Habib University

CS/SDP 262 Computational Social Science

Fall 2020

Agent Based Modelling of Slime Mold

Submitted to: Dr. Shah Jamal Alam

Authors: Ali Hamza; Bahzad Badvi; Muhammad Usaid

December 9, 2020

Contents

1 Introduction

1.1 Abstract

Anthropocentrism and the capitalistic commodification of resources has led humans down a spiral of self-destruction, and equally increased a general disregard for nature. For most nature is an external entity, unbeknown to them that humanity is not exclusive from it. Following from this stems the negligence of intelligence shown in nature however, as we shall be exploring in our project, signs of intelligence can be found everywhere in nature. Our project deals with modelling the semi-intelligent behavior shown by a cellular organism known as Physarum Polycephalum or more commonly as Yellow slime mold. This organism has baffled biologist and researcher over its exhibition of intelligence besides the fact that it does not have a central nervous system. Surely without a central processing unit how can an organism function let alone solve difficult problems? Intriguingly enough P.Polycephalum cannot only perform basic life function but is able to solve many complex problems such as path optimization. Research suggest that a pulsating flow of biochemicals within its tube-like structure handles P.Polycephalum's intricate mobility and that the control of this fluid is the cause of its coordinated growth. Another research suggests that P.Polycephalum makes use of an external 'spatial' memory by using pheromones to mark visited regions.

Github: https://github.com/hurryingauto3/CS262-Computational-Social-Science

1.2 Approach

To try and obtain a holistic understanding and to model the behavior of P. polycephalum with a reasonable degree of accuracy, we will be using two different agent based models - each with a separate goal. The first ABM attempts to model the behavior of the slime mold as it coalesces around food sources in a maze. The maze is generated randomly and slime mold nuclei are randomly distributed across the maze and then are linked. The ABM intends to model the primitive behavior that the mold exhibits - it moves closer and closer to the food source, while the nuclei away from the food source tend to die off. However in doing so, it effectively solves the maze.

The second agent based model we implemented attempts to explore the problem solving behavior of the P. polycephalum. Our motivation behind this was found in Rules for Biologically Inspired Adaptive Network Design. Slime mold replicated the Tokyo rail network - solving a complex compuational problem with seemingly very little intelligence. Therefore, we wanted to explore how this behavior comes about.

2 Technical Documentation

2.1 Agent Based Model 1

1. **Name:** `Maze Solving Simulation via Replication of Slime Mold Behavior`
2. **Purpose**

   Using an agent-based model allows us to explore how the agents (nuclei) react with the surroundings and how they can adapt to randomized mazes and find an optimal solution. We cannot model slime mold using a simple algorithmic approach.
3. **Agents**
   (a) Turtles - represent the P. Polycephalum nuclei
      i. `distfood` - The Euclidean distance from the nucleus to the food source stored as an attribute for Agents.

   (b) Links - represent the tubular connectivity between the nuclei
      i. `midptx`, `midptx1`, `midptx2` - The x-coordinates of three midpoints of a link stored as an attribute for links.
      ii. `midpty`, `midpty1`, `midpty2` - The y-coordinates of three midpoints of a link stored as an attribute for links.

   (c) Patches - represent paths and walls in the maze
      i. `pcolor`: Black patches are paths; White patches are walls.

   (d) **Time**

   In reality Slime mold grows at about a centimeter an hour so it's ideal to have the time scale of one tick being equal to an hour. Anything less would be redundant. In our environment we've made it as such so that a step of 0.3 is equivalent to a step of $1cm$ in a real-world scenario.
4. **State/Global Variables**
   (a) **Global Variables**:
      i. `Left-Height` - Represents `pycor` for the start food source (patch with red color)
      ii. `Right-Height` - Represents `pycor` for the end food source (patch with red color)
      iii. `prob_random` - Represents the probability that a nucleus would move randomly, lies within the range 0 and 1 and the default value is 0.6
      iv. `prob_static` - Represents the probability that a nucleus would remain static, lies within the range 0 and 1 and the default value is 0.6
      v. `prob-reproduce` - Represents the probability that a nucleus divides creating another,lies within the range 0 and 1 and the default value is 0.15
      vi. `num-agents` - Represents the number of initial nuclei in the model, lies in range 0 to 6000 default value is 1000

vii. `sumfooddist` - Represents the sum of distances of nuclei from a food source.

