Abhiroop Kodandapursanjeeva

Slime Mold Simulation Tool

https://github.com/Abhiroopks/SlimeMoldSim

Motivation

This software tool was created to simulate and visualize a slime mold algorithm as presented in [1]. This algorithm is capable of solving the shortest path problem in a maze by iteratively reducing the widths of all edges that are not along the optimal path. Specifically, this is simulating the experiment performed in [2], where the slime mold organism is initially spanning every edge of the maze. Given two food sources in the maze, the slime mold can be observed to retract to the shortest path between those food sources. That is, they will retract their body mass from the edges of the maze that don't lie along the shortest path. The biological mechanism by which the slime mold organism is able to calculate this path is not fully understood, but studies such as [3] have shown that it seems to be related to chemical feedback loops that can strengthen the tubes of the organism that transport nutrients with lower energy costs. Conversely, the same chemical feedback loops weaken and shrink tubes that are deemed "high cost" in transportation of nutrients. The tool has an approximation of the same maze used in [2] as an example of functionality. The sequence of images below show an example of how the tool will visualize time steps of this algorithm.

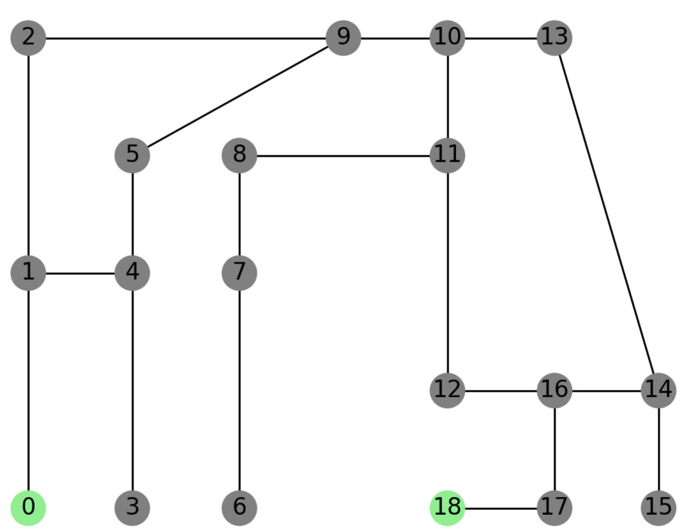

Figure 1: The initial state of the Nakagaki maze. This is an approximation of the actual maze used. Source and sink nodes are shown in green – these are analogous to food sources.

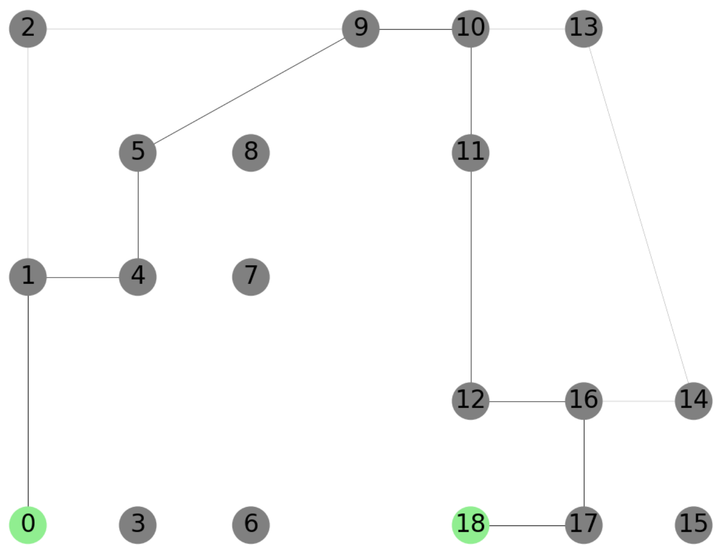

Figure 2: After 10 time steps, we can see the edges start to become faint, and more-so if they do not lie along the shortest path between food sources.

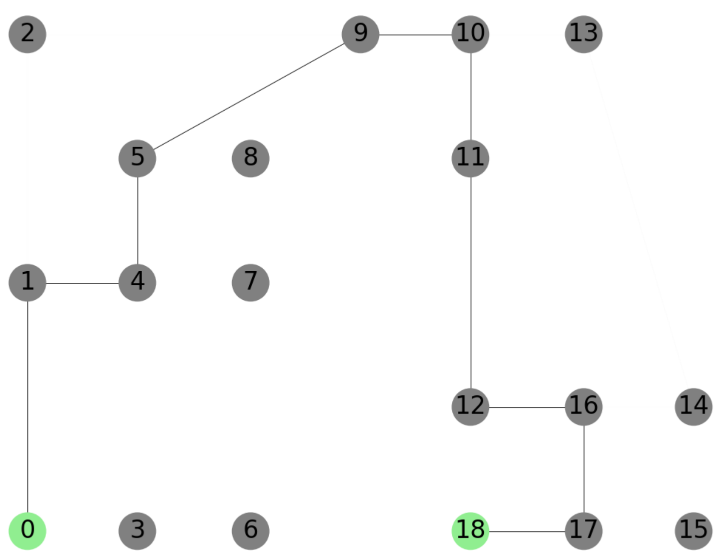

Figure 3: After 30 time steps, the system converges to the final solution. Edges that do not lie along the shortest path are not visible, and the remaining edges show the calculated shortest path between food sources.

