Applied Soft Computing 23 (2014) 19–26

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Rapid Physarum Algorithm for shortest path problem

Xiaoge Zhang[a,1], Yajuan Zhang[a,1], Zili Zhang[a,b], Sankaran Mahadevan[c], Andrew Adamatzky[d], Yong Deng[a,c,e,*]

[a] School of Computer and Information Science, Southwest University, Chongqing 400715, China
[b] School of Information Technology, Deakin University, Locked Bag 20000, Geelong, VIC 3220, Australia
[c] School of Engineering, Vanderbilt University, Nashville, TN 37235, USA
[d] Unconventional Computing Center, University of the West of England, Bristol BS16 1QY, UK
[e] School of Automation, Northwestern Polytechnical University, Xi'an 710072, China

ARTICLE INFO

Article history:
Received 14 January 2013
Received in revised form 24 May 2014
Accepted 29 May 2014
Available online 13 June 2014

Keywords:
Rapid Physarum Algorithm
Physarum polycephalum
Shortest path problem
Heuristic rule

ABSTRACT

As shortest path (SP) problem has been one of the most fundamental network optimization problems for a long time, technologies for this problem are still being studied. In this paper, a new method by integrating a path finding mathematical model, inspired by *Physarum polycephalum*, with extracted one heuristic rule to solve SP problem has been proposed, which is called Rapid Physarum Algorithm (RPA). Simulation experiments have been carried out on three different network topologies with varying number of nodes. It is noted that the proposed RPA can find the optimal path as the path finding model does for most networks. What is more, experimental results show that the performance of RPA surpasses the path finding model on both iterations and solution time.

1. Introduction

The shortest path (SP) problem is one of the most fundamental problems in a wide variety of applications, such as road navigation in transportation systems [1–4], traffic routing in communication networks [5,6], and path planning in robotic systems [7,8] and so on [9–12,3]. And many classic algorithms has been developed to solve the SP problem, such as the Dijkstra labeling algorithm [13], Bellman–Ford successive approximation algorithm [14,15]. However, these traditional algorithms have a common shortcoming that they exhibit excessive computational time as the complexity of the SP problem becomes higher. Therefore, many biologically inspired algorithms have emerged, such as artificial neural networks [16], genetic algorithm [17,18], ant colony algorithm [19] and particle swarm optimization [20] and so on [21,22].

Recently, a new method for shortest path problem, inspired by *Physarum polycephalum*, is proposed. The plasmodium of *P. polycephalum* is a large amoeboid organism with a body made up of tubes and is capable of solving many graph theoretical problems, including shortest path problem [23–27,9,28–31], network design [32–35]. By extracting the underlying physiological mechanism of tube construction and degeneration, a path finding mathematical model is constructed [36]. And it is shown that the model is capable of finding the shortest route and road navigation in complex road networks [37–39]. What is more, as a photophobic organism, the plasmodium of *P. polycephalum* still exhibits the minimum-risk path while partially illuminated [40]. Furthermore, it is found that the plasmodium has the ability of forming networks with properties comparable to or better than the Tokyo rail network [32].

However, with a more detailed study into the path finding process of the model, it is revealed that the degenerating process for paths which tend to vanish requires much calculation and time. Therefore, it is necessary to reduce the complexity of calculation and accelerate the path finding process of the model. In order to solve the problem, a new method called Rapid Physarum Algorithm (RPA) is proposed in this paper, by integrating one heuristic rule into the mathematical model. In addition, the efficiency of the proposed algorithm RPA has been compared to the mathematical model on various network topologies. The analysis of the results indicates the superiority of the proposed algorithm over the basic mathematical model.

The rest of the paper is organized as follows. In Section 2, the path finding mathematical model inspired by *P. polycephalum* is briefly introduced. Section 3 describes the overall flow of the proposed RPA for solving the SP problem. The implementation and

* Corresponding author at: School of Computer and Information Science, Southwest University, Chongqing 400715, China. Tel.: +86 23 68254555; fax: +86 23 68254555.
E-mail addresses: prof.deng@hotmail.com, yongdeng@nwpu.edu.cn (Y. Deng).
[1] These authors contributed equally to this work.

http://dx.doi.org/10.1016/j.asoc.2014.05.032
1568-4946/

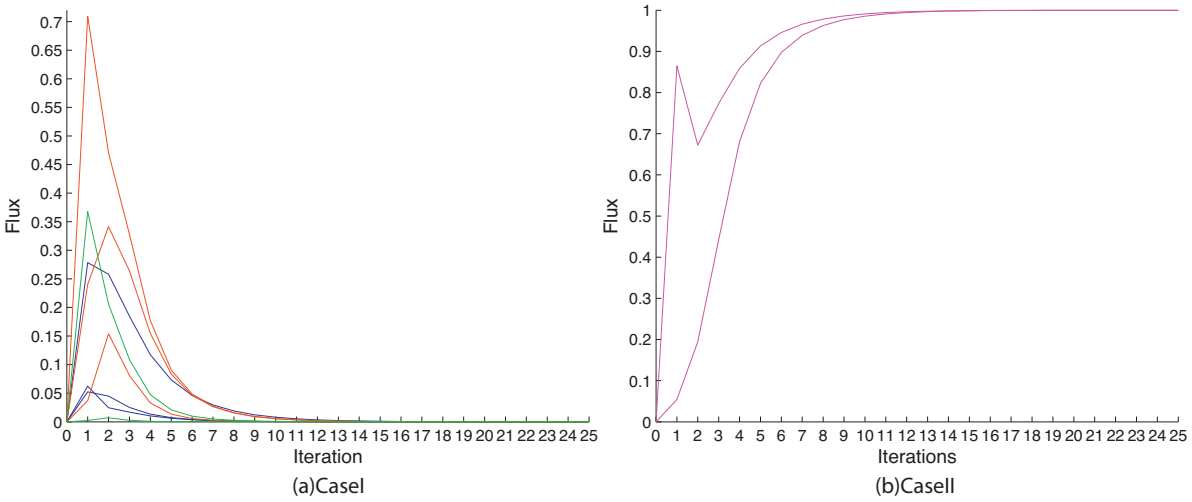

Fig. 1. Flux variation during different iterations in a fully connected graph with five nodes using the path finding mathematical model. (a) Flux variation of tubes which tend to disappear. (b) Flux variation of tubes which tend to maintain in the shortest path.

experimental results of RPA on several networks are discussed in Section 4. Finally, the conclusion is given in Section 5.