5. **Process Overview and Scheduling:**

   Following actions take place every time step (hour):

   - For each agent draw a random-float between 0 and 1 to check if `prob_static` is achieved or if agent is on a food source.
     - If above mentioned condition is achieved do nothing
     - Else store a link-neighbor which is closest to any randomly chosen food source (could be both start or finish) in a local variable called `face-to`. If `face-to` is equal to nobody or a random-float in range 0 to 1 is within `prob_random` face any randomly chosen link-neighbor else face the agent stored in `face-to`. Move one centimeter (that is equivalent to `fd 0.3` in our case) if the topology of the maze allows for it.
   - Then call the following procedures:
     - `foodupdate`
     - `make-edges`
     - `reproduction`

6. **Sensing**

   The agent's movement is such that in each step they either move randomly, remain static, or they move towards a neighbor which is closer to the food node. The movement of an agent towards their neighbor is motivated from the fact that the nuclei prioritizes movement towards the direction it's obtaining the greatest nutrition hence an assumption is made that the flow of nutrients from the neighbor closest to the food source would be greater.

7. **Interaction**
   Each agent is connected with another agent via a link these links are thence contribute to the movement the agents make throughout the simulation as discussed.

8. **Stochasticity/Randomness**

   Randomness occurs in the following ways:

   (a) The maze generation is done randomly and has been taken from:
   http://ccl.northwestern.edu/netlogo/models/community/BAM20Maze20Generator2)

   (b) Locations of agents which are generated to represent the nuclei are set to a random coordinate

   (c) Normally (that is when `prob_random` and `prob_static` is not achieved) when an agent chooses which connected neighbor to face and move towards, the selection is done randomly from two choices:

      i. A link-neighbor closest to the starting food node

      ii. A link-neighbor closest to the ending food node

(d) A random-float is usually generated to check if probabilities `prob_random`, `prob_static` are achieved

(e) Once `prob_random` is achieved the agent may face in any random direction and take a step

(f) A random-float is used to compare the value of the number with the variable `prob-reproduce` to determine whether a nucleus would divide, therefore the reproduction action relies on randomness.

(g) If the topology doesn't allow for a newly generated agent to be placed nearby parent agent, the newly generated agent is placed at a randomly selected co-ordinate in the environment.

9. **Setup and Initialization**

   (a) **Environment**

   The environment is a geo-spacial environment where we can setup mazes/walls to show how P.Polycephalum is able to overcome such scenarios.

   (b) **Turtles/ Links/ Patches**

      i. `setup-maze, setup-start-finish, make-maze` is used to create the maze at the start. (Details of these processes can be referred from BAM's maze generation model inspired from `http://ccl.northwestern.edu/netlogo/models/community/BAM20Maze20Generator2)`

      ii. Create `num-agents` number of nuclei and then place them randomly around the world.

      iii. Delete all nuclei that are created on a wall (white patch).

      iv. Update distance of the nuclei from the nearest food source.

      v. Create links between nuclei.

10. **Input Data**

    This ABM does not take any input.

11. **Outputs**

    Plots display the number of turtles and the average distance from food source (`sumfooddist / count turtles`).

12. **Sub models and Processes**

    (a) `setup-maze, setup-start-finish, make-maze` (Details of these processes can be referred from BAM's maze generation model inspired from `http://ccl.northwestern.edu/netlogo/models/community/BAM20Maze20Generator2)`

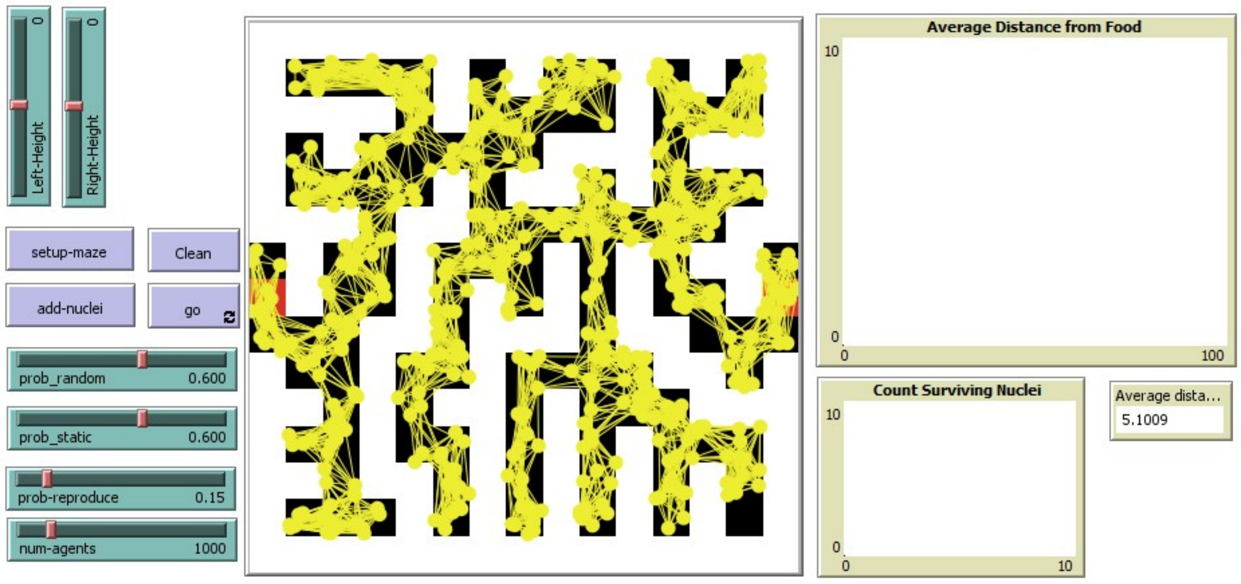
    (b) `kill` - This process is used to kill the maze-making agents once the maze is constructed

