# Exploring the Network Attributes of a Physarum Polycephalum Simulation

Tristan Larkin
*Dept. of Computer Science*
*University of New Mexico*
Albuquerque, United States
trlarkin@unm.edu

Jack Wickstrom
*Dept. of Computer Science*
*University of New Mexico*
Albuquerque, United States
jwickstrom@unm.edu

Edgar Zapata
*Dept. of Computer Science*
*University of New Mexico*
Albuquerque, United States
ezapata14@unm.edu

***Abstract*—Physarum Polycephalum is a very special type of slime mold that has complex behaviors for when foraging for nutrients and building transport networks. This paper examines the network properties that come from a computer simulation of Physarum based on the work of Jones [1]. We examine how rapidly the slime finds food sources, the qualities of the network structures and node degree distributions that form, and the sensitivity of the simulation to small variations in initial conditions. Our key findings are: the slime quickly localizes all food sources and builds connected networks spanning them, node degree distributions do not follow a power law, and the networks exhibit high sensitivity to slight changes in initial conditions, a common feature of complex adaptive systems. We analyze how different simulation parameters like particle sensor angle and rotation angle impact the resulting networks. Overall, the Physarum simulation shows the ability of basic local rules to generate dynamic, adaptive networking behavior, with potential applications to decentralized computing and routing solutions.**

## I. Introduction

In this project, we use a Physarum Polycephalum model developed by user Fogleman on GitHub [2]. The algorithm derived from Fogleman is highly resemblant of the one presented by Jones [1]. Similarly, it is a multi-agent model that operates on two "overlapping" grid maps, the data map and the trail map. The 'top' data map is the environment where the many slime agents reside, whereas the 'bottom' trail map contains the information which influences the movement of the slime particles. This information consists of 'temperature' values dictating where slime trails and food exist on the grid.

Physarum particles scan this map using a cone-shaped sensor whose vertex points to the center of the particle and whose base points in the direction the particle was previously moving. The sensor helps guide particles by analyzing the different trail grid map temperature values contained within it and shifting the particle a certain number of grid cells towards the highest temperature value scanned as long as the new grid cell is not occupied by another particle. Once a particle makes a step, it increases the temperature within its newly occupied cell. After each iteration in the simulation (all slime particles have made a step), each trail map cell diffuses (blurs) by spreading out some of its temperatures to neighboring cells.

The behavior and overall nature of the simulation are reliant on its configuration which consists of variables: simulation width, simulation height, number of particles, blur radius, blur passes, zoom factor, particle sensor angle, particle sensor distance, particle rotation angle, particle step distance, particle deposition amount, and the simulation decay factor. Definitions of what what each configuration variable entails is as follows:

- Simulation width & height: size dimensions of the simulation environment.
- Number of Particles: Number of particles established in the simulation.
- Blur Radius: How much temperature of a cell disperses.
- Blur Passes: Number of times a blur pass is performed
- Zoom Factor: How zoomed in the simulation is
- Sensor Angle: Angle between the far left or far right of the sensor from the forward position.
- Sensor Distance: Size of the region in front of the particle that the sensor can see.
- Rotation Angle: Angle at which the particle is able to rotate.
- Step Distance: How far a particle is allowed to move.
- Deposition Amount: Temperature deposited by a particle.
- Decay Factor: Rate at which temperature diffuses/decays.

The similar simulation used by Jones [2] posed some interesting findings. By modifying the configuration parameters, Jones demonstrates how the simple local interactions, specifically chemotaxis, can lead to the formation of complex and dynamic transport networks, observable in the real Physarum Polycephalum. Furthermore, behaviors such as network formation, contraction, bifurcation, and repair were all highlighted by the study. These findings help suggest that these biological patterns can inform the development of new computation strategies.

Influenced by Jones' findings, in this study, we seek to further understand the network behaviors exhibited by the simulation, the time it takes to create a network, the distribution of node degrees throughout the network, and the impact of initial conditions on the network.

## II. Methodology

We explored the networks that occurred from a variant of the Physarum slime algorithm presented in the Jones paper [1]. We started our exploration by building off the code by Michael Fogleman [2]. This implementation did not include

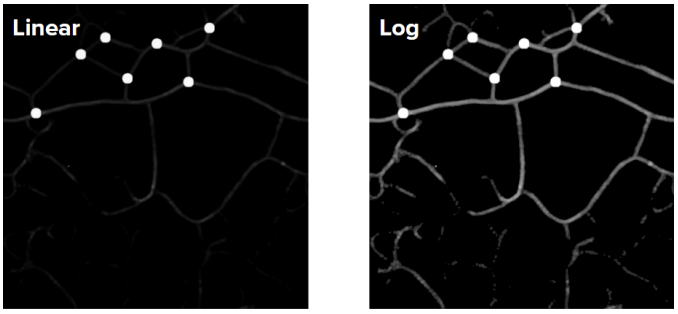


Fig. 1. Example of how viewing images in logarithmic scale can reveal hidden attributes. This technique applies a log function to the magnitudes of the pixels.

