Research Article
An Improved \textit{Physarum polycephalum} Algorithm for the Shortest Path Problem

Xiaoge Zhang, Qing Wang, Andrew Adamatzky, Felix T. S. Chan, Sankaran Mahadevan, and Yong Deng

1 School of Computer and Information Science, Southwest University, Chongqing 400715, China
2 Unconventional Computing Center, University of the West of England, Bristol BS16 1QY, UK
3 Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hum, Kowloon, Hong Kong
4 School of Engineering, Vanderbilt University, Nashville, TN 37235, USA

Correspondence should be addressed to Yong Deng; ydeng@swu.edu.cn

Received 8 January 2014; Revised 13 February 2014; Accepted 14 February 2014; Published 25 March 2014

Shortest path is among classical problems of computer science. The problems are solved by hundreds of algorithms, silicon computing architectures and novel substrate, unconventional, computing devices. Acellular slime mould \textit{P. polycephalum} is originally famous as a computing biological substrate due to its alleged ability to approximate shortest path from its inoculation site to a source of nutrients. Several algorithms were designed based on properties of the slime mould. Many of the \textit{Physarum}-inspired algorithms suffer from a low converge speed. To accelerate the search of a solution and reduce a number of iterations we combined an original model of \textit{Physarum}-inspired path solver with a new a parameter, called energy. We undertook a series of computational experiments on approximating shortest paths in networks with different topologies, and number of nodes varying from 15 to 2000. We found that the improved \textit{Physarum} algorithm matches well with existing \textit{Physarum}-inspired approaches yet outperforms them in number of iterations executed and a total running time. We also compare our algorithm with other existing algorithms, including the ant colony optimization algorithm and Dijkstra algorithm.

1. Introduction

Shortest path problem (SPP) is one of the fundamental problems in the field of network optimization: given a network, it is to find a path between two nodes such that the sum of the weights of its edges is minimized. Due to its wide application in many practical applications, for example transportation of food and commodities \cite{1–4}, wireless networks \cite{5, 6}, complex networks \cite{7–10}, and so forth \cite{11–16}. A number of researchers have developed many efficient algorithms to deal with this problem. For example, one of the most famous algorithms, Dijkstra algorithm \cite{17} was proposed by Edsger Dijkstra in 1959 to solve the single-source shortest path problem. Bellman-Ford algorithm \cite{18} is another well-known algorithm that computes the shortest paths starting from a single source vertex to all of the other vertices in a weighted graph. Also, label correcting algorithm has been proposed by various researchers to deal with this problem \cite{19, 20}. However, these algorithms have one common feature: they need excessive computational time when the scale of the network becomes very large. As a result, many bioinspired algorithms have emerged, such as genetic algorithm \cite{21–23}, ant colony algorithm \cite{24}, and particle swarm optimization \cite{25}.

With regard to future and emergent computing architectures, a first two-dimensional cellular automaton computing shortest path was designed in \cite{26}. The algorithm was used in developing a reaction-diffusion chemical processor to compute a collision-free shortest path in \cite{27}. Recently, an amoeboid organism, \textit{Physarum polycephalum} has been shown to be capable of solving many graph theoretical problems \cite{28–30}, including finding the shortest path [30–32], network design [33–39], population migration [40] and others [35, 41–43]. Inspired by this intelligent organism,
a path finding mathematical model has been constructed [44]. Moreover, this organism has been shown to be able to form networks with features matching those of motorway networks [38]. In addition, Baumgarten et al. have proved that the mass of mold will eventually converge to the shortest path of the network that the mold lies on [45].

However, when the original *Physarum polycephalum* model is implemented to handle shortest path problem, it needs substantial, often to a degree of excess, number of iterations. In present paper we aimed to improve the efficiency of the original *Physarum polycephalum* model. Here, a new parameter called “energy” is incorporated with the original model. We call the new method as improved *Physarum polycephalum* algorithm (IPPA). In fact, many systems have implied different kinds of “energy” when dealing with the shortest path problem. A number of them employ this kind of “energy” to solve many optimization problems [46–48]. For example, Taherian et al. [49] make full use of the Resistive Network concept to simulate the trust networks, in which every node in the trust network is mapped to a node in the resistive network, where the resistors’ values are inversely proportional to the trust values.

Fuerstman et al. [50] use the pressure-driven flow in microfluidic network to deal with the maze-like problems by searching all the possible solutions in a parallel way. Liu et al. [51, 52] found that current flows along the branch with lower impedance in circuit are fundamentally similar to the aim of path planning for a shorter path with better feasibility in the map. They employed this phenomenon to find out a short and wide path with light traffic jam for robots, which overcome a lot of shortcomings of the previous approaches. Zelek [53] treats analogous representations of harmonic functions as Markov chains and combines them with resistor networks to develop a novel method to handle the dynamic path planning problem.