    (c) `add-nuclei` - This process adds `num-agents` number of agents in the environment at random co-ordinates and then removes those which are generate on a wall. The agents created by this process are the abstraction of P.Polycephalum's nuclei. This procedure then calls `foodupdate` and `make-edges`

(d) `foodupdate` - This process updates the distfood for each agent by storing the euclidean distance to the closest food source.

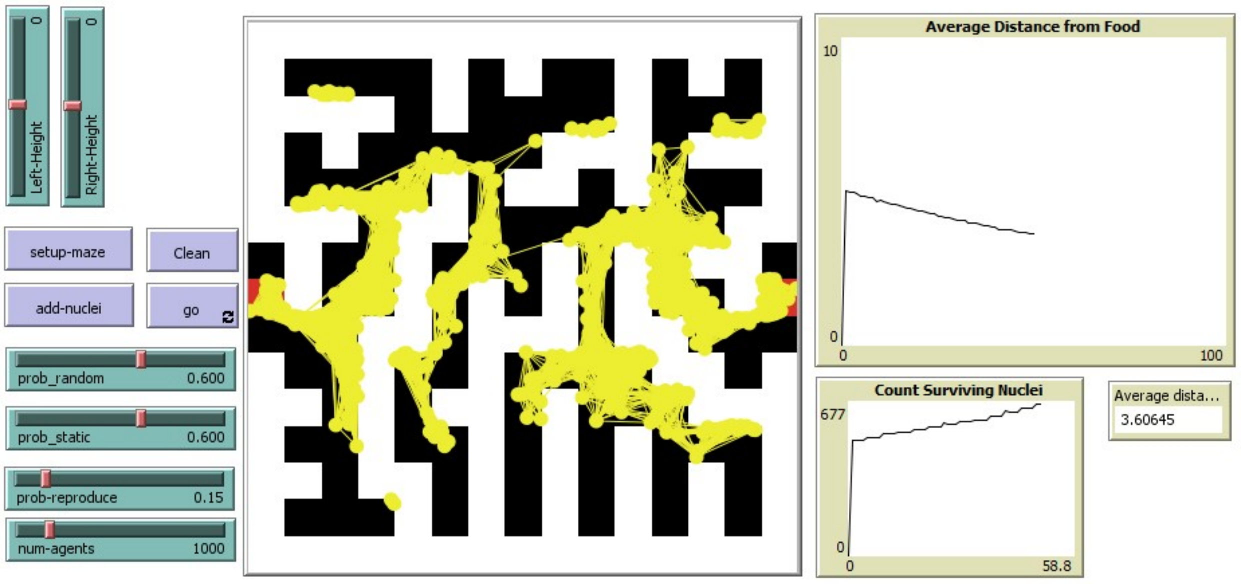
(e) `make-edges` - This process connect each agents with 15 agents which are closest to it and are not already connected. Once linking is done this process also checks if a link is crossing over a wall using `midptx`, `midptx1`, `midptx2`, `midpty`, `midpty1`, and `midpty2`, if so that link is terminated. Lastly any agent which is left with no `link-neighbor` is killed.

(f) `reproduction` - Every 4 ticks this process ask 100 randomly selected agents with their `distfood` within `sumfooddist / count turtles` to produce another nuclei if a randomly generated number falls within the prob-reproduce and locate them nearby the parent agent if topology allows else set the coordinates randomly.



Left-Height 0
Right-Height 0
prob_random 0.600
prob_static 0.600
prob-reproduce 0.15
num-agents 1000
Average dista... 5.1009

Figure 1: Start state of ABM 1



Left-Height 0
Right-Height 0
prob_random 0.600
prob_static 0.600
prob-reproduce 0.15
num-agents 1000
Average dista... 3.60645