| Config Setting | Our Base Config | Jones' Config |
|---|---|---|
| simulation width: | 512 | 200 |
| simulation height: | 512 | 200 |
| numParticles: | $2^{17}$ | 1200-6000 |
| blurRadius: | 1 | 3 |
| blurPasses: | 2 | N/A |
| zoomFactor: | 1 | N/A |
| SensorAngle: | 45° | 45° or 22.5° |
| SensorDistance: | 8 | 9 |
| RotationAngle: | 45° | 45° |
| StepDistance: | 1 | 1 |
| DepositionAmount: | 2 | 5 |
| DecayFactor: | 0.05 | 0.1 |

TABLE I

"OUR BASE CONFIG" IS THE SET OF PARAMETERS WE USED AS OUR BASELINE WHEN DOING TESTS. IT IS BASED ON "JONES' CONFIG" WHICH IS THE SET OF STANDARD CONFIGURATIONS USED IN THE JONES PAPER [1].

the collisions discussed in the Jones paper but still produced similar results to the paper in initial testing. One important test was changing the parameters to produce chaotic, labyrinth, island, or mesh results, and the code produced results predicted by Jones when we used similar settings.

Unfortunately, the Foglemen implementation contained no way to place food particles in the simulation. Therefore, we had to fork the repository and implement this ability, which can be found in a Github repository here [3]. To implement food particles, we allow the user to input an image of where they would like food particles to be. Then, the code will continuously deposit food on the trail/heat map every simulation iteration, which encourages the slime to go to these locations.

To perform the tests we wanted to, we had to implement some additional functionality to the code from Fogleman. Food locations were added, which increases the temperature by a large amount in any location they are placed. The food can also be controlled throughout the simulation to see how the slime reacts to changing food locations.

We also have three ways to contain the slime. First is the default as implemented by the original code, which just connects the top and bottom together and the same for the sides. There is also a method used that forces the slime to redirect back into the center when it gets too far out, requiring the slime to remain in a location of interest. Finally, we could put a less powerful food source in some shape on the board. Since the slime will be attracted to food more than the outside, it will not leave that shape, but will still be attracted to stronger food sources and other slime paths.

Most analysis is done by examining the images or videos produced by the simulations. One important technique for examining the images is looking at them in logarithmic scale as demonstrated in figure 1.

We also used a standard set of base conditions that we compared everything against. These conditions were developed based on the standard settings used in the Jones paper, with some alterations that produced better results for our specific simulation. These can be seen in table III. The following section (Findings) will explain the specific ways we explored how networks were built along with the attributes of those networks.

## III. FINDINGS

The work in [1] lays the groundwork for the exploration of what types of networks the Physarum simulation produces and what factors affect the network. Understanding the ways different configurations affect the development of networks can assist later researchers in developing successful algorithms using slime simulations to solve problems.

### *A. How the Slime Finds Food*

Two parameters affect how the slime detects the food points: the sensor angle and sensor offset. We generally set these parameters to 45° and 8 pixels respectively. Figure 2 shows how the sensor offset affects the slime's ability to see the food. While the slime does not specifically recognize the food as a separate object, the food produces enough temperature that the slime will almost always be attracted to food when one of its sensors sees it.

Figure 3 shows the way changing food sizes affect how the slime finds the food. When the food is large (image D), and hence the area that slime particles can be in to find the food is large, the slimes find the food and create structure around it. In the same number of time steps when the food is very small (image A), the slime has not found all the food particles. This matches what we would expect, as it is just a smaller chance for the slime to ever stumble upon the food in its exploration.

Understanding how the slime reacts to the food is important if we want to use the slime to connect points for us. If two separate points are close enough together, we want to understand how small we can make the food sources around them. If the algorithm requires the food sources to be larger than the points are separated, the slime might end up just resolving the two points as one larger point and not act like we want.

### *B. Speed Considerations*

For an algorithm to be useful we need it to be able to run at reasonable speeds. We believed that this slime simulation had the chance to be good at solving problems since it is similar

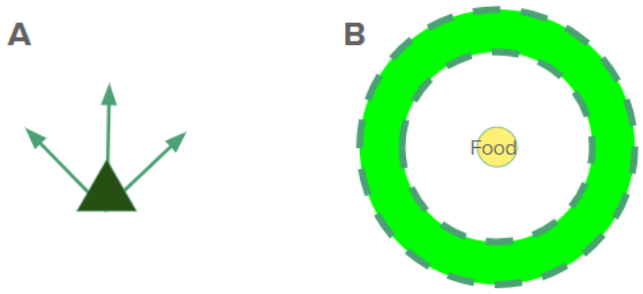


Fig. 2. (A) Slime particles look in three directions for the location with the highest temperature. It specifically looks at the pixels a predefined distance away, called the sensor distance or sensor offset, to determine where to move. (B) This means there is a ring around the center of the food where there is a chance the slime particle will find the food (in green). This example shows a food radius smaller than the radius of the sensor distance, but this is not required. The width of the ring is related to the radius of the food and the radius of the ring is related to the sensor distance.

