PHYSAREALM

A Bio-inspired Stigmergic Algorithm Tool for Form-Finding

YIDONG MA¹ and WEIGUO XU²
¹,² Tsinghua University, China
¹myd15@mails.tsinghua.edu.cn ²xwg@tsinghua.edu.cn

Abstract. Physarum Polycephalum is a widespread eukaryotic microbe capable of producing effective networks between food particles to solve spatial planning problems. This paper investigates a previous algorithm for simulating Physarum Polycephalum. An open-source tool named Physarealm is developed for simulation in Rhino’s graphical algorithm editor, Grasshopper. The tool adopts a previous stigmergic multi-agent algorithm for simulation and expands its boundary into three dimensions. In addition, this tool adds some custom rules, thus giving the designer more creative control over the produced results. Two research projects have applied this tool in the design process. The first project mainly takes advantage of the tool’s path-planning ability, while the second one utilizes its aesthetic values, demonstrating the potential of the tool for further applications.

Keywords. Stigmergy; multi-agent systems; form finding; computation; biomimicry.

1. Background

Biomimicry seeks solutions to human challenges by applying strategies from nature. Researchers have shown that Physarum Polycephalum, a widespread eukaryotic microbe growing in nature, can solve several spatial planning problems. A group of Japanese researchers has shown that P. polycephalum can find the shortest route connecting two food sources when placed in a maze with two oatmeal flakes (Nakagaki et al. 2000). It is also effective at dealing with more sources. In a 2010 paper, P. pocephalum created a network similar to the existing Tokyo train system when oatmeal flakes were dispersed to represent towns on a map of the Tokyo area (Tero et al. 2010).
Inspiration from this natural phenomenon has raised questions regarding how we can mimic and emulate it to find solutions to path planning. Architects and urban designers often face spatial planning problems, such as finding effective paths between points of interest. Biomimicry of this organism’s behavior will be helpful in solving these problems. In one paper, researchers applied the mechanism of *P. pocephalum* to the spatial planning of architectural design (Meyboom and Reeves 2013). In addition, *P. pocephalum*’s network can provide aesthetic inspiration. It is necessary to introduce a tool to designers that adopts this bio-inspired form-finding method.

2. Related Works

Several approaches have been proposed by scientists to simulate the behavior of *P. pocephalum*. The first approach applied the Hagen-Poiseuille law and Kirchhoff’s circuit laws in a mathematic model (Tero et al. 2007). The second method employed the Oregonator, a model for oscillatory BZ reactions, to simulate its behavior (Adamatzky 2007). The third method utilizes a stigmergic multi-agent algorithm (Jones 2015). Among these methods, the third one produces results that are as good as the results of the others but is simpler and easier for designers to understand because it does not involve complex mathematical knowledge. These are the reasons for choosing the stigmergic algorithm in simulation.

Stigmergy is a mechanism of self-organization wherein agents coordinate through environments and collaborate to achieve a goal. The collective behavior of ants and termites are good examples of stigmergy that can be simulated using such an algorithm. In Jones’s research, several simple individual behaviors were employed to create a model reproducing the biological behavior of *P. pocephalum* (Jones 2015). However, current methods of simulation have some drawbacks that must be addressed before they can be used in applications. First, it is difficult to integrate current methods into a designer’s workflow due to the lack of interoperability. Implementation of the algorithm in the 3D modelling platform is required to simplify the exchange of data between 3D modelling software and the coding simulation environment and to reduce designers’ difficulties when using it. Sec-
ond, the current stigmergic approach to simulating *P. pocephalum* mostly works in a 2D plane. There are few studies that extend the results into 3D space. This paper tackles the above-mentioned issues by implementing the algorithm in Rhino’s graphic algorithm editor, Grasshopper.

### 3. Algorithm and Implementation

An open-source tool called *Physarealm* has been developed for simulating *Physarum Polycephalum* in Rhino’s graphic algorithm editor Grasshopper. The tool adopts a previous stigmergic algorithm and expands its boundary into three dimensions so that agents can spread inside a Brep and generate spatial networks between points. In addition, this tool adds some new rules, giving designers more creative control over the produced results. This tool has already been released on Food4Rhino, and its code repository is on Github.

#### 3.1. MULTI-AGENT MODEL

This expansion requires some modification of Jones’ original model, including the agent’s sensory behavior, motor behavior, growth and adaption behavior. In addition, we add some custom rules for creative control.