Figure 2: End state of ABM 1

2.2 Agent Based Model 2

1. **Name:** `Minimum Spanning Graph Simulation via Replication of Slime Mold Behavior`
2. **Purpose**: Since there isn't a mathematical model that can be used to explain the behavior of slime mold, agent-based modelling can be used. Although the agents are homogeneous, they interact with the environment around them to find solutions to problems related to food sources, for example, finding an optimal path between multiple food sources.
3. **Agents**
   (a) Turtles - represent the P. Polycephalum nuclei
   (b) Links - represent the tubular connectivity between the nuclei
   (c) Patches - represent physical space of the world
      i. `trail-val`: The measure of food chemicals on a certain patch.
      ii. `is-food?`: A Boolean value that tells whether a patch is a food patch or not.
   (d) **Time** - Each tick represents an hour.
4. **State/Global Variables**
   (a) **Global Variables**
      i. `num-food-nodes`: The number of patches that are a food source. It lies within 0 and 60 and has a default value of 11.
      ii. `angle-vision`: It lies within 0 and 100 and has a default value of 30.
      iii. `num-nuclei`: The number of slime mold nuclei in the experiment, ranging from 0 to 5000 and having a default value of 900.
      iv. `food-secrete`: The amount of chemicals released by a food patch, ranging from 0 and 10, with a default value of 3.7.
      v. `step`: It represents how much a nucleus would move in a tick. It lies between 0 and 2 and has a default value of 1.0.
      vi. `evaporation-rate`: The rate at which the food chemicals evaporate on a patch, ranging from 0 to 1 and having a default value of 0.1.
      vii. `turtle-secrete`: The amount of chemicals left by a turtle (nucleus) on a patch, ranging from 0 to 10 and having a default value of 0.6.
5. **Process overview and Scheduling**
   (a) Make the turtles move.
   (b) Update the chemical trail each tick.
   (c) After the tick, delete existing edges.
   (d) Create edges again between the nuclei.

These processes are further explained in point 12 below.

6. **Sensing**
   The nuclei can pick up food chemicals that are left by food sources and by other nuclei. They can do so only within a set angle of vision determined by `angle-vision`.

7. **Interaction**
   The slime mold nuclei are connected with each other using links. The nuclei interact with patches which determines their movement, which is expounded upon below.

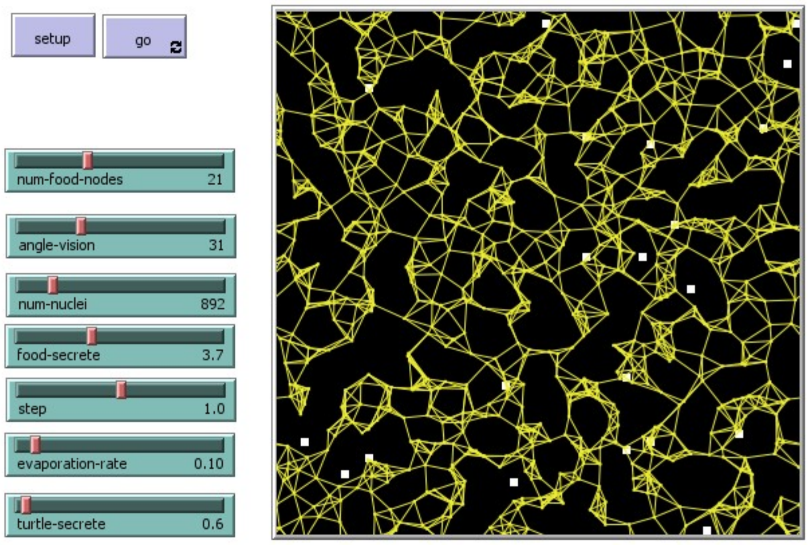
8. **Stochastic/Randomness**
   (a) The patches that are food sources are assigned randomly based on the value of `num-food-nodes`.
   (b) The positions of the nuclei - the `x` and `y` coordinates - also assigned randomly.

9. **Setup and Initialization**
   (a) **Environment**: This model uses a geo-spatial, wrapped environment that represents an abstract 2D space where the slime mold can interact with the food.
   (b) **Turtles/ Links/ Patches**
      i. Default Number of Turtles = 900
      ii. The turtles are connected with 5 other turtles in the shortest distance via links
      iii. The patches are initialized with `trail-val` = 1 and randomly sets `is-food?` = `true` for `num-food-nodes` patches

10. **Input Data**
    This ABM does not take any inputs.

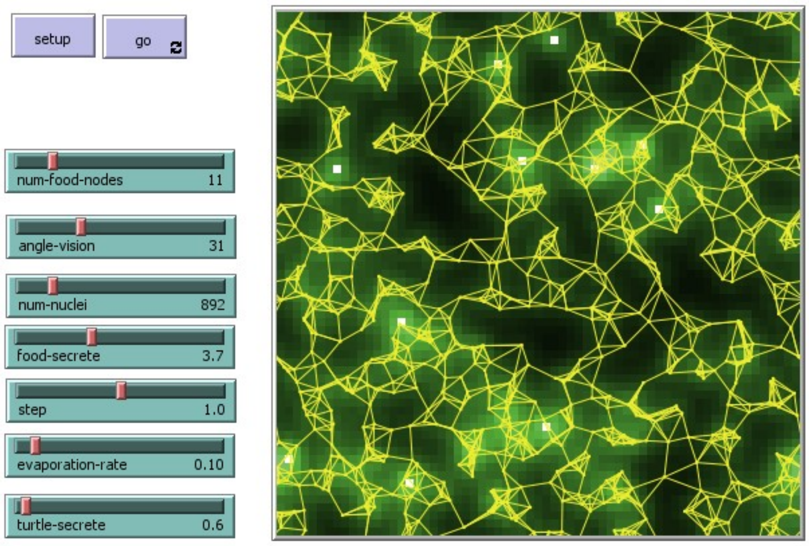
11. **Outputs**
    This ABM does not give any quantitative output. It gives a visual representation of a minimum-spanning tree formed by the nuclei and the food nodes.