2. Mathematical model for path finding

P. polycephalum is a large, single-celled amoeboid organism that can form a dynamic tubular network linking the discovered food sources during foraging. And the physiological mechanism behind the tube formation and selection contributes to the *Physarum*'s ability of path finding, which is: tubes thicken in a given direction when the flux through it persists in that direction for a certain time. With this mechanism, a mathematical model for path finding has been constructed to simulate the adaptive dynamics of tubular network. A brief introduction for the model is given below.

Assume the shape of *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 acting as food sources are labelled as N_1 and N_2 in the graph, while other nodes are labelled as N_3, N_4, N_5, ... And the edge between node i and j is labelled as M_{ij}. The variable Q_{ij} denotes the flux through tube M_{ij} from node N_i to N_j. Assuming approximate Poiseuille flow, the flux is given as:

$$Q_{ij} = \frac{D_{ij}}{L_{ij}}(p_i - p_j) \tag{1}$$

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

By considering the inflow and outflow must be balanced, we have $\sum_i Q_{ij} = 0 (j \neq 1, 2)$. For the nodes correspond to the food sources, $\sum_i Q_{i1} + I_0 = 0$, $\sum_i Q_{i2} - I_0 = 0$, where I_0 is the flux from the source node which is set as a constant in the model.

With the equations above, the network Poisson equation for the pressure is derived:

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

By setting $p_2 = 0$ as a basic pressure level, all p_i's can be determined by solving equation (Eq. (2)), and each Q_{ij} is also obtained by Eq. (1).

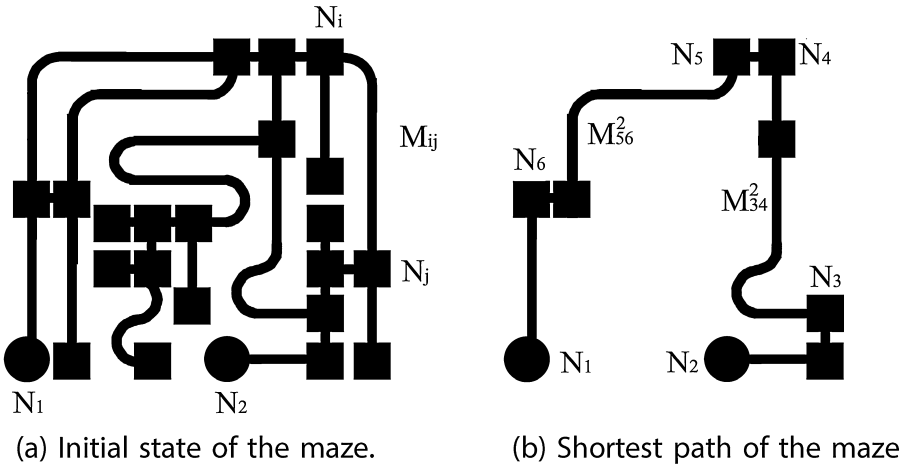

Fig. 2. Shortest path in maze found by RPA and the mathematical model. In this maze, N_1 and N_2 represented by circles denote the starting node and ending node, respectively. The other squares represent the vertices in this network. M_{ij} denote the edge connecting node N_i with N_j.

With the mechanism between the flux and tube thickness (described as the conductivity of the tube), the adaptation equation for conductivity is constructed as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}D_{ij} = f(|Q_{ij}|) - \alpha D_{ij} \tag{3}$$

It is obvious that positive feedback exists in the model, as $f(Q)$ is a increasing function with $f(0) = 0$.

3. Rapid Physarum Algorithm

With experiments of path finding process of the mathematical model described in Section 2, it is observed that although the shortest paths can be automatically selected as others disappear, much time and calculation is required to derive the final solution. However, in some SP applications, it is required that shortest paths need to be quickly identified either because an immediate response is needed or because the shortest paths need to be recalculated repeatedly. For this reason, a number of heuristic approaches have been advocated for decreasing the computation time. In this section, with the integration of one heuristic rule extracted from experiments, a Rapid Physarum Algorithm (RPA) is proposed to improve the efficiency of the original mathematical model.

3.1. Extraction of the heuristic rule

With observation of path finding process of the model in fully connected graphs, some phenomena have been found. After several iterations, the flux through each tube shows regularity in variation, which means that the flux through some tubes keeps increasing as the tubes thicken, while other keeps decreasing as the tubes thinner until the flux converge to 0 and the tubes disappear. However, when the flux through the tube decreases to a very small value. Normally, when the flux is lower than a small value θ, a lot of iterations are still needed until the tube vanishes. θ's value varies from network to network, which is determined by the properties of the network, including its structure, the edge length, etc. According to our observation, when θ is in the range [0, 0.1] in most of the networks, we can change the flux associated with corresponding edges to 0. As we can see in Fig. 1, almost 2/3 of the iterations are used for paths to adjust its flux to decrease from a small value to zero.

If some heuristic rules can be integrated to reduce the iteration process by cutting those tubes which will potentially vanish, the efficiency of the path finding process can be improved with less computation complexity. Therefore, after a detailed study into our simulation of the model and results, an empirical heuristic rule is adopted from those phenomena and integrated into the model.

Rule: If the variation of the flux through the tube maintains a stable decreasing trend, then the tube will be cut.

