A Bio-Inspired Algorithm for Maximum Matching in Bipartite Graphs

Chunxia Qi* *Member, IAENG,* Jiandong Diao

***Abstract*—Recently, an ancient slime mold, *Physarum polycephalum*, has been proved being capable of finding shortest path in physical maze environment, which inspires researchers to extract the core foraging mechanism – positive feedback – to simulate or model such intelligent behavior. Among most well-known mathematical models developed so far, physarum solver, has been adopted and extended to solve a plethora of optimization problems in different fields, including computer science, operations research and transportation etc. In this paper, we first adopt and apply a variant of modified physarum solver, called iterative physarum model, to solve bipartite matching problem. Specifically, the maximum bipartite matching problem is first equivalently transformed to single-source single-sink maximum flow problem. Then the iterative physarum model is used to solve the maximum flow problem adaptively. As iterative physarum solver does not involve solving the systems of linear equations associated with global network flow balance constraints, time complexity of updating node status for one iteration can be reduced from $O(n^3)$ to $O(m)$, where n and m are numbers of nodes and edges in the graph, respectively. Extensive numerical studies on both sparse and complete bipartite graphs demonstrate the validity and efficiency of this method**

***Index Terms*—maximum matching, max flow problem, bio-inspired algorithm, physarum solver, bipartite graph.**

I. Introduction

SOME biological organisms often perform highly intelligent behaviors, which have been inspiring humans to find underlying core mechanism that can possibly explain the intelligence. Recent biological experiments showed that a billion years old slime-mold organism, called *Physarum polycephalum*, exhibited a surprising ability to solve maze and construct high-performance networks [1], [2], [3]. As an aggregate of protoplasm with a network of tubular elements, physarum transports signals and nutrients through its physical body parts. Its tubes act as "legs", helping it to explore and navigate around the physical environment. Specifically, it responses to external conditions, such as locations of nutrient sources, terrain surface, and light brightness, where tubes can disassemble and reassemble within a few hours to optimize the shape of body network to improve the performance of absorbing the available nutrients. For example, when a starved physarum is distributed throughout a maze with "food" only at the entry and exit of the maze, it can quickly concentrate its body network at each food source and occupy only the shortest path from maze entry to maze exit.

Manuscript received Jan. 24, 2019; revised April 10, 2019. The work is partially supported by Shandong Vocational Education Reform Project (No. 2017121) and Project of Shandong Province Higher Educational Science and Technology Program (No. J17RB149).

C. Qi is with Shandong Foreign Trade Vocational College, No. 201 Jufeng Road, Licang District, Qingdao, 266100, China e-mail: chunxiaqi2@163.com.

J. Diao is with Shandong Foreign Trade Vocational College, No. 201 Jufeng Road, Licang District, Qingdao, 266100, China.

Afterward, the inherent mechanisms of physarum were mathematically modeled as continuous time dynamical systems and natural numerical representations of these dynamics can work as algorithms to conquer several graph optimization problems. Among these algorithms, a mathematical model, called *physarum solver* [4], has attracted a lot of researchers attention. It vividly captures the adaptive dynamics that exhibits path-finding behavior in a maze. Specifically to say, the flow through each tube is approximately described as *Poiseuille* flow, which depends on parameters about tubes, such as length, radius, endpoint pressures of an edge and the viscosity coefficient of the flow. They also define a set of variables called conductivity to control the flow quantity on tubes. The conductivity variable acts as tube thickness indicator. By using these variables, the dynamic system evolves and converges to equilibrium state as some edges grow or remain while others disappear. Eventually, the shortest path wins when all flows converge to it. More detail about the classical physarum solver can be found in Section II-B.

Despite of physarum solver's diverse applications on network optimization, none has worked on solving maximum matching problem by physarum solver [6], [7], [9] so far. A *matching* in a graph $G = (V, E, L)$ is to find a subset of edges with the property that no two edges share a common vertex. A graph is bipartite if vertices on graph G can be divided by two disjoint vertex subsets A and B, and no two vertices within the same sets are adjacent. In reality, when modeling relations between two different classes of objects, bipartite graphs arise naturally. For instance, a graph of basketball players and clubs, with an edge between a player and a club if the player has played for that club. Such affiliation networks are very commonly used in social network analysis. A basic version of bipartite matching is as follows: A matching $M = \{m_{ab} | a \in A, b \in B\}$ gives an assignment of a person $a \in A$ to a task $b \in B$, where each person can only work on one task and each task can be handled only by one person. The goal is to get as many tasks done as possible. Mathematically, the objective is to determine a maximum matching: one that contains as many edges as possible in this bipartite graph. A *maximum matching* is a matching $\bar{M}$ such that $|\bar{M}|$ is maximum. That is, $\bar{M}$ is maximum if there is no matching M' such that $|M'| > |\bar{M}|$.