12. **Sub models and Processes**
    (a) `move` - This sub process determines the movement of each turtle based on the above mentioned global variables
    (b) `update-secretion` - This sub process updates the chemical trail values on each patch and changes their colors accordingly.
    (c) `del-edges` - This sub process is called at every tick and it deletes previous links that are no longer the shortest length links.
    (d) `make-edges` - This sub process is called at every tick to create new links of minimum distance between each turtle and 5 other turtles



num-food-nodes 21
angle-vision 31
num-nuclei 892
food-secrete 3.7
step 1.0
evaporation-rate 0.10
turtle-secrete 0.6

Figure 3: Start state of ABM 1

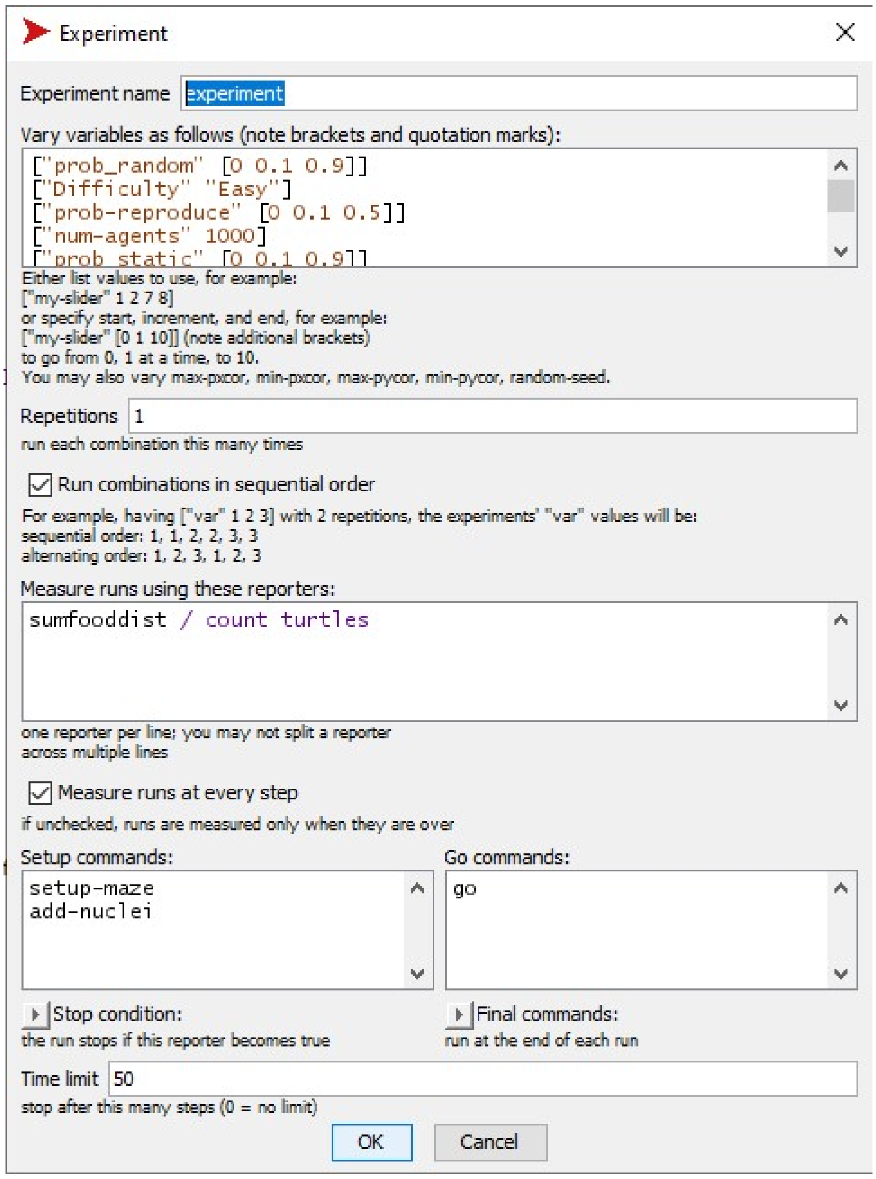


num-food-nodes 11
angle-vision 31
num-nuclei 892
food-secrete 3.7
step 1.0
evaporation-rate 0.10
turtle-secrete 0.6