This heuristic rule means that if a "stable decreasing trend" is determined through variation process of the flux which may imply the flux will continue decrease until it reaches 0, the tube will not be part of shortest path and is deleted directly with its flux assigned 0. The "stable decreasing trend" criterion is met if Qv_{ij}, the times for which the flux has continuously decreased/increased, satisfies the following condition:

$$S(Qv_{ij}, Q_{ij}) = Qv_{ij} + C_1 + R_1(Q_{ij}) + R_2(n) \tag{4}$$

where positive value of Qv_{ij} means the flux has continuously increased for Qv_{ij} time while negative value means continuously decreasing times. And $R_1(Q_{ij})$ denotes the influence of current flux Q_{ij} on S. If Q_{ij} is very small, the tube is more likely to vanish and small $|Qv_{ij}|$ is needed to identify "stable decreasing trend". In contrast, if Q_{ij} is very large, more iterations are needed in case of wrong decision on cutting tubes. What is more, considering that the scale of the SP problem can also affect the decision criterion, $R_2(n)$ is included in the Eq. (4). Therefore, with Eq. (4), if $S(Qv_{ij}, Q_{ij}) \le 0$, the "stable decreasing trend" of tube between node i and j is identified and Q_{ij} can be assigned 0.

3.2. General flow of RPA

Based on the heuristic rule extracted above and the path finding mathematical model, the Rapid Physarum Algorithm is proposed. The main idea of the RPA for SP problem is presented as Algorithm 1, with the entire graph of the SP problem regarded as the tube network that *P. polycephalum* initiates at the very beginning, the starting/ending node of SP problem (N_S/N_E) as the food sources in the tube network and the paths connecting nodes as tubes.

Algorithm 1. RPA(L,S,E)

// L is an $n \times n$ matrix, L_{ij} denotes the length between node N_i and N_j
// N_S is the starting node, N_E is the ending node
$D_{ij} \leftarrow (0, 1]$ $(\forall i, j = 1, 2, \ldots, N \wedge i \neq j)$
$Q_{ij} \leftarrow 0$ $(\forall i, j = 1, 2, \ldots, N)$
$Qp_{ij} \leftarrow 0$ $(\forall i, j = 1, 2, \ldots, N)$ // flux in previous iteration
$Qv_{ij} \leftarrow 0$ $(\forall i, j = 1, 2, \ldots, N)$
$P_i \leftarrow 0$ $(\forall i = 1, 2, \ldots, N)$
$count \leftarrow 1$
repeat
  $P_E \leftarrow 0$ // pressure at the ending node N_E
  Calculate the pressure P_i of each node using Eq. (2)

$$\sum_i \frac{D_{ij}}{L_{ij}}(P_i - P_j) = \begin{cases} -1, & \text{for } j = S \\ +1, & \text{for } j = E \\ 0, & \text{otherwise} \end{cases}$$

  for $i = 1$ to $N - 1$ **do**
    for $j = i + 1$ to N **do**
      $Q_{ij} \leftarrow D_{ij} \times (P_i - P_j)/L_{ij}$ // using Eq. (1)
      Update Qv_{ij} according to Q_{ij} and Qp_{ij} using the equation below

$$Qv_{ij} = \begin{cases} Qv_{ij} + 1, & |Q_{ij}| - |Qp_{ij}| \ge 0 \text{ and } Q_{ij} \ge 0 \\ 1, & |Q_{ij}| - |Qp_{ij}| \ge 0 \text{ and } Q_{ij} < 0 \\ Qv_{ij} - 1, & |Q_{ij}| - |Qp_{ij}| < 0 \text{ and } Q_{ij} \le 0 \\ -1, & |Q_{ij}| - |Qp_{ij}| < 0 \text{ and } Q_{ij} > 0 \end{cases}$$

      if $Qv_{ij} \le f(Qv_{ij})$ **then**
        $Q_{ij} \leftarrow 0$
        $Qv_{ij} \leftarrow 0$
      end if
    end for
  end for
  $D_{ij} \leftarrow f(|Q_{ij}|) - \alpha D_{ij}$
  $count \leftarrow count + 1$
until a termination criterion is met

Firstly, the conductivity of each tube (D_{ij}) is initiated with random value between 0 and 1 and other variables are assigned with 0, such as flux through each tube (Q_{ij}), pressure at each node (P_i). Secondly, the mathematical model is used to adapt the flux and conductivity of each tube with the flux variation (Qv_{ij}) is recorded. Thirdly, according to the flux variation recorded, the heuristic rule will be applied to determine whether the tube should be cut or not. The last two steps are repeated until the termination criterion is satisfied.

There are several possible solutions to decide when to stop execution of Algorithm 1, such as the maximum number of iterations is arrived, conductivity of each tube converges to 0 or 1, and flux through each tube remains unchanged.

As a matter of fact, the time complexity of the original *Physarum* Algorithm is determined by solving the linear equations shown in Eq. (2). Its time complexity is $O(N_3)$, where $|N|$ is the number of nodes in the network. As shown in Algorithm 1, for RPA, it is still necessary to solve the linear equations. Consequently, the time complexity is the same as that of the original *Physarum* Algorithm. However, the RPA reduce the external loop iterations. In other words, RPA cost less iteration to reach the termination criterion. By this way, it accelerates the search of a solution and reduces the executing time and running iterations. In Section 4, numerical

Table 1
Comparison on road navigation in China's highways.

S	E	Existing mathematical model			Rapid Physarum Algorithm		
		Shortest path	Iteration	Time (s)	Shortest path	Iteration	Time (s)
27	20	(27,28,19,18,20)	298.0	0.502	(27,28,19,18,20)	31.0	0.057
3	17	(3,18,17)	142.7	0.148	(3,18,17)	23.0	0.047
18	21	(18,20,21)	136.0	0.220	(18,20,21)	22.0	0.100
27	31	(27,28,31)	104.3	0.318	(27,28,31)	17.0	0.082
29	19	(29,24,19)	293.3	0.895	(29,24,19)	26.0	0.032

experimental results are used to demonstrate this point. It can be noticed that in all the experiments, RPA outperforms the *Physarum* Algorithm to a great extent on executing time and iterations.

4. Implementation and experimental results

To demonstrate the effectiveness and efficiency of the proposed RPA, a series of experiments have been conducted on different datasets, in comparison with the existing path finding mathematical model.

4.1. Implementation

In order to make a fair comparison, parameters existing in both RPA and the mathematical model are set identical. The setting of parameters is as follows.