Let us incorporate the parameter “energy” with the original *Physarum solver*. Advantages of our approach are manifold. It completes the *Physarum* model with rather physical notion of energy. During the actual expanding process of *Physarum*, it needs to consume energy to expand its tubes. At the same time, its tubes can absorb energy from the surrounding environment. As a result, there is a trade-off between the consumption and the absorption of the energy. By the introduction of the parameter “energy”, it makes this mode more effective when we employ it to design adaptive networks. By defining in this parameter, the executing time and iterations of the original algorithm has been decreased to a great extent. To the authors’ knowledge, this is the first attempt to combine “energy” parameter into the *Physarum* model. In addition, we have compared the efficiency of the proposed method and the original (or basic) *Physarum polycephalum* algorithm when they deal with the networks with various network topologies and nodes. Also, we have shown its advantages by comparing with other existing algorithms, including the ant colony optimization algorithm and Dijkstra algorithm.

The remainder of this paper is organized as follows. Section 2 briefly introduces the mathematical model of *Physarum polycephalum*. Section 3 presents the improved *Physarum polycephalum* algorithm for path finding. We compare the improved *Physarum polycephalum* algorithm with the basic *Physarum polycephalum* algorithm and other existing algorithms in Section 4. Section 5 concludes this paper.

## 2. *Physarum polycephalum* Inspired Shortest Path Finding Model

*Physarum polycephalum* is a large, single-celled amoeboid organism forming a dynamic tubular network connecting the discovered food sources during foraging. The mechanism of tube formation can be described as tubes become thicker in a given direction when shuttle streaming of the protoplasm persists in that direction for a certain time. It implies positive feedback between flux and tube thickness, as the conductance of the sol is greater in a thicker channel. With this mechanism, a mathematical model illustrating the shortest path finding has been constructed [44].

Suppose that the shape of the network formed by the *Physarum* is represented by a graph, in which a plasmodial tube refers to an edge of the graph, and a junction between tubes refers to a node. Two special nodes labeled as $N_1$ and $N_2$ act as the starting node and ending node, respectively. The other nodes are labeled as $N_3, N_4, N_5, N_6$, and so forth. The edge between node $N_i$ and $N_j$ is expressed as $M_{ij}$. The parameter $Q_{ij}$ denotes the flux through tube $M_{ij}$ from node $N_j$ to $N_i$. Assume the flow along the tube as an approximately Poiseuille flow; the flux $Q_{ij}$ can be expressed as

$$Q_{ij} = \frac{D_{ij}}{L_{ij}} (p_i - p_j),$$

where $p_i$ is the pressure at the node $N_i$; $D_{ij}$ is the conductivity of the tube $M_{ij}$; $L_{ij}$ is its length.

By considering that the inflow and outflow must be balanced, we have:

$$\sum_{j=1,2} Q_{ij} = 0.$$  \hspace{1cm} (2)

For the source node $N_1$ and the sink node $N_2$ the following two equations hold:

$$\sum_{i} Q_{i1} + I_0 = 0,$$

$$\sum_{j} Q_{j2} - I_0 = 0,$$  \hspace{1cm} (3)

where $I_0$ is the flux flowing from the source node, and $I_0$ is a constant value here.

In order to describe such an adaptation of tubular thickness we assume that the conductivity $D_{ij}$ changes over time according to the flux $Q_{ij}$. The following equation for the evolution of $D_{ij}(t)$ can be used:

$$\frac{d}{dt}D_{ij} = f \left( |Q_{ij}| \right) - rD_{ij},$$

where $f$ is a function representing the adaptation of tubular thickness.
where \( r \) is a decay rate of the tube. It can be obtained that the equation implies that the conductivity ends to vanish if there is no flux along the edge, while it is enhanced by the flux. The \( f \) is monotonically increasing continuous function satisfying \( f(0) = 0 \).

Then the network Poisson equation for the pressure can be obtained from (1) to (3) as follows:

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

(5)

By setting \( p_2 = 0 \) as a basic pressure level, all \( p_\text{i} \) can be determined by solving (5) and \( Q_{ij} \) can also be obtained. In this paper, \( f(Q) = |Q| \) is used. With the flux calculated, the conductivity can be derived, where (6) is used instead of (4), adopting the functional form \( f(Q) = |Q| \):

\[
\frac{D_{ij}^{n+1} - D_{ij}^n}{\delta t} = |Q| - D_{ij}^n. 
\]  

(6)

In order to illustrate the basic process of Physarum polycephalum algorithm, one simple example is shown.