##### 3.1.1. Overview

Agents can run in two scenarios, inside a Brep or on a Nurbs surface. The model for the Nurbs surface is similar to Jones’s model in a 2D lattice and is thus omitted from discussion in this paper, except for replacing the planar Cartesian coordinates with Nurbs UV coordinates. However, the scenario for Brep is different. To expand the model into three dimensions, we utilize a 3D lattice base model for agents to move upon. Each agent represents a Physarum plasmodium particle. Another isomorphic lattice is employed to record each position at which the chemo-attractive trails diffuse over time in the environment. There are boundaries and obstacles. Lattice points inside the boundaries and outside the obstacles are capable of accommodating agents. Emitters in the model spawn agents near their locations and give them random orientations. Foods constantly emit chemo-attractive trails to attract agents through the high density of trails around them.

##### 3.1.2. Sensory Behavior

Sensory behavior has major modifications compared to Jones’ original model. The agent has multiple sensors to detect the level of chemo-attractive trail concentrations in front of it. All the sensors are on a sphere centered at the agent’s location. We can describe sensor locations using four parameters in a spherical coordinate system. The coordinate system use the agent’s direction as the z axis. Sensing Offset (SO) is the radial distance to each sensor. Sensing Angle (SA) is the maximum polar angle. Detect Directions R(NR) is the number of sensors at a particular latitude, and therefore the azimuthal angle between neighboring sensors at the same latitude $\theta = \frac{2\pi}{NR}$. Detect Directions Phy (NPhy) is the number of sensors at the same longitude, and therefore the number i sensor at the same longitude has the
polar angle of $\phi = \frac{SA \cdot i}{NPhy}$. At each step, the agent iterates over all the sensors and obtains the one with highest concentration of chemo-attractive trails; the direction toward that sensor will influence its behavior together with Rotate Angle ($RA$). The actual polar angle at which the agent moves is $\phi = \frac{RA \cdot i}{SA \cdot NPhy}$.

Figure 2. Sensory, growth and adaption behavior.

3.1.3. Growth and Adaption Behavior

As a simplification of the organism’s growth and adaption, agents will divide into two when neighborhood space availability is high and die when the neighborhood is too crowded. The algorithm computes the number of agents within a local cube of lattices centered at the current agent location and compares it to certain parameters. The radius of the cube for division detection is $DvR$, and for death it is $DeR$; therefore, the length of the first cube is $2 \cdot DvR + 1$ and the second is $2 \cdot DeR + 1$. Let $n$ be the number of neighbor agents. $DvMin(>0)$ and $DvMax( < (2 \cdot DvR + 1)^3$ for Brep environment) are the local division parameters, outside which growth cannot occur. $DeMin(>0)$ and $DeMax( < (2 \cdot DeR + 1)^3$ for Brep environment) are local diminishing parameters, outside which the agent will be removed. For example, the algorithm will count 26 neighbor cells and check if any other agents have occupied them when $DeR = 1$. If the count of neighbor-occupied cells is between $DvMin$ and $DvMax$, the agent will reproduce itself (figure 2b).

3.1.4. Chemo-attractive Trails Behavior

At each step, chemo-attractive trails in the 3D lattice are diffused by a simple mean filter kernel (typically 3*3*3). Next, the diffused trails are damped by a value $Dd$ (typically 0.1). In addition to the diffusion, food and agents deploy trails into the environment, but they are not independent. A single parameter $TrRat$ is used to
describe the amount of trail deployed by the agent to food after every scheduler step. The difference in both damping value and trail deploying rate affects the growth pattern.

3.1.5. Other Behavior

In addition to standard behaviors proposed by Jones, this model adds some behaviors for more creative control over the overall form. The first parameter is *Vertical Guide Factor (VGF)*, which affects the agent’s movement. With a high VGF, agents tend to move more vertically, which is helpful for finding forms such as columns. The second is *Initial Velocity (IV)*. We can find this behavior in many multi-agent systems such as flocking, but we can hardly assume that Physarum can do the same. However, this behavior is important for initializing directional agents. The third is *Escape Possibility (EscP)*. A high EscP allows agents to escape the boundaries or to enter obstacles, which enables more stochastic boundaries.

3.2. IMPLEMENTATION

The algorithm is implemented as a plugin for Grasshopper. A typical program involves a group of inputs, a core Physarealm component and a group of outputs. Inputs include environments, foods, emitters and settings. Each type of input has a corresponding category of components on Grasshopper’s component palettes (figure 3b). Output batteries are under the analysis category on the component palettes. Users can assign points or curves as foods and emitters, and Breps as containers and obstacles in the workspace. The program runs with a default set of parameters, but users can customize the simulation process by plugging in setting components into the core Physarealm component. A Boolean toggle and a timer control the start and loop of this program (figure 3a).