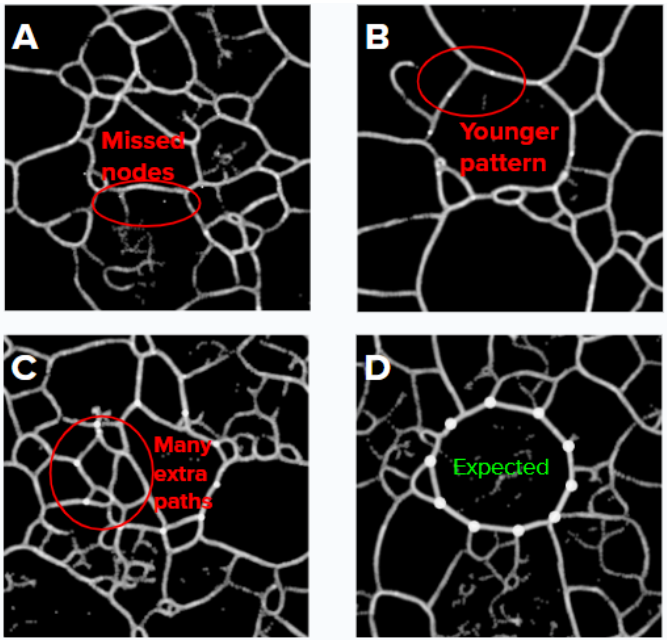


Fig. 3. Changing the size of the food without changing the particle settings produces different results. (A) The food has a 2-pixel radius and the slime did not find all the food locations. (B, C) The food has 4-pixel and 6-pixel radiuses respectively and the slime found all the food particles, but it has not settled into a clean pattern. (D) The food has a 10-pixel radius and has settled into a clear pattern with the shortest paths between each food particle. All simulations were run for the same number of time steps and are displayed where the intensity is log-scaled. The configurations are the same as in table III.

to a parallel terraced scan. While it does not have a clear goal, it is many separate particles that do come to an ”agreement” on where to move. In general, the slimes find the shortest path between nearby food particles and create extra nodes around and inside the circle of particles shown on the left in figure 8. The slime is also a time-developing algorithm so there are interesting characteristics that come into play when the surroundings change.

The number of time steps it took for the slimes to find all the food particles is shown in figure 8. We tested it both with and without the border. It took longer to find all the food without the border, where slimes could travel the whole screen and loop back around. This follows expectations since the border is essentially just decreasing the amount of space needed to search.

Two attractive attributes of this algorithm are that it is adaptive and continuous. When testing how the slime adapts to changing food locations over the course of a single simulation, we found two interesting attributes of the networks built. The first was that even when food sources were deleted, there structure that was around those nodes often persisted for a while. Figure 5 demonstrates an example of this where the structure that was built around a deleted food source persists well after 1000 time steps. This could come from the fact that there is not a mechanism employed to specifically tighten the network. A method for removing strands like this is presented in [1], but it involves weighting slime residue more than food nodes after a certain amount of time, and this was not part of our implementation. The other useful observation was that the slime does not take very long to incorporate new nodes into the network, as shown in figure 6. The chaotic nature of the slimes seems to play to its advantage since it creates smaller strands in locations that don't have a food node but might get one in a changing environment.

Although the slimes are quick to adapt to important changes, our simulations still ran much longer than it would take to just point out by hand the solutions the slimes came up with. The real value of this algorithm will have to be determined by large simulations with thousands of food nodes. We also would like to point out that we used $2^{17}$ particles in comparison to the 1200-6000 particles that were used by the Jones paper. We used more particles to produce better images and have not properly experimented with how few particles are required to produce results, which is important to know if speed is a concern.

### C. Nature of Node Connections

As mentioned in part A, the sensor angle and sensor offset affect the way the slime finds food. However, we also wanted to investigate how node connections were established and if the slime exhibited any sort of power law distribution. To see the impact of configuration sets on this behavior; sensor angle, sensor distance, and rotation angle were individually increased or decreased within our base configuration. To measure how each configuration parameter affected the slime, each was changed one at a time, 6000 iterations were conducted, and

