An Electric Circuit Analogue of a Mathematical Model for Adaptive Transport Network in True Slime Mold

Yuta Kondo† and Hisa-Aki Tanaka†

†Department of Electronic Engineering, The University of Electro-Communications (UEC)
1-5-1 Choufugaoka, Choufu-shi, Tokyo 182-8585 Japan
Email: kondouyut@synchro2.ee.uec.ac.jp

Abstract—
An electric circuit analogue is obtained for a mathematical model of adaptive transport networks in true slime mold. The proposed method is easily implemented on SPICE and the shortest path is always found, as confirmed by systematic simulations and analysis. The required time for finding the shortest path is also numerically analyzed, where a systematic comparison is made between the data from the proposed method and the Dijkstra's algorithm.

1. Introduction

Recently a pioneering study by Tero et. al. [1] has proposed a new shortest path finding method derived from observation of an amoeba-like organism, the true slime mold *Physarum polycephalum*. Here, we propose a SPICE-oriented shortest path finding method, which is an electrical circuit analogue of [1]. From a systematic comparison between the proposed method and the Dijkstra's algorithm, we found an interesting feature in the proposed method; competing, multiple short paths are found simulaneously at an early stage of our path finding process.

2. Mathematical Model for Transport Network in *Physarum polycephalum* and Its Electric Circuit Analogue

Experimental observations have uncovered that adaptive transport networks in true slime mold *Physarum polycephalum*(; abbreviated as *Physarum*, below) have a shortest path finding ability [2]. Motivated by this observation, Tero et. al. [1] derived a simple mathematical model for adaptive transport networks in *Physarum*, and they numerically confirmed that the model shows a shortest path finding ability.

In this section, we review the mathematical model in [1] (2.1) and propose a circuit analogue of this model which is easily implemented on SPICE (2.2).

2.1. Mathematical Model for Transport Network in *Physarum*

We start from considering the adaptive transport network in *Physarum* as shown in Fig. 1(a). Initially, we have set

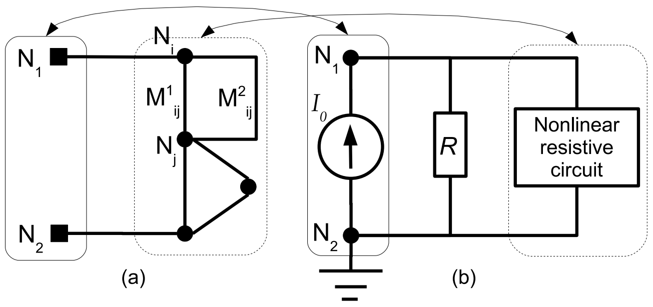

Fig. 1 Network architecture in
(a) adaptive transport networks [1]
(b) nonlinear resistive circuits in the proposed method

two 'food sources' N_1 and N_2 in the network. The network is formed with bunch of 'tubes' that are filled with certain liquid transporting nutritions and sensing informations. The tube is denoted as M_{ij} that connects the nodes N_i and N_j. If there are multiple tubes connecting the same pair of nodes, we distinguish these tubes as M^1_{ij} and M^2_{ij}. The total numbers of tubes and nodes are M and N, respectively. In [1] the dynamics of network adaptation in *Physarum* is modelled from the experimental observations [2] and the physiological data (for detail, see [1]), as follows.

First the variable Q_{ij} is introduced, which is the flux through M_{ij} from N_i to N_j. As a Poiseuille flow is assumed in the tube, the flux Q_{ij} is given by

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

where p_i is a pressure at the node N_i, L_{ij} is a length of the tube M_{ij} and D_{ij} is its conductivity. As the total amount of liquid is concerved, Kirchhoff's law holds at each node;

$$\sum_i Q_{ij} = 0, \ (j \neq 1, 2). \tag{2}$$

If the food sources N_1 and N_2 act as the source and the sink respectively, the following two equations hold

$$\sum_i Q_{i1} + I_0 = 0, \ \sum_i Q_{i2} - I_0 = 0, \tag{3}$$

where I_0 is the flux from the source node. In [1] this I_0 is set to be a constant.