- C_1: It is set to 2, which is used to guarantee the least continuously decreasing times for the "stable decreasing trend" criterion.
- $S(Qv_{ij}, Q_{ij})$: It is set as $S(Qv_{ij}, Q_{ij}) = Qv_{ij} + C_1 + \lceil 10 \times Q_{ij} \rceil + \lceil \log_{10}(n) \rceil$, in which $R(Q_{ij})$ is $\lceil 10 \times Q_{ij} \rceil$ and $R(n)$ is $\lceil \log_{10}(n) \rceil$, as the larger current flux Q_{ij} or the scale of the SP problem (n) is, the more $|Qv_{ij}|$ is need to satisfy the "stable decreasing trend" condition.
- $f(Q)$: It is set as $f(Q) = |Q|$, with which the final solution would not depend on the initial state, that is, the shortest path always survives whether the distribution of conductivities in the initial state is random or biased [36].
- α: It is set to 1, which denotes the decay rate of the tube.
- Termination criterion: It is set as the flux through each tube remains unchanged, which practically means that the variation of flux in each tube falls in the range of $[0, 10^{-5}]$.

4.2. Experimental results

With the parameters set as described in Section 4.1, the proposed RPA is applied to three different networks, comparing with the existing mathematical model on two criteria: iterations and solution time.

4.2.1. SP in maze

In the previous study of the model, simulation of maze solving has been conducted, which shows the ability of the model to find the shortest path in a maze, as shown in Fig. 2(a). Without loss of generality, our proposed RPA is also used to solve the maze. It is shown that RPA is capable of finding the shortest path as the model does in the maze, which is depicted in Fig. 2(b). What is

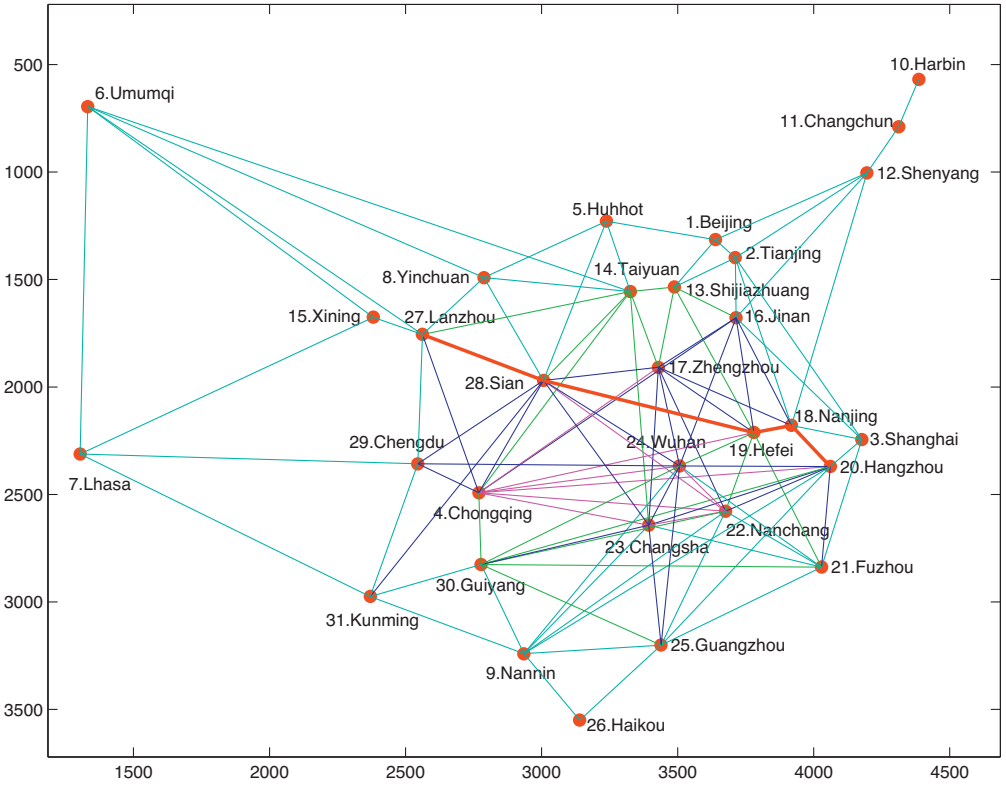

Fig. 3. Simulation results on the map of China. (For interpretation of the references to colour in the text, the reader is referred to the web version of this article.)

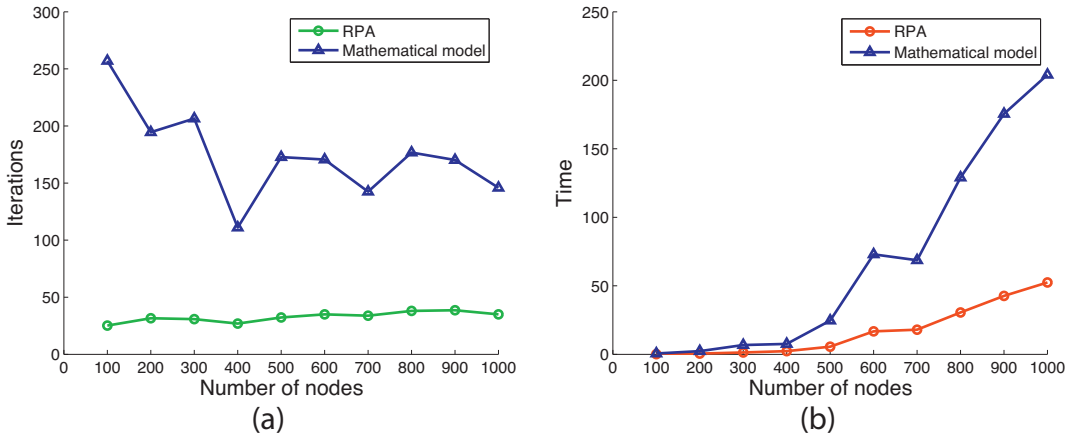

Fig. 4. Comparison on randomly fully connected networks.