**Example 1.** Consider the network shown in Figure 1; as can be seen, the numbers along each edge represent the edge length. Now, the shortest path between node 1 and node 4 needs to be found. First of all, we initialize the conductivity of all edges as 1. By implementing our proposed method, the conductivity associated with each edge is recorded, which is shown in Figure 2.

As can be seen in Figure 2, the flux of edges (1,2), (2,3), and (3,4) converge to 1 while that of the other edges converge to 0. Therefore, the shortest path found by the proposed method is \( 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \) and the results are the same as that of other algorithms, such as Dijkstra algorithm.

### 3. Improved Physarum polycephalum Algorithm

In this section, the improved Physarum polycephalum algorithm is introduced in detail. In this approach, a new parameter called “energy” is incorporated with the original mathematical model. The maintenance of the tubes in Physarum polycephalum model needs to consume energy while this energy comes from the flowing nutrients in the tubes. When the obtained energy is greater than the consumed energy, these tubes become coarser and the conductivity increases. Otherwise these tubes will vanish. The change of the tubes results in the change of the allocated flux associated with each tube. In turn, the variance of the flux changes the energy balance in the tubes. As a consequence, the tube changes further. Through a series of change associated with energy, flux, and tube, the Physarum polycephalum tends to converge to a steady state.

First of all, the energy \( E \), the flux \( Q \), and the conductivity \( D \) are defined as below

\[
E_1 = f(Q), 
\]  

(7a)

\[
E_2 = g(D), 
\]  

(7b)

\[
\Delta D = h(E_3), 
\]  

(7c)

where (7a) represents how much energy can be provided by the tube when its flux reaches \( Q \); (7b) denotes the energy consumed by the tube with conductivity equal to \( D \); (7c) means how the conductivity will change when the remaining energy (here, remaining energy is equal to the energy provided by the flux minus the energy consumed by the tube itself) is \( E_3 \). Therefore, (6) is changed to the following form:

\[
\Delta D_{ij} = h\left(f\left(Q_{ij} - g\left(D_{ij}\right)\right)\right) \times \Delta t. 
\]  

(8)

In the differential form, it will be

\[
\frac{dD_{ij}}{dt} = h\left(f\left(Q_{ij} - g\left(D_{ij}\right)\right)\right), 
\]  

(9)

which is similar to (6). But it is in more accordance with biological significance.

In what follows, we will construct the functions \( f \), \( g \), and \( h \) in (7a)-(7c). Equation (7a) reflects the relationship between the flux and the energy. In the original Physarum polycephalum model, this is defined by the absolute value of \( Q \) as shown in (6). However, it breaks the basic principle of conservation of energy. Consider a period of tube containing \( Q \)’s flux; then the energy acquired by this tube is \( E = |Q| \).
// L is an n x n matrix, L_{ij} denotes the length between node i and node j, V denotes the set of arcs.
// s is the starting node, e is the ending node
D_{ij} \leftarrow (0, 1) \ (\forall i, j = 1, 2, \ldots, N)
Q_{ij} \leftarrow 0 \ (\forall i, j = 1, 2, \ldots, N)
p_i \leftarrow 0 \ (\forall i = 1, 2, \ldots, N)
count \leftarrow 1
repeat
\quad p_e \leftarrow 0 \ // the pressure at the ending node e
\quad Calculate the pressure of every node using (5)
\quad \sum_{j \in V} \left( \frac{D_{ij}}{L_{ij}} \right) (p_i - p_j) = \begin{cases} +1 & \text{for } i = s \\ -1 & \text{for } i = e \\ 0 & \text{otherwise} \end{cases}
\quad Q_{ij} \leftarrow D_{ij} \times (p_i - p_j) / L_{ij} \ // Using (1)
\quad D_{ij} \leftarrow (1/2) \left( \left( Q_{ij} \times (p_i - p_j) \right) / (L_{ij} \times (p_s - p_e)) + D_{ij} \right) \ // Using (6)
\quad count \leftarrow count + 1
until a termination criterion is met

**Algorithm 1:** Improved *Physarum polycephalum* algorithm \((L, s, e)\).

![Figure 3: Comparison of executing time on randomly generated networks. IPPA refers to the improved *Physarum polycephalum* algorithm while BPPA represents the basic *Physarum polycephalum* algorithm.](image)

![Figure 4: Comparison of running iterations on randomly generated networks. IPPA refers to the improved *Physarum polycephalum* algorithm while BPPA represents the basic *Physarum polycephalum* algorithm.](image)

according to (6). If we regard the above tube as two connected tubes, calculate their energy, respectively; it is found that they all get the energy \(|Q|\). Therefore, the flux \(Q\) offers 2\(|Q|\)'s energy for this tube. Obviously, this fact contradicts with the previous result. It means that the relationship between the flux and the energy described by (6) in Section 2 is not reasonable.