To model the adaptation of tubular thickness to the flux Q_{ij}, the conductivity D_{ij} is assumed to change in time as

$$\frac{d}{dt}D_{ij} = f(|Q_{ij}|) - rD_{ij}, \tag{4}$$

where $f(Q)$ is a certain increasing function with $f(0) = 0$. By this equation, the conductivity D_{ij} decreases to 0 by its self, but it can be increased when a certain amount of flux exists in the tube M_{ij}. For simplicity, the function f is given as $f(|Q_{ij}|) = \alpha|Q_{ij}|$ and a simple equation of D_{ij} is obtained as

$$\frac{d}{dt}D_{ij} = \alpha|Q_{ij}| - rD_{ij}. \tag{5}$$

Then, by solving Eq. (5) we have D_{ij} at time t; $D_{ij}(t)$. Using this $D_{ij}(t)$, $Q_{ij}(t)$, $p_i(t)$, and $p_j(t)$ are determined by Eqs. (1), (2), and (3). More precisely, Eqs. (1), (2), and (3) contain $M + N$ equations with $M + N$ unknown variables. However, as p_i appears in the form of $p_i - p_j$, it remains arbitrariness. To remove this arbitrariness, we assume $p_2(t) \equiv 0$. This assumption is reasonable because the pressure p_2 is always 0 at the sink node N_2.

2.2. Electric Circuit Analogue

Here, we consider an electric circuit analogue of the mathematical model [1]. Such an analogue can be obtained quite naturally, if we assume the flux Q_{ij} as the current I_{ij}, the pressure p_i as the voltage V_i, and D_{ij}^{-1} as the unit resistance R_{ij}. Then, equivalent circuit equations are obtained for Eqs. (1), (2), (3), and (5) in terms of I_{ij}, V_{ij} ($\equiv V_i - V_j$), and R_{ij}. Circuit analogue of Eqs. (1), (2), and (3) is simple, because these equations represent linear relations of I_{ij} and V_{ij}. On the contrary, circuit analogue of Eq. (5) is nontrivial, because it might not be an easy task to directly implement an Eq. (5) analogue with 'known' simple circuit elements.

Here, using SPICE as a reliable circuit analyzer in mind, we propose a nontrivial circuit analogue of Eq. (5) as follows. First, from Eqs. (1) and (5), an equivalent circuit equation is obtained as

$$\frac{d}{dt}\frac{I_{ij}}{V_{ij}}L_{ij} = |I_{ij}| - \frac{I_{ij}}{V_{ij}}L_{ij}, \tag{6}$$

where L_{ij} is the length of the 'resistance' between node i and j. In Eq. (6), we assume the time evolution of V_{ij} is relatively slow, compared to that of I_{ij} (which is confirmed in simulations, later). Then, for an infinitesimally short time span, V_{ij} becomes a constant and the following is obtained directly from Eq. (6):

$$\frac{d}{dt}I_{ij} = |I_{ij}|\frac{V_{ij}}{L_{ij}} - I_{ij}, \tag{7}$$

where the solution of I_{ij} is given by

$$I_{ij} = I_0\exp\left\{\left(\frac{V_{ij}}{L_{ij}} - 1\right)t\right\}, \tag{8}$$

and

$$I_{ij} = -I_0\exp\left\{\left(\frac{-V_{ij}}{L_{ij}} - 1\right)t\right\}, \tag{9}$$

for the initial $I_{ij} = I_{ij}(t_0) \equiv I_0 > 0$ and for $I_{ij}(t_0) = -I_0 < 0$ respectively.

It is noted that in Eqs. (8) and (9) V_{ij} can be time-dependent as far as its change in time is slow, compared to I_{ij}. What we see in Eqs. (8) and (9) is that the current I_{ij} is given as a strange form of the voltage and time-dependent current source.

However, in finding the unknown shortest path in the network, the directions of links between adjacent nodes (; the direction of current I_{ij}) are not known in advance. Then, we propose a linear combination of Eqs.(8) and (9) to alleviate the above point as

$$I_{ij} = \exp\left\{\left(\frac{V_{ij}}{L_{ij}} - 1\right)t\right\} - \exp\left\{\left(\frac{-V_{ij}}{L_{ij}} - 1\right)t\right\}, \tag{10}$$

where $I_0 = 1$ is assumed in Eqs. (8) and (9). It is noted that Eq. (10) becomes 0 at $t = 0$ and it can be easily implemented as a voltage-dependent current source.

Thus far, we have derived a circuit analogue of the model [1] which can be easily implemented on SPICE. In simulating this circuit on SPICE, we set a constant current source ($\sim$ 10[A]) between node N_1 and N_2, and connect a large resistance R ($\sim$ 100[kΩ]) in parallel to the nonlinear resistive circuit in which each link has the current source of Eq. (10). Then, this path finding circuit is simulated until all currents reach to steady values, which is judged by the time change of all currents becomes less than a threshold ($\sim 1.0 \times 10^{-3}$ for instance).