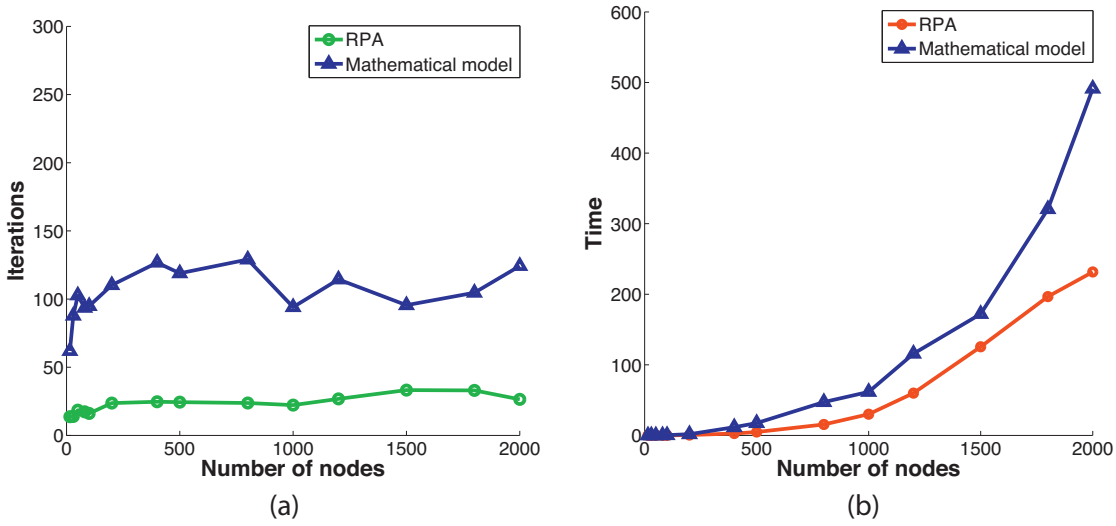

Fig. 5. Comparison on randomly distributed networks.

more, it is only 12 iterations and 0.015 s that RPA uses to derive the right solution, while the model uses 276 times and 0.25 s. For more detailed information, please refer the previous work [41,36].

4.2.2. SP in map of China

With the idea of applying our proposed algorithm to road navigation in China's highways, road network between 31 cities with 108 edges has been constructed. Let U be a set of 31 cities, for any two cities a and b from U, the cities a and b are connected by an edge (ab) if there is a shortest path starting in vicinity of a and ending in vicinity of b and not passing in vicinity of any other cities $c \in U$. The road network connecting 31 cities in China is shown in Fig. 3.[2]

Assume Lanzhou (27) as the starting node and Hanghou (20) as the ending node, the shortest path derived by RPA is illustrated as red line in the figure, which is $27 \rightarrow 28 \rightarrow 19 \rightarrow 18 \rightarrow 20$ and the length of this path is 2002. As shown in Table 1, the proposed RPA can derive the same results as the mathematical model does, which are the shortest paths of the same network with different starting/ending nodes. What is more, it is almost only 23.8 times as an average iteration that RPA costs to solve the 31 nodes' SP problem, which is nearly 13% of what the model costs.

[2] Fig. 3 is extracted from the road network as shown in maps.google.com.

4.2.3. SP in random fully connected network

Without loss of generality, RPA and the model are also tested on randomly generated networks of various sizes. The generated networks are undirected and fully connected, which guarantees that there exists at least one path from each node to every other node in the network. The length L_{ij} of each edge is uniformly distributed random integer in the range [1, 1000].

Table 2
The related parameters of the *erdos.renyi.game* function. In this function, the parameter p denotes the probability for drawing an edge between two arbitrary vertices.

Test problem	Number of nodes	Number of edges	p
1	15	23	0.20
2	30	45	0.15
3	50	107	0.10
4	80	240	0.08
5	100	304	0.06
6	200	819	0.04
7	400	1634	0.02
8	500	2496	0.02
9	800	3229	0.01
10	1000	3950	0.008
11	1200	4347	0.006
12	1500	6822	0.006
13	1800	6521	0.004
14	2000	4044	0.002

Results of the experiments are summarized in Fig. 4, comparing RPA and the model on both iterations and computation time. According to the results, although RPA sometimes cannot find the shortest path in some network topologies, resulting from wrong decisions on "stable decreasing trend", it can find a sub-optimal path instead, which is a little longer than the shortest path. However, the RPA's performance versus that of the model is most noticeable. At first, from the view of iterations (Fig. 4(a)), RPA performs stably with the iterations increase in a narrow range as the number of nodes varies from 100 to 1000. However, the performance of the model is obviously influenced by the network structure, which is unstable and lower than that of RPA. At second, as the number of nodes increases, the model exhibits excessive solution time, while the time RPA spends increases at a low growth (Fig. 4(b)). As a consequence, our proposed RPA is superior to the existing model.

4.2.4. SP in randomly distributed network

In order to verify the efficiency of RPA, we have employed R Language to generate randomly distributed networks using *erdos.renyi.game* function of the igraph package in R Language [42]. The following Table 2 displays the attributes of the generated networks. The length of each edge is randomly distributed between 0 and 100. Each instance has been run for 40 times and we compute the average executing time and average iterations.

The results are shown in Fig. 5. It can be noted that RPA outperforms *Physarum* Algorithm on executing time and iterations. From the view of running iterations as shown in Fig. 5(a), due to the randomness of the edge length and network structure, for both RPA and the mathematical model, executing iterations fluctuate slightly. However, RPA still has obvious priority when compared with the mathematical model for all the testing instances. In addition, RPA is more stable than the mathematical model. On the other hand, for the executing time, with the increase of the network size, the difference between RPA and *Physarum* Algorithm becomes bigger and bigger. It is obvious when the scale of the network gradually increases, the advantage of RPA becomes more noticeable. In summary, RPA is more efficient than the mathematical model, which makes it more applicable to real-world applications.

5. Conclusion

Technologies for navigating complex networks such as road network or the Internet by finding the shortest path are fundamental to modern society. In this paper, a Rapid Physarum Algorithm (RPA) is proposed to solve the classical shortest path (SP) problems. RPA is mainly based on the path finding mathematical model constructed in [36], which is inspired by the foraging process of the plasmodium of *P. polycephalum*. In order to avoid redundancy in computational procedures and to improve the efficiency of the model, the heuristics rule is extracted from experiments and statistics are integrated with the model in our proposed RPA. As the original model is proved mathematically rigorously that the equilibrium point corresponding to shortest path is globally asymptotically stable for the model on Riemannian surface [43], the convergence properties is also obviously exhibited by RPA.