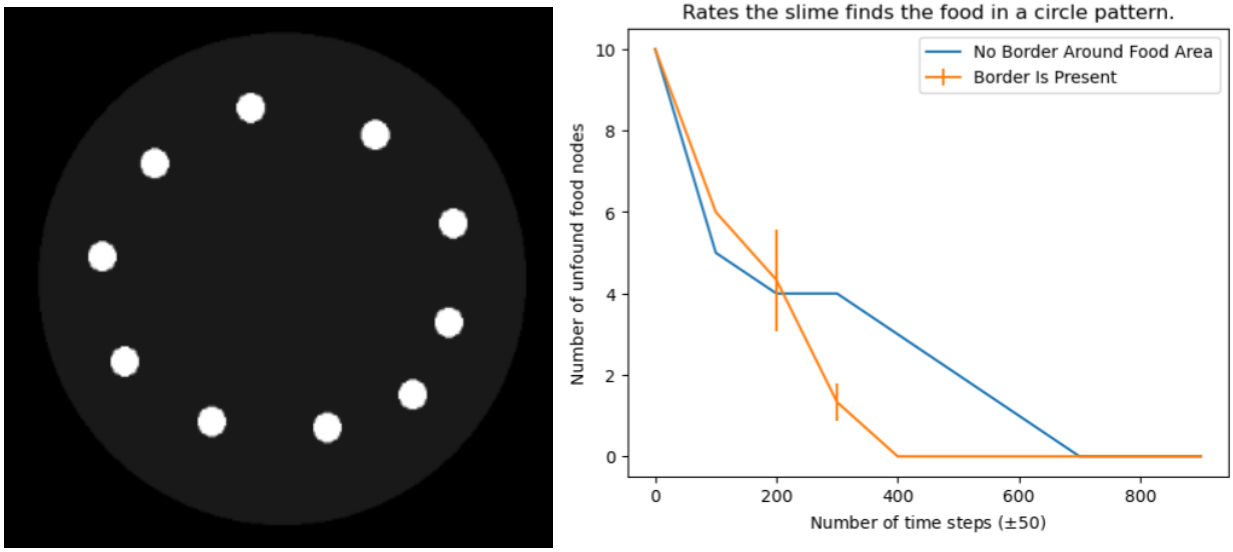


Fig. 4. (LEFT) A food map that distributes 9-pixel radius food blobs around a circle. The Light gray circle is a border that keeps the slime particles inside it by weighting movements to the outside very low. (RIGHT) The speed at which the slime found all the food particles. We defined this as a large number of slimes congregated around the food particle. It does not imply that there was an efficient network involved. The orange plot has error bars where the error is the standard deviation from multiple runs. Blue is the same configuration but without the light gray border circle.

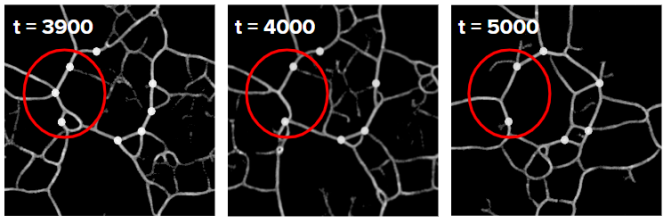


Fig. 5. When a food node is removed, there is a persistent structure from where it started.

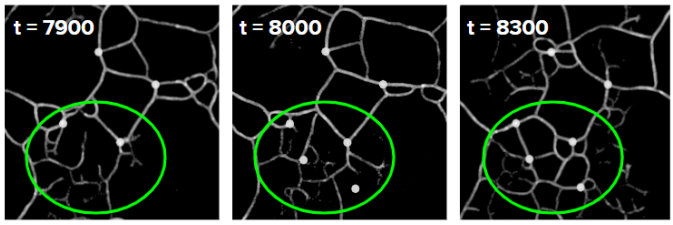


Fig. 6. New food nodes tend to be found quickly as there are slimes all over the board and they tend to get found and incorporated.

| Node Degrees | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Base Config: | 0 | 0 | 7 | 8 | 1 | 0 | 0 |
| SensorAngle Increased: | 4 | 0 | 6 | 4 | 1 | 0 | 0 |
| SensorAngle Decreased: | 0 | 0 | 5 | 19 | 5 | 1 | 0 |
| RotationAngle Increased: | 0 | 0 | 10 | 8 | 1 | 0 | 0 |
| RotationAngle Decreased: | 0 | 0 | 8 | 9 | 0 | 0 | 0 |
| SensorDistance Increased: | 0 | 0 | 2 | 24 | 14 | 0 | 0 |
| SensorDistance Decreased: | 1 | 0 | 8 | 4 | 0 | 0 | 0 |

TABLE II

SHOWS THE DATA REGARDING THE CIRCULAR FOOD NODE LAYOUT. DEPICTS THE NUMBER OF NODES (FOOD & INTERMEDIATE) FOR EACH DEGREE PRESENT IN THE FINAL SIMULATION STEP OF A SIMULATION RAN WITH THE LEFT CONFIGURATION MODIFICATIONS ON THE.

each node in the final state of the algorithm was analyzed. Furthermore, two different food node layouts were used: twelve nodes forming the perimeter of a circle and twelve nodes randomly spread out amongst the grid.