Here, in order to satisfy the law of conservation of energy, we assume that the total energy provided by the flux beginning from the starting node to the ending node is constant and has nothing to do with the path. Therefore, the function \(f\) is defined as follows:

\[
f(Q_{ij}) = \frac{Q_{ij} \times (p_i - p_j)}{p_s - p_e}, \tag{10}\]

where \(s\) and \(e\) represent the starting node and the ending node, respectively, \(p_i\) and \(p_j\) denote the pressure at the node \(i\) and node \(j\).

Consider (7b); it reflects the consumed energy for maintaining the tubes. Naturally, the consumed energy is not only relevant with the conductivity, but also with the length of the tubes. Thus, we define it as follows:

\[
g(D_{ij}) = D_{ij} \times L_{ij} \tag{11}\]

Similarly, the effect of the energy on the change of the conductivity is also relevant with the length of the tube. The longer the tube is, the more energy it consumes. As a result, we define (7c) as

\[
h(E_3) = \frac{E_3}{L_{ij}} \tag{12}\]
After we combine (10)–(12), the following equation can be constructed:

$$\frac{dD_{ij}}{dt} = \frac{Q_{ij} \times (p_i - p_j)}{L_{ij} \times (p_s - p_e)} - D_{ij}. \quad (13)$$

Based on the above constructed model, the main procedures of this model for the shortest path problem are presented as Algorithm 1.

There are several possible solutions to decide when to stop execution of Algorithm 1, such as the maximum number of iterations is arrived, the flux through each tube remains unchanged.

As for the time complexity of this bioinspired algorithm, it is $o(n^2)$, where $n$ is the number of the nodes in the network. When we implement this algorithm, it is necessary to solve the linear equations shown in (5). Although the time complexity is substantial; different strategies can be applied to reduce the time cost, such as parallel computing and approximate approaches to solve the equations, and these advantages make this algorithm promising.

4. Comparison of Algorithms

In order to demonstrate the efficiency of the proposed method, a number of experiments have been conducted on different datasets. In addition, we have compared the computational results between our method and the existing path finding mathematical model. In addition, we have compared the computational efficiency with other state-of-the-art algorithms, including Dijkstra algorithm and Ant Colony Optimization Algorithm. All the approaches are tested on networks with random and varying topologies through computer simulations using Matlab on an Intel Pentium Dual-Core E5700 processor (3.00 GHz) with 2 GB of RAM under Windows Seven.

As for the random networks, they are generated by `erdos.renyi.game` function of the igraph package in R language [54]. Table I shows the size of testing networks. In this paper, we test the proposed method on 15 networks with different topologies and their network size ranges from 15 to 2000.
Each instance is run for 40 times, and we compute the average executing time and average accuracy.

4.1. Comparison with the Basic Physarum polycephalum Algorithm. In order to ensure that there exists at least one path from starting node to ending node in the network, we make the network fully connected. The length of each edge is uniformly distributed integer ranging from 1 to 100. The scale of the tested network varies from 15 to 2000. In our experiments, when \( \sum_{i,j=1,2,...,n}(|D_{ij}^{\text{count}+1} - D_{ij}^{\text{count}}|) \leq 0.01 \) (\( D_{ij}^{\text{count}+1} \) and \( D_{ij}^{\text{count}} \) refer to the conductivity associated with the edge \( L_{ij} \) during the \( n+1 \)th and \( n \)th iteration, resp., and \( n \) represents the scale of the network), the procedure ends.

Normally, the performance of an algorithm is its accuracy and executing time. First of all, consider the accuracy; both IPPA (short for improved Physarum polycephalum algorithm) and BPPA (short for basic Physarum polycephalum algorithm) are capable of finding the optimal path as Dijkstra algorithm with one hundred percent. Secondly, as for running time, the results are summarized in Figures 3 and 4. It is obvious that IPPA outperforms BPPA at all testing instances on both executing time and running iterations. Moreover, when the scale of the network gradually increases, the advantage of IPPA becomes more noticeable. From the view of running iterations as shown in Figure 4, due to the randomness of the edge length, for both IPPA and BPPA, executing iterations fluctuate slightly. However, as can be seen, IPPA still has obvious priority when compared with BPPA for all the testing instances. In addition, IPPA is more stable than BPPA. In summary, the above features make IPPA more applicable to real-world applications.