3. Systematic Comparison of Path Finding Times between the proposed method and the Dijkstra's algorithm

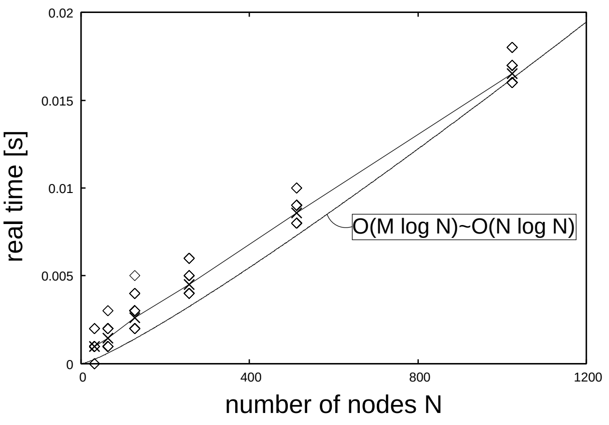

Fig. 2 Elapsed times for the Dijkstra's algorithm

To make a systematic comparison between the proposed method and the Dijkstra's algorithm for its path finding ability, first we have generated randam networks

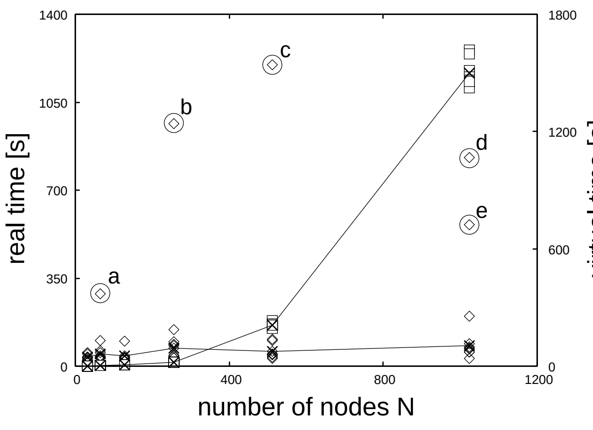

Fig. 3 Elapsed times for the proposed method

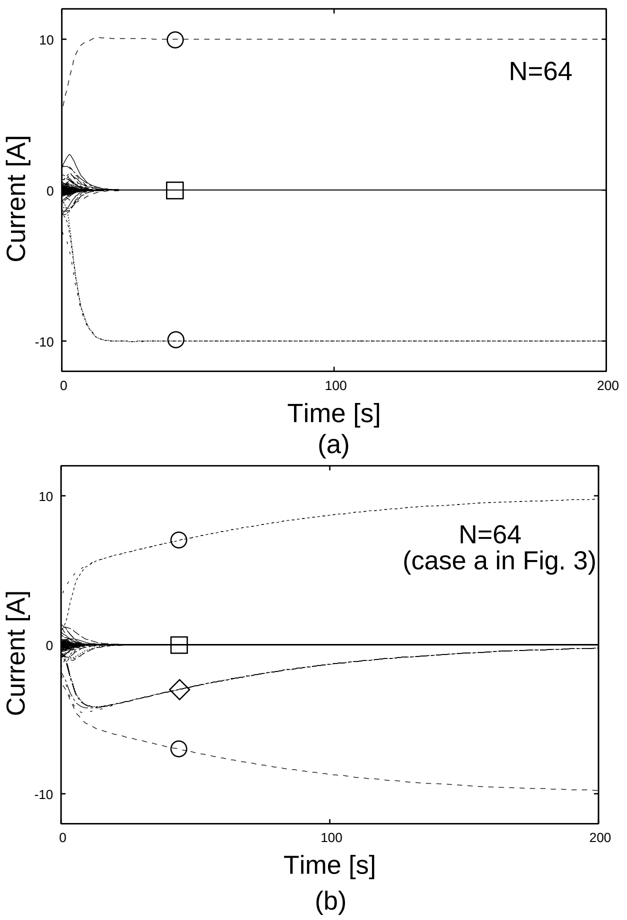

Fig. 4 Path finding process in the proposed method
(a) fast convergence case
(b) slow convergence case

with N nodes and $M(= 4N)$ links for different N; $N = 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}$. For each N, 10 networks are generated and the both two algorithms are applied to each network.