In the simulations, regardless of the configuration parameters, there was a tendency for the development of 'intermediate nodes' within the slime. These nodes were characterized by impactful connections between food node edges which allowed for the attraction of more or the creation of new edges. The degrees of these nodes were also recorded. However, it's important to note, that for a connection to increase a node's degree in the data, it had to be of noticeable strength/temperature in the image and show considerable impact on the behavior of the network (connection over distance or point for multiple connections).

Tables II and III below show the node counts for varying degrees regarding each parameter change and food node layout. Graphing these values reveals some interesting network properties of the slime mold. In general, it was common for

most nodes to be of degree 2 or 3 with lower values for all other degrees. Depending on which configuration parameter was changed, the likelihood of the emergence of new nodes dramatically changed. Typically, increasing the rotation angle, increasing the sensor distance, or decreasing the sensor angle resulted in a large increase in the likelihood of new nodes and connections emerging regardless of the food node placements. On the other hand, decreasing the sensor angle or sensor distance resulted in lower node degree counts and more defined node connections.

Graphing the table values lets us see how spread out the node degree counts for each simulation are. For the sake of

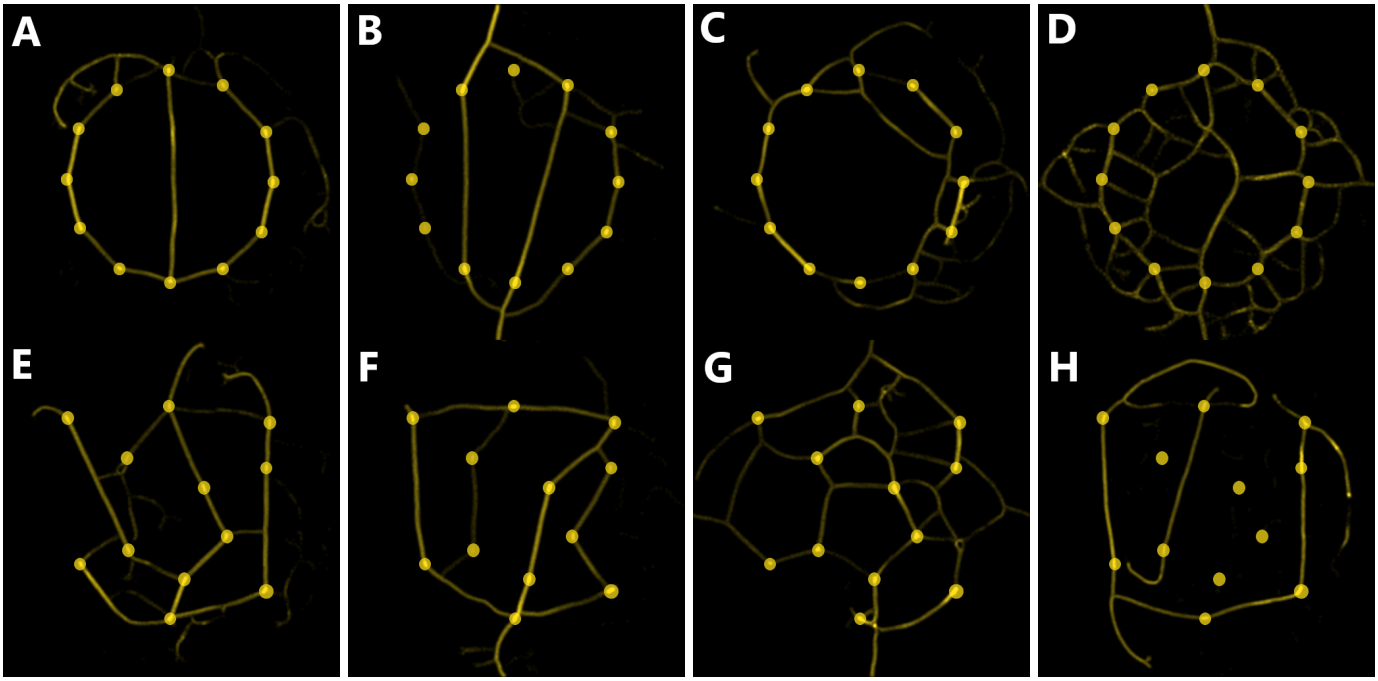


Fig. 7. Depicts the network established after 6000 iterations by 8 different simulations using different configuration sets. Letters A, B, C, and D depict the networks created using our base configuration, increasing the sensor angle, increasing the rotation angle, and increasing the sensor distance respectively; on a grid with a circular food node layout. Letters E, F, G, and H depict the networks created using our base configuration, increasing the sensor angle, increasing the rotation angle, and decreasing the sensor distance respectively; on a grid with a random food node layout.