Figure 4: End state of ABM 1

3 Analysis of Data

Since, the primary goal of this project was an inquiry into the inner workings of the slime mold, we chose to not limit our study to just using ABM's. Through the use of the ABM we built that is referenced in section 2.1 we were able to acquire 3 independent variables that could be used to model the slime mold, and predict its decisions. We did this by using the BehaviorSpace tool present in Netlogo 6.1. We set out a range of changes in the independent variables that we were interested in to see how these changes impact the final average euclidean distance from the two food sources. The experiment was setup as follows:



Experiment

Experiment name: experiment

Vary variables as follows (note brackets and quotation marks):

```
["prob_random" [0 0.1 0.9]]
["Difficulty" "Easy"]
["prob-reproduce" [0 0.1 0.5]]
["num-agents" 1000]
["prob_static" [0 0.1 0.9]]
```

Either list values to use, for example:
["my-slider" 1 2 7 8]
or specify start, increment, and end, for example:
["my-slider" [0 1 10]] (note additional brackets)
to go from 0, 1 at a time, to 10.
You may also vary max-pxcor, min-pxcor, max-pycor, min-pycor, random-seed.

Repetitions: 1
run each combination this many times

☑ Run combinations in sequential order

For example, having ["var" 1 2 3] with 2 repetitions, the experiments' "var" values will be:
sequential order: 1, 1, 2, 2, 3, 3
alternating order: 1, 2, 3, 1, 2, 3

Measure runs using these reporters:

```
sumfooddist / count turtles
```

one reporter per line; you may not split a reporter across multiple lines

☑ Measure runs at every step

if unchecked, runs are measured only when they are over

Setup commands:

```
setup-maze
add-nuclei
```

Go commands:

```
go
```

Stop condition:
the run stops if this reporter becomes true

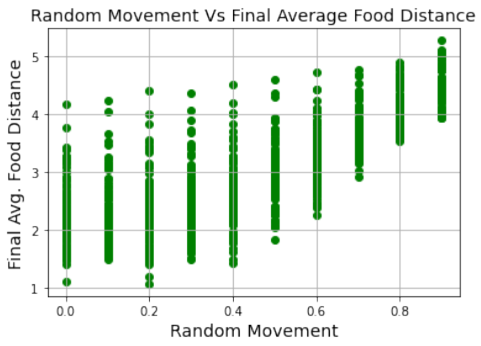
Final commands:
run at the end of each run

Time limit: 50
stop after this many steps (0 = no limit)

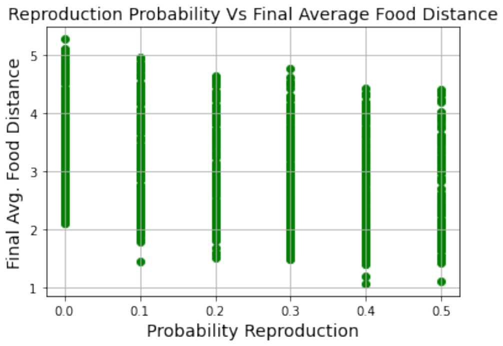
We were able the run the experiment twice and obtained 920 data points, which we then analysed using Python, and some libraries. Through this we were able to obtain visualization between the 3 independent variables and the dependent variable. However, the visualizations were very not able to tell us much about how the variables were affecting the final average food distance as the Netlogo model varied multiple quantities simultaneously. This led us to conducting a multivariate least squares regression analysis of the data. This resulted in an equation to predict the final average food distance as follows:

$$FinalAvg.FoodDist = 1.8157 + 2.4967 \cdot P(RandomMovement) - 1.7596 \cdot P(Reproduction) + 1.2086 \cdot P(Static)$$

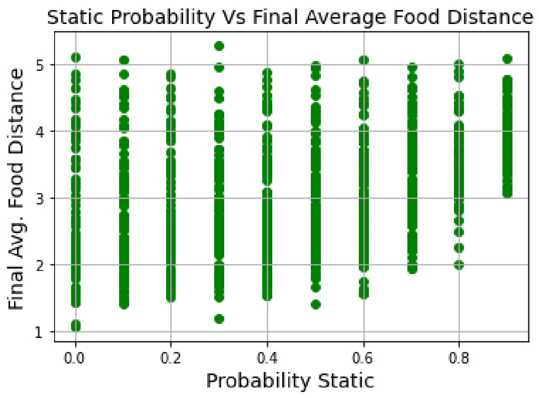
We were also able to obtain other useful statistical data from the data set that has been summarized on the next page however, we did not further investigate the data as that it was beyond the scope of the project.