In this paper, we are motivated to extend the classical physarum solver to solve bipartite maximum matching problem via the following two procedures:

1) **Equivalent problem transformation**: we notice that the original maximum matching problem can be converted to single-source single-sink maximum flow problem (MFP). We construct a new graph G' by adding one dummy source node s and one dummy sink node t to a bipartite graph G. Source node s

is connected with the first class of vertices. Sink t is connected with the second class of vertices. Our analysis implies finding the maximum matching in a bipartite graph is equivalent to determining maximum s-t flow in this new graph G'.

2) **Solve MFP by iterative physarum model**: create a path from s to t in the graph G' with "rich" flow on it. We assign it with large enough length to make sure the flow converges to the original graphs. When convergence is reached, the maximum flow is found for G'.

The rest of the paper is organized as follows: in Section II, we briefly introduce the classical physarum solver and its faster variant, called iterative physarum model, which is proposed to accelerate convergence process. In Section III, we elaborate above two equivalent transforming procedures. Section IV includes extensive randomly generated graphs to test the performance of the model; Finally, we draw conclusions and implications for future research in Section V.

II. Preliminaries

In this section, we first briefly describe the classical physarum solver [4] and its modified variant [8], [10]. A simple but fundamental comparison between these two models will be analyzed from the perspectives of time complexity. Later, we will also describe the basic definition of maximum flow problem. Interested readers are referred to some well-known the-state-of-art algorithms developed for MFP [10], [11], [12], [13].

A. Classical Physarum Solver

Variable Q_{ij} is defined to denote the flux through tube (or edge) E_{ij} from node v_i to node v_j. The flow along the tube is assumed approximately *Poiseuille* flow, which can be expressed by the formula,

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

where D_{ij} is the conductivity of tube E_{ij} and p_i is the pressure at node v_i. **The capacity at each node is assumed zero**. Therefore, the flow balance equation at each node except source s and sink t is,

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

and two equations at s and t hold

$$\sum_i Q_{si} = -1, \quad \sum_i Q_{it} = 1, \tag{3}$$

where one unit flux is injected to source s and leave from sink t. Note that, instead of one unit flux, any amount of flux is feasible in the model as long as it is constant from start to finish.

As experimental observation shows that tubes with larger fluxes are reinforced, while those with smaller fluxes disappear the following equation is proposed for simulating such positive feedback behavior,

$$\frac{d}{d\tau} D_{ij} = |Q_{ij}| - r D_{ij}, \tag{4}$$

where r is a decay rate of the tube. This equation implies that conductivity tends to vanish if flux decreases to zero, while it will be enhanced by large flux.

B. Iterative Physarum Model

In iterative physarum model [8], [10], **the assumption of no capacity at each node is relaxed; That is, each node is capable of storing flux temporarily (no up limit theoretically)**. The number of flux stored in a node indicates the flow level of this node's current state, denoted by Φ. The flux tends to flow from nodes of high-level to ones of low-level. To differ from the notations in the class physarum solver, we denote F_{ij} as the flow from node v_i to node ${}_v j$, which is formulated as,

$$F_{ij} = D_{ij} \times \frac{\Phi_i - \Phi_j}{L_{ij}}, \tag{5}$$

As there is inflow importing to the node and outflow pouring out simultaneously, the flow level at the node i is updated as,

$$\Phi_i^{\tau+\delta\tau} = \Phi_i^{\tau} + \sum_{e \in E_i} F_e^{\tau}, \tag{6}$$

where E_i is the set of edges adjacent to node i, the upper index τ denotes a time step and $\delta\tau$ is time mesh size.