| **Node Degrees** | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|---|
| Base Config: | 0 | 1 | 6 | 6 | 2 | 1 | 0 |
| SensorAngle Increased: | 0 | 1 | 10 | 3 | 1 | 0 | 0 |
| SensorAngle Decreased: | 0 | 0 | 5 | 16 | 5 | 0 | 0 |
| RotationAngle Increased: | 0 | 0 | 3 | 15 | 4 | 0 | 0 |
| RotationAngle Decreased: | 3 | 0 | 6 | 3 | 1 | 0 | 0 |
| SensorDistance Increased: | 0 | 0 | 3 | 17 | 3 | 0 | 0 |
| SensorDistance Decreased: | 4 | 0 | 6 | 4 | 0 | 0 | 0 |

TABLE III
SHOWS THE DATA REGARDING THE RANDOM FOOD NODE LAYOUT. DEPICTS THE NUMBER OF NODES (FOOD & INTERMEDIATE) FOR EACH DEGREE PRESENT IN THE FINAL SIMULATION STEP OF A SIMULATION RAN WITH THE LEFT CONFIGURATION MODIFICATIONS ON THE.

saving space, figure 8 below only shows 8 of these graphs, but they are well representative of the different 'bell' shaped spreads seen in the data. The respective slime network for each graph can also be seen in figure 7. Analyzing these figures quickly draws us to the realization that the slime network doesn't exhibit a power-law distribution or preferential attachment. It's clear to see from the graphs how lower and higher node degrees were less present within the networks than the middle node degrees.

The lack of a power-law distribution and preferential attachment can be attributed to the way the slime particles work. Since a slime particle can only scan a certain region in front of it, it has the possibility of completely missing a node next to it which might already be of a higher degree. As network connections become more established and stronger in temperature, this effect snowballs and can lead to a lower likelihood of high-degree nodes increasing in degree even more as they might be outshined by greater temperature connections. Even in simulations where a configuration set allows particles to see more around them, this behavior is still present at some level. This is clearly seen in our tests where the sensor angle was increased. Ironically, decreasing the sensor angle resulted in particles being attracted to more nodes which led to the creation of even more intermediate nodes.

Regarding the development of each network before achieving its final state, there's some interesting behavior that must be noted. At some point, regardless of the configuration, each simulation had a step where all nodes were incorporated into the network. Even in the less chaotic simulations, the slime contained a path that connected all nodes. Also, as the simulation progressed, intermediate nodes were created and extinguished. There was no such thing as a 'stationary' intermediate node or an intermediate node that existed throughout the entire simulation. Intermediate nodes were frequently shifting location until either slime particles were pulled away from it or merged with another node. Most intermediate nodes were short-lived but still had an impact on how the network achieved its shape in the final iteration.

### *D. Sensitive Dependence On Initial Conditions*

For our final point of research, we chose to examine the sensitivity of the slime mold simulation on the initial conditions. From the previous experiments, we knew that changing the initial conditions could lead to drastically different final outcomes after the simulation had evolved. However, all the

