, we further add a property of growth curve that take a function of density and convert a growth score that has a maximum possibility of generate a new cell at density = 3.

**Two cluster cells interaction**

****

**Figure 8. Simulation of Source and Sink Model**. The blue cluster represents source cells, and the yellow cluster represents sink cells. Initial cell number 4, number of generation 150, search radius 20.0, birth radius 5.0, death radius 1.0, birth rate 0.3, death rate 0.1, width 500.0

When simulating the source and sink model, sink cells that are at the closest boundary to the source cluster start to show a yellowish color. It is clear that yellowish sink cells are most dense at the boundary, and the number decreases as away from the boundary, suggesting that the simulation of signal transmission works correctly with right direction.

**Conclusion and Future Direction**

By our model with four basic behaviors, we successfully generate different growth patterns. The larger area the cell takes feedbacks from, the more clear direction can be seen from the result. On the contrary, if search radius is similar to birth radius, cells tend to expand evenly into the surrounding.

In the interaction with environment, cells can avoid barriers and search for place without block. This is because the area with block has accumulate sufficient cells and no longer count for the active growth point. However in zones, since the birth possibility actually change, there is no cutoff for densities so cells will always continue to grow in the zone. By this property, we can form a partial resistance zone.

In the simulation of source and sink model, sink cells at the boundary did show different signal levels once the distance between two clusters is small. However, our simulation showed a slightly different pattern with the original model (Rudge *et al.*, 2012). The reason behind is that the growth pattern of the original model is static, i.e. once cells generated in the original model, the cells do not move or die. In our model, cells are more dynamic with a variety of behaviors.

For future studies, we first want to optimize the runtime. Currently, we can run a 300 generation of simulation with birth radius 5 and search radius 20 in about 30 seconds. However, when birth radius or search radius raise, this time could increase to several minutes for the exponential increase fitting. We had tried store neighers for each cell and using BFS to traverse this tree structure, but still estimating if a cell was visited could use O() time and the actual run time increases by ½ parameters described above. Similarly, Voronoi failed to be efficient for the frequent ranging of the whole board. Future solutions may lays in parallel running and adjusting parameters since a few function such as direction selection and death elimination may not use the information of all cells but only adjacent cells.

Besides, we would like to introduce cell competition based on the existed model. The source and sink model only has one-way influence, which only source cells can secrete signals to activate the expression of sink cells. In the future, two clusters of cells that have mutual influence can be introduced. Moreover, different types of cells can have different properties, such as type A cells have higher birth rates but lower search radius than type B cells. It would be fun to see how cells compete each other and which type of cells can survive longer in the end?

**Reference**

Xie, Y. F (2018). Cellular Growth: Self-Organizing of Basic Behaviours. *Grasshopper*. https://www.grasshopper3d.com/video/cellular-growth-self-organizing-of-basic-behaviours

“Fortune’s algorithm and implementation”. *Algorithm and Stuff*.

http://blog.ivank.net/Fortunes-algorithm-and-implementation.html

Rudge, T. J., Steiner, P. J., Phillips, A. & Haseloff, J (2012). Computational Modeling of Synthetic Microbial Biofilms. *ACS Synthetic Biology* 1 (8), 345-352.

Witzany, G (2014). Biological Self-Organization. *International Journal of Signs and Semiotic Systems*. **3**: 1–11.