4.2. Comparison with Other State-of-the-Art Algorithms. Here, we compare the improved Physarum polycephalum algorithm with the classical Dijkstra algorithm [17] and Ant Colony Optimization (ACO) algorithm [55]. We focus on two important factors: a time of execution and a degree of accuracy.

In all experiments discussed here the numeric parameters, except when explicitly shown, are set to the following values: \( \alpha = 2 \), and it denotes the preference weight of pheromone trail; \( \beta = 5 \), and it represents the preference weight of the heuristic parameter \( \eta \), and \( \eta \) is the inverse of the distance between the nodes. The global update evaporation
rate $\rho$ is equal to 0.1. Both the size of the ants in each ant colony system and the maximum number of iterations are equal to the number of nodes in each network.

As can be seen in Figure 5, Dijkstra algorithm is faster than the improved Physarum polycephalum algorithm. At the same time, the improved Physarum polycephalum algorithm is faster than the ant colony optimization algorithm. It is obvious that the improved Physarum polycephalum algorithm outperforms the ant colony optimization algorithm when dealing with the shortest path problem. As for the Dijkstra algorithm, although it is faster than the improved Physarum polycephalum algorithm, it needs extra operation before it can be implemented directly to solve the shortest path problem when there is more than one shortest path in the network. For instance, for the network shown in Figure 6, the length of all the edges is 1. If we want to find the shortest path between node 1 and node 7, it will be hard for the classical Dijkstra algorithm to solve this problem. On the contrary, it is very simple for the improved Physarum polycephalum algorithm. As can be seen in Figure 7, each edge is associated with the flux. If we want to find all the shortest paths from node 1 to node 7, we can follow the direction of the flow to construct these paths, which is very easy to realize.

In order to prove this point, we have observed how Physarum will behave in this network and compare the result with that shown in Figure 7. As can be seen in Figure 8, it has shown the specific process of Physarum connecting the source node and the ending node. Each junction represents the node in the network shown in Figure 6, and Physarum is placed in the left side of the container. The Physarum polycephalum is allowed to spread along each edge in the network. With the time going, an alternative shortest path is gradually constructed as shown in Figure 8(a). At this stage, the Physarum polycephalum does not stop. On the contrary, it starts propagating back to the source as shown in Figures 8(b), 8(c), 8(d), and 8(e). At $t = 72$, the tubes of Physarum were connected with the initial food sources. Finally, the Physarum yields the network starting from the source node to the ending node shown in Figure 8(f). It can be seen that the real Physarum yields similar results as we discuss before. This is a unique feature for Physarum. It can retain all the shortest paths in a network, and its process is continuous, which is totally different from Dijkstra algorithm. This is also why the Physarum can construct the robust network.

As for the accuracy of these algorithms, we also have compared it with the ant colony optimization and Dijkstra algorithm. As shown in Figure 9, both the improved Physarum polycephalum algorithm and Dijkstra algorithm can find the optimal paths with one hundred percent. However, for the ant colony optimization algorithm, its accuracy decreases gradually with the increase of the network. When compared with the improved Physarum polycephalum algorithm and Dijkstra algorithm, its accuracy is very low.

5. Conclusion

In this paper, the basic model of Physarum polycephalum is combined with a new parameter “energy” to solve the shortest path problem. Through this novel parameter, we make the Physarum polycephalum model more reasonable. Furthermore, this parameter helps to accelerate the search speed and to reduce the number of iterations of the basic Physarum polycephalum algorithm. The performance of this novel approach is tested on various networks with different structures, and nodes ranging from 15 to 2000. In addition, we compare the proposed method with the original Physarum polycephalum model, the ant colony optimization algorithm, and Dijkstra algorithm. The results show that the proposed method outperforms the basic model of Physarum polycephalum algorithm and the ant colony optimization algorithm on both running time and executing iterations. Also, when compared with Dijkstra algorithm, it has some obvious advantages, such as finding more than one shortest path at the same time. In the future, we will investigate how to accelerate the speed of solving the linear equations shown in equation (5).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Authors’ Contribution

Xiaoge Zhang and Qing Wang these authors contributed equally to this work.

Acknowledgments

The authors greatly appreciate the reviewer’s suggestions. The work is partially supported by National Natural Science Foundation of China (Grant no. 61174022), R&D Program of China (2012BAH07B01), National High Technology Research and Development Program of China (863 Program) (Grant
References


[47] R. B. Gramacy, Shortest Paths and Network Flow Algorithms for Electrostatic Discharge Analysis [Senior thesis], University of California, Santa Cruz, Calif., USA.


Submit your manuscripts at
http://www.hindawi.com