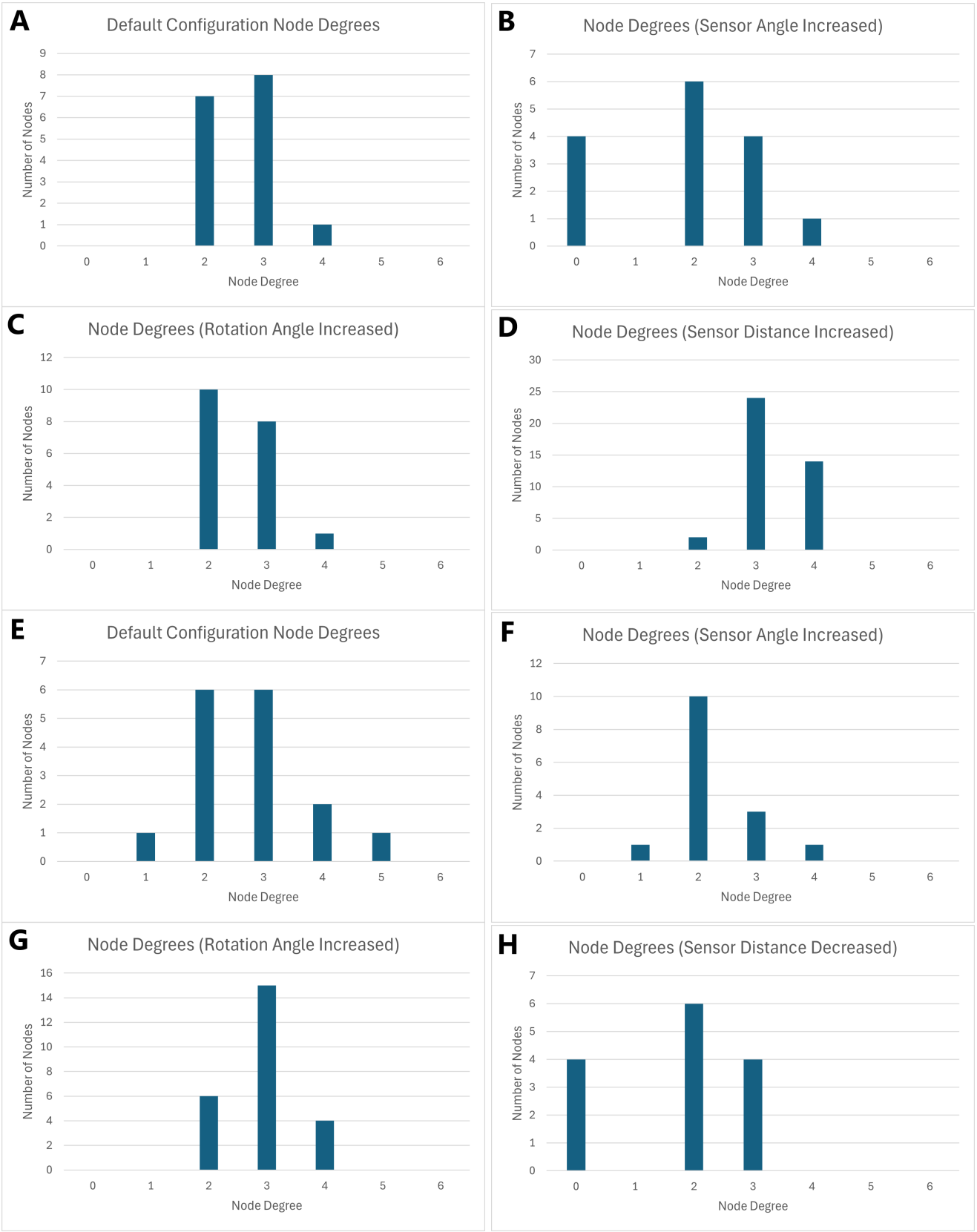


Fig. 8. Graphs depicting the node degree distribution for their respective configuration set. These distributions are the typical shapes found in our simulations, all of which provide support against the existence of a power-law distribution or preferential attachment. The letter at the top left of each graph can be used to view its respective network in figure 7.

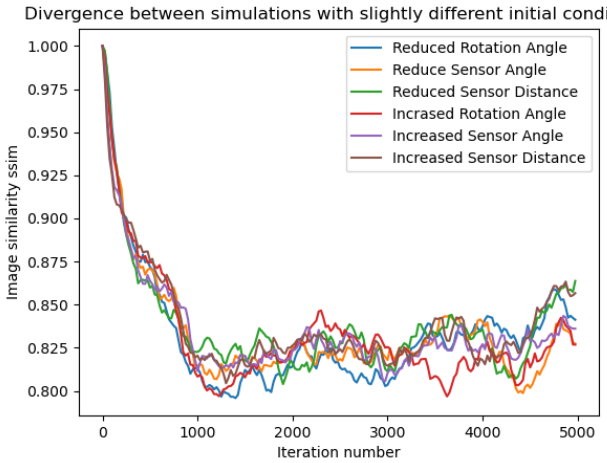


Fig. 9. This is a graph of iteration count vs the image similarity ssim metric compared to the base parameter set. Each time series shows a different edited parameter set.

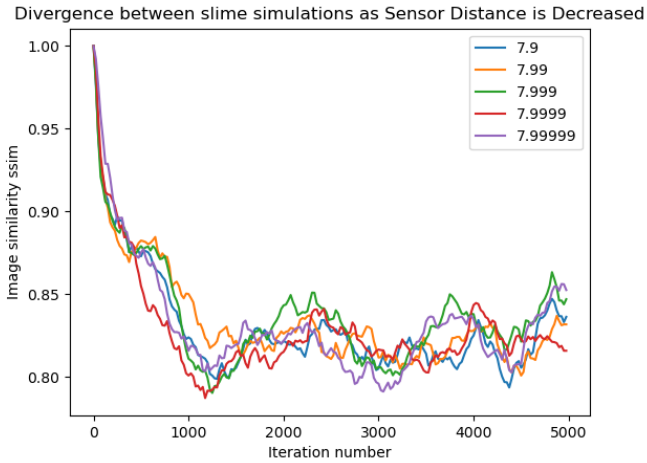


Fig. 10. This is a graph of iteration count vs the image similarity ssim metric compared to the base parameter set. Instead of varying different parameters in this graph, we only changed sensor distance.

previous experiments varied the initial conditions by a large amount.

Often, in simulations for complex adaptive systems, changing the initial conditions even slightly can lead to a great divergence in the two simulations as time moves on. This effect is clearly illustrated in Melanie Mitchell's book *Complexity A Guided Tour*. In chapter two, Mitchell introduces the logistic map, which is a model for population growth that is known to behave like a complex adaptive system. She shows that if you run two simulations of the logistic map, with the $x_0$ initial population values different by a value of $10^{-10}$, the two simulations will have completely diverged after 30 iterations [4].