Figures 2 and 3 shows the computation times for the Dijkstra's algorithm and for the proposed method, respectively. In Fig. 2 , we show a comparison between the real elapsed times on PC (plotted with ⋄), their average (×), and virtual elapsed times (; time complexity of $O(M\log N) \sim O(N\log N)$ [3]). As we observe in Fig. 2 , the real elapsed time shows the same increasing tendency with respect to N, compared to the theoretical estimates of $O(N\log N)$. It is noted that the above elapsed times on PC correspond to the essential computation in the Dijkstra's algorithm; we have carefully removed elapsed times which are not directly related to the Dijkstra's algorithm, such as times for initialization and data file generation in the program.

In Fig. 3, a comparison is shown for the real elapsed times on SPICE (plotted with □), virtual elapsed times (⋄; elapsed times in the 'virtual' circuit on SPICE), and their average (×). We observe interesting features on the virtual elapsed times in Fig. 3;

(i) In all cases except for the data points 'a', 'b', 'c', 'd', and 'e' in Fig. 3, the circuit finds the shortest path within around 200[s], irrespective to the number of nodes N.

(ii) The average time (×) for the above cases does not have an apparent increasing tendency with respect to N, namely its time complexity seems to be nearly O(1).

To get an insight to the above observation, we have numerically analyzed the time course of path finding process in the proposed method in detail. Figs. 4(a) and 4(b) show two typical examples from fast convergence cases and slow convergence cases, respectively. As we observe in Fig. 4(a), in all examples except for 'a', 'b', 'c', 'd', and 'e', the shortest path (denoted by ∘ in Fig. 4(a)) can be found within 200[s], while other paths (□) quickly disappear, irrespectively to N. As opposed to these fast convergence cases, we have slow convergence cases 'a', 'b', 'c', 'd', and 'e' as shown in Fig. 3. For all these five cases, we observe two groups of solutions (; ∘ and ⋄ in Fig. 4(b)) are initially competing in path finding process. The appearance of these two competing groups should explain the slow convergence to the final shortest path, a steady solution. Interestingly, we found that these two groups of solutions correspond to the first and the second shortest paths in the network. As shown in Fig. 5, these two competing paths have quite similar path lengths. This fact suggest that the slow convergence in these larger networks comes from the same mechanism analytically explained in small networks.

	1st shortest	2nd shortest
data a	53	54
data b	264	265
data c	577	579
data d	1319	1324
data e	825	829

Fig. 5 First and second shortest paths found for data 'a', 'b', 'c', 'd', 'e' in Fig. 3

From an application point of view, the above observation suggest an interesting nature of the proposed method (and possibly of the model in [1] as well); **competing multiple paths are found simultaneously at an early stage of path finding process**. Works in this direction will be reported elsewhere. The real elapsed time on PC (□) in Fig. 3 is mainly due to the numerical integration scheme in SPICE, which is one of the most reliable, but possibly the computationally heaviest software on PC. Works for accelerating the proposed path finding method are ongoing.

4. Conclusions

We proposed a SPICE-oriented path finding method obtained from [1], which shows quite different characteristics, compared to the Dijkstra's algorithm. One of the interesting characteristics obtained in the proposed method is that **competing multiple paths are found simultaneously at an early stage of path finding process**. We expect more difficult (; NP-complete) problems can be attacked in the spirit of the proposed method thanks to *Physarum*.

Acknowledgments

We deeply thank Drs. A. Tero, T. Saegusa, T. Nakagaki, and R. Kobayashi. One of the authors (H. T.) thanks Dr. K. Yamamura for kind comments on SPICE. This work was supported in part by the Ministry of Education, Science, Sports and Culture, through a Grant-in-Aid, 18560370 (2006).

References

[1] A. Tero, R. Kobayashi, and T. Nakagaki, "Physarum solver: A biologically inspired method of road-network navigation" *Physica A*, 363, pp. 115–119, 2006.

[2] T. Nakagaki, H. Yamada, and A. Toth, "Maze-solving by an amoeboid organism" *Nature*, 407, p. 470, 2000.

[3] T. Ibaraki, "Dijkstra's algorithm for Shortest Path Problem" Ibaraki Laboratory. (online), available from ⟨http://ist.ksc.kwansei.ac.jp/ ibaraki/⟩ , (accessed 2007-2).