(a)

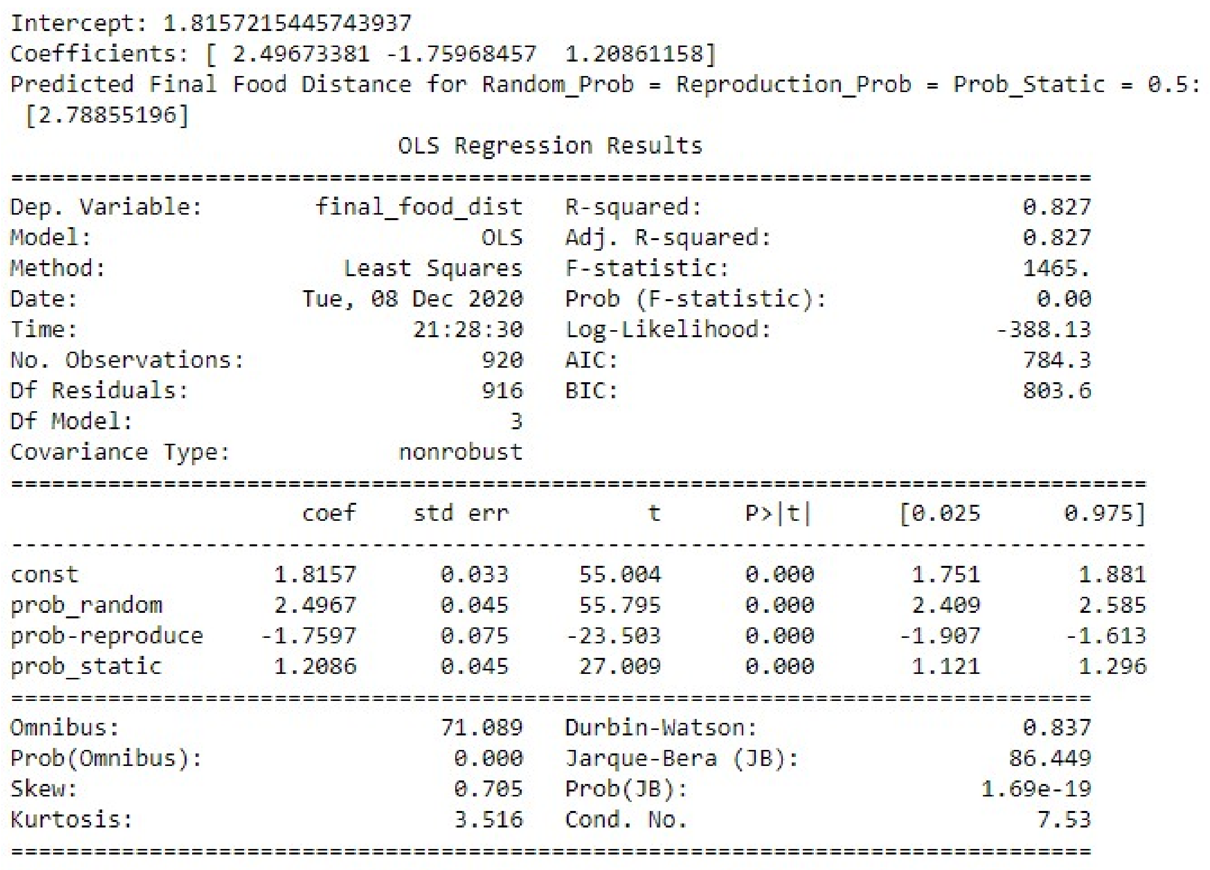


(b)



(c)

Figure 5: Scatter Plots of 3 Independent Variables against the Dependent Variable



```
Intercept: 1.8157215445743937
Coefficients: [ 2.49673381 -1.75968457  1.20861158]
Predicted Final Food Distance for Random_Prob = Reproduction_Prob = Prob_Static = 0.5:
 [2.78855196]
                            OLS Regression Results
==============================================================================
Dep. Variable:         final_food_dist   R-squared:                       0.827
Model:                             OLS   Adj. R-squared:                  0.827
Method:                  Least Squares   F-statistic:                     1465.
Date:                 Tue, 08 Dec 2020   Prob (F-statistic):               0.00
Time:                         21:28:30   Log-Likelihood:                -388.13
No. Observations:                  920   AIC:                             784.3
Df Residuals:                      916   BIC:                             803.6
Df Model:                            3
Covariance Type:             nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const              1.8157      0.033     55.004      0.000       1.751       1.881
prob_random        2.4967      0.045     55.795      0.000       2.409       2.585
prob-reproduce    -1.7597      0.075    -23.503      0.000      -1.907      -1.613
prob_static        1.2086      0.045     27.009      0.000       1.121       1.296
==============================================================================
Omnibus:                        71.089   Durbin-Watson:                   0.837
Prob(Omnibus):                   0.000   Jarque-Bera (JB):               86.449
Skew:                            0.705   Prob(JB):                     1.69e-19
Kurtosis:                        3.516   Cond. No.                         7.53
==============================================================================
```

4 Limitations of the Model

4.1 Computational Limitations

Since, our project was focused on an organism that has trillions of of nuclei, we knew computational complexity would be a problem for us from the start. We tried to explore avenues of incorporating multi-threading or perhaps use the CUDA cores of the discrete GPUs in our machines but that required the usage of another ABM modelling software called, Flame GPU. This was, obviously, outside the defined scope of the project so we were unable to rectify this issue.