Tool Details

The slime mold algorithm implemented is shown below, presented in [1].

Algorithm 1 Improved *Physarum Ploycephalum* algorithm

Let N be the number of nodes in the network;
Let L be the $N \times N$ matrix whose (i, j) entry L_{ij} is the length between node i and j;
Let s and e be the start and end nodes;
Let $D_{ij} \in (0, 1]$ for all $i, j = 1, 2, \ldots, N$;
Let $Q_{ij} := 0$ for all $i, j = 1, 2, \ldots, N$;
Let $p_i := 0$ for all $i = 1, 2, \ldots, N$;
Set iteration counter $t = 0$;
while *termination criteria not met* **do**
    Let $p_e := 0$ (pressure at ending node e) ;
    Calculate the pressure of every node in the network solving (5);
    Let $Q_{ij} := D_{ij}(p_i - p_j)/L_{ij}$ using (1);
    Let $D_{ij} := \frac{1}{2}\left(\frac{Q_{ij}(p_i - p_j)}{L_{ij}(p_s - p_e)} + D_{ij}\right)$ using (6);
    Let $t = t + 1$;
end

Equation (5): Fixes the net flow of fluid of all nodes to 0, except for source and sink nodes. The source node will have a net flow of +1 out, while the sink will have a net flow of -1 out (+1 entering).

$$\sum_i \frac{D_{ij}}{L_{ij}}(p_i - p_j) = \begin{cases} +1 & \text{for } j = 1, \\ -1 & \text{for } j = 2, \\ 0 & \text{otherwise.} \end{cases}$$

In the algorithm, D_{ij} corresponds to the diameter of the edge between nodes i and j, L_{ij} is the length of the edge, and Q_{ij} is the fluid flow between the nodes, and p_i is the fluid pressure at node i. Note that all values in D are initialized to 0, while those in L are initialized to 1. Actual values for L are set based on real distances between nodes of the maze.

Solving for node pressures

To solve for the pressure at every node, we use matrix operations. Using Kirchhoff's Current Law, we can create a system of equations. For each node given by index i, in a system with N total nodes the following holds:

$$\sum_{\forall j \in N} \left(\left(p_i - p_j \right) * C_{i,j} \right) = \begin{Bmatrix} 1\ if\ i : source\ node \\ -1\ if\ i : sink\ node \\ 0\ otherwise \end{Bmatrix}$$

This equation simply says the sum of all fluid flow (pressure difference * conductance) out of a node will be 0 for all, except for source and sink nodes. The sum can be expanded to the following equation:

$$EQ1: p_i \left(C_{0,1} + C_{0,2} + \cdots \ C_{0,N-1} \right) - p_1 C_{0,1} - p_2 C_{0,2} \dots p_{N-1} C_{0,N-1}$$

Let us represent the sum

$$\left(C_{0,1} + C_{0,2} + \cdots \ C_{0,N-1} \right) = \sum_{\forall j \in N} C_{0,j} = S_i$$

This is the sum of conductances between node *i* and all other nodes. Note that

$$\forall i \in N : C_{i,i} = 0$$

Then EQ1 can be represented by the inner product:

$$< \{ S_i , -C_{0,1}, -C_{0,2}, \dots , -C_{0,N-1} \} , \{ p_0, p_1, \dots , p_{N-1} \} >$$

This inner product will equal the sum of flows at node i. The first vector in this inner product can be constructed for every single node in the system in a similar fashion, leading to an NxN matrix, **A**. Then the entire system can be represented as:

$$A * p = flow$$

Where **p** is the Nx1 vector of pressures we need to solve for, and **flow** is the Nx1 vector of fluid flow sums for each node. Since we already know **flow** and **A**, we can simply solve for **p** with:

$$p = A^{-1} * flow$$

The rest of the algorithm is straightforward and allows us to update the diameters of all edges for each time step.

Future Work

Although the tool works well for paths between just two nodes, it would be interesting to extend it for multiple sources and sinks. Real slime mold is able to solve more complex problems like minimum spanning tree. Although I had some success implementing this using this algorithm, I found that the math does not converge under some circumstances, and decided to leave it out of the final version.

It would also be worthwhile to simulate slime mold's ability to actually explore a space. Based on some literature it seems slime mold mostly explores its surrounding space randomly, then decides to optimize once it finds food sources. [4] contains some interesting findings about how slime mold uses its own slime deposits as external memory, which may be of use for a simulator.

References

[1] Brabazon, Anthony & McGarraghy, Seán. (2020). Slime mould foraging: An inspiration for algorithmic design. International Journal of Innovative Computing and Applications. 11. 30. 10.1504/IJICA.2020.105316.

[2] Nakagaki, T., Yamada, H. and Toth, A. (2000) 'Maze-solving by an amoeboid organism', Nature, Vol. 407, No. 6803, pp.470.

[3] Mechanism of signal propagation in *P. polycephalum*
Karen Alim, Natalie Andrew, Anne Pringle, Michael P. Brenner Proceedings of the National Academy of Sciences May 2017, 114 (20) 5136-5141; DOI: 10.1073/pnas.1618114114

[4] Slime mold uses an externalized spatial "memory" Chris
R. Reid, Tanya Latty, Audrey Dussutour, Madeleine Beekman Proceedings of the National Academy of Sciences Oct 2012, 109 (43) 17490-17494; DOI: 10.1073/pnas.1215037109