The performance of the RPA has been compared with the model by carrying out four experiments on different networks with different topology and varying number of nodes. According to the experimental results, it is proved that RPA is capable of finding the shortest path as the existing path finding model does. What is more, all the results are highly encouraging with much superior performance exhibited by the proposed RPA.

Further studies could develop in terms of software, hardware and wetware. The software component of the algorithm could be improved to handle very large sets of data with millions of nodes and dynamically changing links betweens the nodes and properties of the links. The algorithm can be also applied to motion planning, search on real-world maps and incorporated into controllers for the autonomous mobile robots. Hardware implementation of

Table A1
Distance of highways between 31 cities on map of China (1).

City No.	1	2	3	4	5	6	7	8	9	10	11
1	0	127	Inf	Inf	484	Inf	Inf	Inf	Inf	Inf	Inf
2	127	0	1140	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
3	Inf	1140	0	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
4	Inf	Inf	Inf	0	Inf	Inf	Inf	Inf	Inf	Inf	Inf
5	484	Inf	Inf	Inf	0	Inf	Inf	718	Inf	Inf	Inf
6	Inf	Inf	Inf	Inf	Inf	0	2671	2123	Inf	Inf	Inf
7	Inf	Inf	Inf	Inf	Inf	2671	0	Inf	Inf	Inf	Inf
8	Inf	Inf	Inf	Inf	718	2123	Inf	0	Inf	Inf	Inf
9	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	0	Inf	Inf
10	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	0	274
11	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	274	0
12	684	672	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	291
13	325	318	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
14	Inf	Inf	Inf	1572	531	2691	Inf	701	Inf	Inf	Inf
15	Inf	Inf	Inf	Inf	Inf	1745	1919	Inf	Inf	Inf	Inf
16	Inf	326	887	1575	Inf	Inf	Inf	Inf	Inf	Inf	Inf
17	Inf	Inf	Inf	1207	Inf	Inf	Inf	Inf	Inf	Inf	Inf
18	Inf	911	305	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
19	Inf	Inf	Inf	1340	Inf	Inf	Inf	Inf	Inf	Inf	Inf
20	Inf	Inf	178	1610	Inf	Inf	Inf	Inf	1718	Inf	Inf
21	Inf	Inf	805	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
22	Inf	Inf	Inf	1246	Inf	Inf	Inf	Inf	1197	Inf	Inf
23	Inf	Inf	Inf	1040	Inf	Inf	Inf	Inf	894	Inf	Inf
24	Inf	Inf	Inf	898	Inf	Inf	Inf	Inf	1216	Inf	Inf
25	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	585	Inf	Inf
26	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	486	Inf	Inf
27	Inf	Inf	Inf	1140	Inf	1912	Inf	440	Inf	Inf	Inf
28	Inf	Inf	Inf	928	1003	Inf	Inf	731	Inf	Inf	Inf
29	Inf	Inf	Inf	329	Inf	Inf	2096	Inf	Inf	Inf	Inf
30	Inf	Inf	Inf	379	Inf	Inf	Inf	Inf	626	Inf	Inf
31	Inf	Inf	Inf	Inf	Inf	Inf	2231	Inf	837	Inf	Inf

Table A2
Distance of highways between 31 cities on map of China (2).

City No.	12	13	14	15	16	17	18	19	20	21
1	684	325	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
2	672	318	Inf	Inf	326	Inf	911	Inf	Inf	Inf
3	Inf	Inf	Inf	Inf	887	Inf	305	Inf	178	805
4	Inf	Inf	1572	Inf	1575	1207	Inf	1340	1610	Inf
5	Inf	Inf	531	Inf	Inf	Inf	Inf	Inf	Inf	Inf
6	Inf	Inf	2691	1745	Inf	Inf	Inf	Inf	Inf	Inf
7	Inf	Inf	Inf	1919	Inf	Inf	Inf	Inf	Inf	Inf
8	Inf	Inf	701	Inf	Inf	Inf	Inf	Inf	Inf	Inf
9	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	1718	Inf
10	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
11	291	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
12	0	Inf	Inf	Inf	978	Inf	1554	Inf	Inf	Inf
13	Inf	0	221	Inf	294	422	Inf	964	Inf	Inf
14	Inf	221	0	Inf	Inf	441	Inf	Inf	Inf	Inf
15	Inf	Inf	Inf	0	Inf	Inf	Inf	Inf	Inf	Inf
16	978	294	Inf	Inf	0	439	636	640	Inf	Inf
17	Inf	422	441	Inf	439	0	685	595	Inf	Inf
18	1554	Inf	Inf	Inf	636	685	0	183	278	Inf
19	Inf	964	Inf	Inf	640	595	183	0	Inf	998
20	Inf	Inf	Inf	Inf	Inf	Inf	278	Inf	0	689
21	Inf	Inf	Inf	Inf	Inf	Inf	Inf	998	689	0
22	Inf	Inf	Inf	Inf	Inf	884	Inf	437	591	583
23	Inf	Inf	1277	Inf	Inf	858	Inf	Inf	920	913
24	Inf	Inf	Inf	Inf	866	521	Inf	408	736	934
25	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	1356	941
26	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
27	Inf	Inf	1008	216	Inf	Inf	Inf	Inf	Inf	Inf
28	Inf	Inf	611	Inf	Inf	475	Inf	920	Inf	Inf
29	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
30	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	1672	1665
31	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf

the proposed algorithm, especially in massive-parallel processors, will allow us to increase speed of computation by several orders. Some progress have been already and basic hardware architectures, simulating behaviour of *P. polycephalum*, have been developed, see e.g. [44]. With regards to wetware, we can explore mechanisms of shortest path finding in other physical, chemical and living systems and convert these mechanisms to algorithms. Thus for example, chemotactic droplets are capable for solving mazes [45]; physical principles of path finding in networks memristors [46] are somewhat analogous to biophysical mechanisms of shortest path

Table A3
Distance of highways between 31 cities on map of China (3).