Figure 3. Usage and interface in Grasshopper.
Development of the plugin is based on Grasshopper’s SDK. Implementation of components and data-types is inherited from Grasshopper’s base classes, which separate the user interface from the back-end logic of the plugin. The code for logic is mainly implemented in the agent class (Amoeba) and population class (Physarum). Behaviors of the environment are implemented in each EnvironmentType class. The core algorithm is a modified version of a typical particle system algorithm, except that behaviors between agents and environments are different. A pseudocode is presented below:

```
InitializeAgentsAndEnvironment
Loop
  updatePopulation
  updateEnvironment
  If (someCriteria)
    removeAgentTest
    divideAgentTest
  Endif
Endloop
```

Because this plug-in is open source with a code repository released on https://github.com/maajor/Physarealm, this paper will not go into the details of the implementation.

3.3. EXPERIMENT

3.3.1. Approximation of Steiner Tree

A Steiner tree problem is a class of problems in graphs. It requires a tree of optimal interconnections for a set of given objects. This is an abstraction for designers seeking the shortest interconnecting route between POIs (Points of Interest). Most versions of the Steiner tree problems are NP-hard, while there are some algorithms to approximate them. The Physarum model in a 2D scenario gives an approximation as illustrated by Jones, and it also works well for the 3D scenario implemented in Physarealm.

The following experiments illustrate how Physarealm tries to solve the Steiner tree problems of four points. However, the four points are placed in 3D space rather than in a plan. The program loops for 2000 iterations before it reaches stability (figure 4a). The final shape between four points is an approximation of the Steiner tree in both left and front views (figure 4b).

3.3.2. Form Possibility

Apart from their interest in efficiency, as in the case of Steiner tree problems, designers also care about the potentials of the forms generated. We experimented with changing parameters hoping to exhaust the possibilities of forms. Several different forms can be generated (figure 5).
The plugin provides some components to visualize the simulated results. Functions include the density of the chemo-attractive trails (figure 6-1), interconnections of lattices based on chemo-attractive trail levels (figure 6-2), chemo-attractive trail levels at different heights (figure 6-3), interconnections of agents based on distance (figure 6-4), traces of agents (figure 6-5), locations of agents (figure 6-6) and velocity of agents (figure 6-7).
Figure 6. Visualization components.

Figure 7. Applications in projects.
4. Applications

Two research projects have applied this tool in the design process, demonstrating its potential for further applications.

4.1. THE BLOB

The first project aims to build a new Olympic stadium that congregates several sports facilities (figure 7a). The strategy is to create a park pathway that allows people to reach all the venues easily. The formation of the pathway is generated using this tool. After setting the boundary and all venues as food in the workspace, the designer runs the simulation and obtains a network prototype for the pathway model. The network, just like the nutrient-ferrying tubes created by *P. polycephalum*, is strong, resilient and effective. The designers also considered expanding the project as a new type of fabric throughout the city. This will first occur in the perimeter and then spawn further, connecting and renovating empty areas with the pathway. Finally, the pathway acts as a net to knit the city fabric together.

4.2. FUSED SYNERGY

The second project fabricates a column-like prototype of approximately 30 cm x 30 cm x 100 cm whose shape is created in this tool; within it, agents move vertically, leaving their traces behind as curves to be fabricated (figure 7b). Those curves are further simplified and optimized for fabrication. Through using stigmergic algorithms, we created an elegant weaving shape that is complex and unique. A multi-property FDM printer was designed and mounted on a robotic arm to 3D print this column prototype. The first project mainly takes advantage of the tool’s path-planning ability, while the second one utilizes its aesthetic values.

5. Discussion

The algorithm discussed in this paper is similar to other stigmergic algorithms such as Ant Colony Optimization (Dorigo et al. 2006), but there are some differences. For example, in the conventional Ant Colony Algorithm, the agent’s environment is often abstracted as a graph and path where a chemotaxic trail is deposited. However, in Physarealm, agents actually deposit and sense trails in 3D space. This tool is able to simulate the behaviors of *Physarum Polycephalum* in 3D space, with some additional features allowing creative control over its form. Moreover, it is easy to use for form-finding, as shown in some research projects. However, there are still some limitations to this tool. First, the tool does not provide methods to evaluate planning performance. Quantitative studies of this algorithm’s planning performance in 2D and 3D are lacking. Although the tool can approximate Steiner trees and create various forms, we need further experiments on which parameters influence its capability. As a result, validation of our application in projects is falling short. It is hard to measure whether the pathway finding is successful. Second, the tool functions well with Breps as its environment but may have bugs with Nurbs surfaces in Rhinoceros. These problems require further research.
Acknowledgement

I would like to thank Prof. Weiguo Xu and Weixin Huang for their advice. This tool was developed in Graduate Studio I at Tsinghua University with their guidance. I thank Yuxiang Zhang, my colleague in this studio, who helped me test the plugin and provided valuable feedback. He also created the form of Fused Synergy. I would also like to thank Davide Masserini and Violetta for their use of this tool in their design project, The Blob.

References