4.2 Linking the Nuclei

Another major challenge was to ensure that links were being formed properly. We had to make sure that the links between two nuclei do not cross over a wall in the maze (a white patch). We simulated the slime's inability to create linkages through walls by simulating only links that were not crosses the wall. We calculated the 3 midpoints across each link and if any of those were intercepting the x and y coordinates of a white patch, we would eliminate the links. This added another layer layer of computational complexity within the model which furthered the limitations of the computational aspects of our ABM.

4.3 Preventing Clustering

Another challenge was the fact that clusters form in areas in the maze. Theoretically, these clusters should die because they get stuck and cannot access food source, and are disconnected from the rest of the P.Polycephalum. Unfortunately, eliminating these clusters would be very computationally expensive - which means that its more effective to let them be for the sake of practicality of the model, since the model itself is also computationally expensive due to the large number of nuclei and links that need to be iterated very often.

4.4 Lack of Pre-existing Research

A difficulty that we faced was the fact that there does not exist a lot of reasearch on the computational aspect of P. polycephalum. It was difficult to try and emulate a deterministic biological aspect into a probabilistic model in order to emulate the behavior of the slime mold. Availibility of more quantitative research would have helped us to think algorithmically about its behavior, but unfortunately that was difficult for us to do.

5 Conclusion

It would be unfair to say that this study in the behavior of the very titular slime mold, P.Polycephalum, via Agent Based Modelling has been conclusive. However, it is important to understand that it has provided us new insights, and questions, about its mechanism. We explored the slime mold using two Agent Based Models that tested out two different phenomena the slime mold has shown in the real world. 1) Its ability to solve mazes with food particles at the start and the end. 2) Its ability to form highly optimized paths between many food particles.

It is apparent that recreating the behavior of the slime mold is a highly difficult task due to the lack in substantial information present about its inner workings, a very important aspect that is required in algorithmic; deterministic processes. However, the use of ABM's has proven to be quite useful in allowing us to circumvent this limitation via the ability to introduce stochastic measures within the world that the potentially correct model of the slime mold can then react to. The simulations of these ABMs have provided us great insights as to how the slime mold *may* react in various conditions.

Moreover, our models were not able to completely do justice to the slime mold as it is much more optimized than our models that we produced but we were able to create data points from the model and create a least squares regression of the slime molds position with respect to 3 data points. However, we must understand that the model can incorporate many more data points to predict better positional changes in the slime mold but such a model was out of the scope of this study.

The slime mold seems to have great potential in being able to solve various graph theory problems as we have observed but to be able to replicate its processes requires much more research into its behaviors from a computational perspective at this point in time.

6 Bibliography

1. Alim, K., Amselem, G., Peaudecerf, F., Brenner, M., Pringle, A. (2013). Random network peristalsis in Physarum polycephalum organizes fluid flows across an individual. Proceedings of the National Academy of Sciences of the United States of America, 110(33), 13306-13311. Retrieved September 12, 2020, from http://www.jstor.org/stable/42712903

2. Alim, K., Andrew, N., Pringle, A., Brenner, M. (2017). Mechanism of signal propagation in Physarum polycephalum. Proceedings of the National Academy of Sciences. Retrieved September 12, 2020, from https://www.pnas.org/content/114/20/5136.short.

3. Nakagaki, T., Yamada, H. Tóth, Á. Maze-solving by an amoeboid organism. Nature 407, 470 (2000). https://doi.org/10.1038/35035159

4. Reid, C., Latty, T., Dussutour, A., Beekman, M. (2012). Slime mold uses an externalized spatial "memory" to navigate in complex environments. Proceedings of the National Academy of Sciences of the United States of America, 109(43), 17490-17494. Retrieved September 12, 2020, from http://www.jstor.org/stable/41829697

5. Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebber, D., Flicker, M., . . . Nakagaki, T. (2010). Rules for Biologically Inspired Adaptive Network Design. Science, 327(5964), new series, 439-442. Retrieved September 13, 2020, from http://www.jstor.org/stable/40508592

6. Becker, Matthias (2011). [IEEE 2011 IEEE Congress on Evolutionary Computation (CEC) - New Orleans, LA, USA (2011.06.5-2011.06.8)] 2011 IEEE Congress of Evolutionary Computation (CEC) - Design of fault tolerant networks with agent-based simulation of Physarum polycephalum. http://10.1109/CEC.2011.5949630