City No.	22	23	24	25	26	27	28	29	30	31
1	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
2	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
3	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
4	1246	1040	898	Inf	Inf	1140	928	329	379	Inf
5	Inf	Inf	Inf	Inf	Inf	Inf	1003	Inf	Inf	Inf
6	Inf	Inf	Inf	Inf	Inf	1912	Inf	Inf	Inf	Inf
7	Inf	Inf	Inf	Inf	Inf	Inf	Inf	2096	Inf	2231
8	Inf	Inf	Inf	Inf	Inf	440	731	Inf	Inf	Inf
9	1197	894	1216	585	486	Inf	Inf	Inf	626	837
10	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
11	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
12	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
13	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
14	Inf	1277	Inf	Inf	Inf	1008	611	Inf	Inf	Inf
15	Inf	Inf	Inf	Inf	Inf	216	Inf	Inf	Inf	Inf
16	Inf	Inf	866	Inf	Inf	Inf	Inf	Inf	Inf	Inf
17	884	858	521	Inf	Inf	Inf	475	Inf	Inf	Inf
18	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
19	437	Inf	408	Inf	Inf	Inf	920	Inf	Inf	Inf
20	591	920	736	1356	Inf	Inf	Inf	Inf	1672	Inf
21	583	913	934	941	Inf	Inf	Inf	Inf	1665	Inf
22	0	400	372	835	Inf	Inf	1152	Inf	1152	Inf
23	400	0	349	702	Inf	Inf	1136	Inf	836	Inf
24	372	349	0	1018	Inf	Inf	800	1144	1172	Inf
25	835	702	1018	0	603	Inf	Inf	Inf	1292	Inf
26	Inf	Inf	Inf	603	0	Inf	Inf	Inf	Inf	Inf
27	Inf	Inf	Inf	Inf	Inf	0	621	932	Inf	Inf
28	1152	1136	800	Inf	Inf	621	0	719	Inf	869
29	Inf	In	1144	Inf	Inf	932	719	0	Inf	869
30	1152	836	1172	1292	Inf	Inf	Inf	Inf	0	533
31	Inf	Inf	Inf	Inf	Inf	Inf	869	869	533	0

approximation by the slime mould, yet the memristors present a more robust and long-living substrate for computation. Most living creatures are skilful in finding shortest paths. For example, nematodes follow sources of chemo-attractants via shortest paths [47], and plant roots can find their way out of a maze guided by chemicals and gravity [46,48].

Acknowledgments

The authors greatly appreciate the reviews' suggestions. The work is partially supported by National Natural Science Foundation of China (Grant No. 61174022), Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20131102130002), R&D Program of China (2012BAH07B01), National High Technology Research and Development Program of China (863 Program) (Grant No. 2013AA013801), the open funding project of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (Grant No. BUAA-VR-14KF-02).

Appendix A. Distance dataset of 31 cities on map of China

See Tables A1–A3.