To test if the slime mold simulation had the same sensitivity on initial conditions as the logistic map, we started by running seven different simulations for 5000 iterations and recording a photo every 25 iterations. Of the seven simulations, 1 was the base parameter set, while in the other six, we either increased or decreased the following parameters by $10^{-3}$: rotation angle, sensor angle, and sensor distance. For each of these 6 modified simulations, we took each picture and compared it to the corresponding picture from the base parameter set. To do this, we computed the structural similarity index between the two photos, using a function for Scikit-Image [5]. The results of this experiment can be seen in figure 9.

From figure 9, it's clear that the slime simulation has extreme sensitivity on initial conditions the same way that the logistic map does. For all the parameter sets, the image similarity between the base set rapidly drops for the first 1000 iterations. After these first 1000 iterations, the image similarity stays relatively the same for all the parameter sets, wavering between 0.850 and 0.800.

After this experiment, we were curious if the amount of the difference between the initial conditions caused faster divergence. To test this, we chose to vary only one parameter between the base set by different amounts. We chose to vary the sensor distance amount, which is 8 in the default set. We chose the values 7.9,7.99,7.999,7.9999, and 7.99999. The results for this experiment can be seen in figure 10.

By observing 10, we can see that changing the amount of difference between the initial seems to have no effect on the speed of divergence between the different simulations. If this were true, then the purple curve which represents the smallest difference in initial conditions, would diverge between the base set the slowest. However, the orange curve, which represents a larger initial difference seems to diverge slower than the purple curve. This shows that no matter how they are edited, the slime simulation has a chaotic and rapid divergence when initial conditions are changed.

## IV. Summary/Next Steps

The original Jones [1] does a fantastic job of formalizing the idea of a Physarum slime simulation in code. On top of that, they use the simulation that they build to exhibit the complex and dynamic behaviors of the slime.

In this paper, we hoped to use the Jones paper as a jumping off point. With the simulation implementation from Fogleman [3], we investigated three separate but related aspects of the slime simulation:

1) The rate at which the slime finds food
2) The different network qualities resulting from varying simulation parameters
3) Simulation sensitivity on slightly different initial conditions

We found that the slime simulation will rapidly find the food sources, and then build a very optimal network connecting all the nodes together. Even when food sources are deleted/changed throughout the simulation, the slime will dynamically respond to these changes gracefully. We also found that varying the initial parameters of the network can

lead to very different final networks, some with different node densities and average node-degree. Finally, we found that the networks are extremely sensitive to changes in initial conditions, which is in line with the way that real complex adaptive systems progress in the real world.

There are many different directions for the next steps regarding slime simulation research. Firstly, developing a methodology to quantify network properties through code rather than in observation would provide future researchers with more reliable and a higher quantity of data to work with. Next, using statistical analysis techniques could give deeper insights into the significance of observed patterns and behaviors. These might include methods like cluster analysis and regression modeling. This report only tested the slime molds on a relatively small number of food nodes. Further work into the scalability is important to know the limitations of this type of algorithm.

## V. Contribution Statement

**Tristan Larkin:**

- Implemented one of the containment versions.
- Tested various sizes of food and timing of the evolution.
- Wrote Methodology, section II.
- Wrote subsections III-A and III-B.

**Jack Wickstrom:**

- Wrote the ability to use food image maps in the simulation code
- Wrote the ability to change food maps after a number of time steps in simulation code
- Wrote code/performed paper analysis on sensitivity to changes in initial conditions

**Edgar Zapata:**

- Implemented designated spawn region of slime particles.
- Implemented another containment region version.
- Tested for node degree distribution of differing node layouts.
- Wrote Introduction and subsection III-C.

*Chat GPT Uses:*

- Converting citations into bibtex format.
- Wrote the Python scripts for generating gifs and log-scaled images.
- Wrote Python code to compare images.
- Wrote basis for spawn/containment region methods.s

## VI. Acknowledgments

We thank the UNM Center for Advanced Research Computing for the use of the Hopper machine. The paper by Jones [1] was the basis for this exploration. The code written by Michael Fogleman [2] was extremely helpful in allowing us to finish this report on time.

## References

[1] J. Jones. Characteristics of pattern formation and evolution in approximations of physarum transport networks. *Artificial Life*, 16(2):127–153, 2010.

[2] Michael Fogleman. Physarum transport networks. https://github.com/fogleman/physarum, 2020.

[3] Edgar Zapata Jack Wickstrom Michael Fogleman, Tristan Larkin. Physarum transport networks. https://github.com/trlarkin/physarum, 2020.

[4] Melanie Mitchell. *Complexity: A Guided Tour*. Oxford University Press, Inc., USA, 2009.

[5] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.