References

[1] L. FU, D. Sun, L.R. Rilett, Heuristic shortest path algorithms for transportation applications: state of the art, Comput. Oper. Res. 33 (2006) 3324–3343.
[2] Y.-L. Chen, H.-H. Yang, Shortest paths in traffic-light networks, Transport. Res. B: Methodol. 34 (2000) 241–253.
[3] Y. Zhang, Z. Zhang, Y. Deng, S. Mahadevan, A biologically inspired solution for fuzzy shortest path problems, Appl. Soft Comput. 13 (2013) 2356–2363.
[4] C. Gao, C. Yan, Z. Zhang, Y. Hu, S. Mahadevan, Y. Deng, An amoeboid algorithm for solving linear transportation problem, Physica A 398 (2014) 179–186.
[5] M.M. ALi, F. Kamoun, Neural networks for shortest path computation and routing in computer networks, IEEE Trans. Neural Netw. 4 (1993) 941–954.
[6] F. Yu, Y. Li, T.-J. Wu, A temporal ant colony optimization approach to the shortest path problem in dynamic scale-free networks, Physica A 389 (2010) 629–636.
[7] G. Desaulniers, F. Soumis, P. Ecole, Q. Montreal, An efficient algorithm to find a shortest path for a car-like robot, IEEE Trans. Robot. Automat. 11 (1995) 819–828.
[8] Y.Y. Cha, Navigation of a free-ranging mobile robot using heuristic local path-planning algorithm, Robot. Comput. Integr. Manuf. 13 (1997) 145–156.
[9] Y. Deng, Y. Chen, Y. Zhang, S. Mahadevan, Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment, Appl. Soft Comput. 12 (2012) 1231–1237.
[10] J. Liu, F.T.S. Chan, Y. Li, Y. Zhang, Y. Deng, A new optimal consensus method with minimum cost in fuzzy group decision, Knowl.-Based Syst. 35 (2012) 357–360.
[11] B. Kang, Y. Deng, R. Sadiq, S. Mahadevan, Evidential cognitive maps, Knowl.-Based Syst. 35 (2012) 77–86.
[12] X. Zhang, Y. Deng, F.T. Chan, P. Xu, S. Mahadevan, Y. Hu, IFSJSP: a novel methodology for the job-shop scheduling problem based on intuitionistic fuzzy sets, Int. J. Prod. Res. 51 (2013) 5100–5119.
[13] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[14] L.R. Ford, Network Flow Theory, Rand Corporation, 1956.
[15] R. Bellman, On a routing problem, Q. Appl. Math. (1958).
[16] F. Araujo, B. Ribeiro, L. Rodrigues, A neural network for shortest path computation, IEEE Trans. Neural Netw. 12 (2001) 1067–1073.
[17] C.W. Ahn, R.S. Ramakrishna, A genetic algorithm for shortest path routing problem and the sizing of populations, IEEE Trans. Evol. Comput. 6 (2002) 566–579.
[18] L. Li, P. Yang, L. Ou, Z. Zhang, P. Cheng, Genetic algorithm-based multi-objective optimisation for QoS-aware web services composition, in: Knowledge Science, Engineering and Management, Springer, Berlin, Heidelberg, 2010, pp. 549–554.
[19] K. Ghoseiri, B. Nadjari, An ant colony optimization algorithm for the bi-objective shortest path problem, Appl. Soft Comput. 10 (2010) 1237–1246.
[20] A.W. Mohemmed, N.C. Sahoo, T.K. Geok, Solving shortest path problem using particle swarm optimization, Appl. Soft Comput. 8 (2008) 1643–1653.
[21] L. Li, P. Cheng, L. Ou, Z. Zhang, Applying multi-objective evolutionary algorithms to QoS-aware web service composition, Adv. Data Mining Appl. 6441 (2010) 270–281.
[22] H. Wang, X. Lu, X. Zhang, Q. Wang, Y. Deng, A bio-inspired method for the constrained shortest path problem, Scientific World Journal 2014 (2014), http://dx.doi.org/10.1155/2014/271280, Article ID 271280.
[23] T. Nakagaki, H. Yamada, M. Hara, Smart network solutions in an amoeboid organism, Biophys. Chem. 107 (2004) 1–5.
[24] X. Zhang, Q. Wang, F.T.S. Chan, S. Mahadevan, Y. Deng, A *Physarum polycephalum* optimization algorithm for the bi-objective shortest path problem., Int. J. Unconvent. Comput. 10 (2014) 143–162.
[25] T. Nakagaki, R. Kobayashi, Y. Nishiura, T. Ueda, Obtaining multiple separate food sources: behavioural intelligence in the *Physarum plasmodium*, Proc. R. Soc. Lond. Ser. B: Biol. Sci. 271 (2004) 2305–2310.
[26] Z. Zhang, H. Zhou, Y. Wu, T. Qian, Bio-inspired dynamic composition and reconfiguration of service-oriented internetware systems, in: Proceedings of the 2nd International Conference on Swarm Intelligence, 2011, pp. 364–373.
[27] X. Zhang, Q. Wang, A. Adamatzky, F.T. Chan, S. Mahadevan, Y. Deng, A biologically inspired optimization algorithm for solving fuzzy shortest path problems with mixed fuzzy arc lengths, J. Optimiz. Theor. Appl. (2014), http://dx.doi.org/10.1007/s10957-014-0542-6.
[28] A. Adamatzky, Physarum Machines: Computers from Slime Mould, World Scientific Publishing Co. Pte. Ltd., 2010.
[29] X. Zhang, S. Huang, Y. Hu, Y. Zhang, S. Mahadevan, Y. Deng, Solving 0-1 knapsack problems based on amoeboid organism algorithm, Appl. Math. Comput. 219 (2013) 9959–9970.
[30] X. Zhang, Y. Zhang, Y. Hu, Y. Deng, S. Mahadevan, An adaptive amoeba algorithm for constrained shortest paths, Expert Syst. Appl. 40 (2013) 7607–7616.
[31] E.R. Miranda, Harnessing the intelligence of *Physarum polycephalum* for unconventional computing-aided musical composition, Int. J. Unconvent. Comput. 10 (2014) 251–268.
[32] A. Tero, S. Takagi, T. Saigusa, K. Ito, D.P. Bebber, M.D. Fricker, K. Yumiki, R. Kobayashi, T. Nakagaki, Rules for biologically inspired adaptive network design, Science 327 (2010) 439–442.
[33] A. Adamatzky, R. Alonso-Sanz, Rebuilding iberian motorways with slime mould, Biosystems 105 (2011) 89–100.
[34] A. Adamatzky, P.P. de Oliveira, Brazilian highways from slime mold's point of view, Kybernetes 40 (2011) 1373–1394.
[35] A. Adamatzky, X.-S. Yang, Y.-X. Zhao, Slime mould imitates transport networks in China, Int. J. Intell. Comput. Cybernet. 6 (2013) 232–251.
[36] A. Tero, R. Kobayashi, T. Nakagaki, A mathematical model for adaptive transport network in path finding by true slime mold, J. Theor. Biol. 244 (2007) 553–564.
[37] A. Tero, K. Yumiki, R. Kobayashi, T. Saigusa, T. Nakagaki, Flow-network adaptation in Physarum amoebae, Theor. Biosci. 127 (2008) 89–94.
[38] A. Tero, R. Kobayashi, T. Nakagaki, Physarum solver: a biologically inspired method of road-network navigation, Physica A 363 (2006) 115–119.
[39] X. Zhang, Z. Zhang, Y. Zhang, D. Wei, Y. Deng, Route selection for emergency logistics management: a bio-inspired algorithm, Safety Sci. 54 (2013) 87–91.
[40] T. Nakagaki, M. Iima, T. Ueda, Y. Nishiura, T. Saigusa, A. Tero, R. Kobayashi, K. Showalter, Minimum-risk path finding by an adaptive amoebal network, Phys. Rev. Lett. 99 (2007) 068104.
[41] Y. Zhang, Z. Zhang, Y. Deng, An improved maze solving algorithm based on an amoeboid organism, in: Proceedings of the 23rd Chinese Control and Decision Conference, 2011, pp. 1440–1443.
[42] R Language, 2013. http://igraph.sourceforge.net/doc/R/erdos.renyi.game.html
[43] T. Miyaji, I. Ohnishi, Physarum can solve the shortest path problem on Riemannian surface mathematically rigorously, Int. J. Pure Appl. Math. 47 (2008) 353–369.
[44] M.-A.I. Tsompanas, G.C. Sirakoulis, Modeling and hardware implementation of an amoeba-like cellular automaton, Bioinspir. Biomim. 7 (2012) 036013.
[45] I. Lagzi, S. Soh, P.J. Wesson, K.P. Browne, B.A. Grzybowski, Maze solving by chemotactic droplets, J. Am. Chem. Soc. 132 (2010) 1198–1199.
[46] Z. Ye, S.H.M. Wu, T. Prodromakis, Computing shortest paths in 2D and 3D memristive networks, in: Memristor Networks, Springer, 2014, pp. 537–552.
[47] A.M. Reynolds, T.K. Dutta, R.H. Curtis, S.J. Powers, H.S. Gaur, B.R. Kerry, Chemotaxis can take plant-parasitic nematodes to the source of a chemo-attractant via the shortest possible routes, J. R. Soc. Interface 8 (2011) 568–577.
[48] A. Adamatzky, Towards Plant Wires, Biosystems, 2014, http://dx.doi.org/10.1016/j.biosystems.2014.06.006.