Although Eq. (5) looks similar to Eq. (1), the flow process performs in a totally different manner. In classical physarum solver, the pressure maintained at each node must keep the conversation law of flow satisfied globally, which requires solving the system of linear equations for pressure variables p. Meanwhile, the time complexity of solving the system of linear equations by standard elimination method is close to $O(n^3)$. However, the iterative physarum model has the flow levels for all nodes updated locally according to their inflow and outflow. Simply speaking, the process will start from the source s by injecting flow, update flow level for neighbors "iteratively" layers by layers, and end at sink t, which can be run obviously in $O(m)$ time. Note that, due to non-negativity of both L_{ij} and D_{ij}, the flow direction along edge E_{ij} is known from the sign of flow level difference straightforward, that is, if $\Phi_i \geq \Phi_j$, the energy flows from node i to node j; Otherwise it takes the opposite direction.

Assume flow enters the source node s in a constantly fixed rate (usually one unit per time step) and exits from sink node in same rate, thereby we can envision that flow injected to network will become stable as long as the network is s-t path-connected (you can compare it with pipe network in real world). The following *evolution equation* is similarly provided to capture the positive feedback behavior,

$$\frac{d}{dt} D_{ij} = f(E_{ij}) - D_{ij}. \tag{7}$$

For the sake of numerical computing, the iterative physarum model adopts the semi-implicit scheme of *evolution equation* as follows

$$\frac{\Phi_i^{\tau+\delta\tau} - \Phi_j^{\tau}}{\delta\tau} = E_{ij}^{\tau} - D_{ij}^{\tau+\delta\tau}. \tag{8}$$

The iterative physarum model was first proposed to solve shortest path problem [8]. The primary benefit it brings is to accelerate the convergence process as the classical physarum solver have need of solving high complexity linear equations system [10].

C. Maximum Flow Problem

The maximum flow problem is defined on a capacitated directed network $G = (V, E, L, C)$, where $C = \{C_e \mid e \in E\}$ denotes the nonnegative capacity of edge E_{ij}. The objective is to send flow as much as possible from source s to sink t. Let f represent the amount of flow in the network. Then, the maximum flow problem can be formulated as,

$$\max \quad f \tag{9a}$$

$$\text{s. t.} \quad \sum_{j=1}^{n} x_{ij} = f, \qquad i = s, \tag{9b}$$

$$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = 0, \ i \neq s, i \neq t, \tag{9c}$$

$$\sum_{j=1}^{n} x_{ji} = f, \qquad i = t, \tag{9d}$$

$$0 \leq x_{ij} \leq C_{ij}, \qquad i, j = 1, 2, ..., n, \tag{9e}$$

where the sums and inequalities are taken over all edges in the network. All feasible flows must satisfy the above capacity constraint and flow conservation constraint.

Here we provide some classical algorithms proposed for MFP. First, Ford-Fulkerson algorithm [11] keeps finding augmenting path until termination, which can be implemented in $O(m \max(|f|))$. Dinic's algorithm [12] utilizes a dynamic trees data structure to speed up the maximum flow computation in the layered graph to $O(nm \log(n))$. Push-relabel algorithm [13] always selects the most recently active vertex, and performs push operations until the excess is positive or there are admissible residual edges from this vertex, which is run in $O(n^3)$. As mentioned in [10], the main difference between iterative physarum model and these typical MFP algorithms is that the flow plays a quite intelligent role; That is, it can be adaptively pushed back to the original network until the maximum flow is reached and no more flow can enter. Such every unit of flow that contributes to the max flow can be "monitored" during this pushing-back process.

III. Methodology

In this section, we will first transform matching problem in bipartite graphs into max flow problem. Then, the iterative physarum model will be modified and extended to solve MFP. Such modification to iterative physarum solver will significantly improve the efficiency of solving max flow problem comparing with classical physarum solver, which will be demonstrated by extensive numerical studies in Section IV.

A. Equivalent Problem Transformation

By letting the capacity of each edge from $a \in A$ and $b \in B$ be 1, we claim that solving the maximum flow problem with multiple sources A and multiple sinks B is equivalent to finding the maximum matching problem with corresponding bipartite nodes sets A and B. The inherent logic is trivial: if the maximum flow from A to B is larger than the current feasible matching, we can find a better match according to the maximum flow since every unit of flow from A and B can be numerically regarded as matching from A to B. If

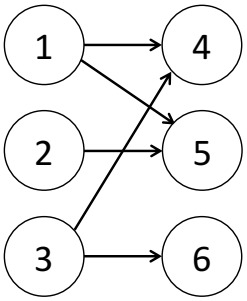

Fig. 1: A simple bipartite graph G with two separate sets of nodes $A = \{1, 2, 3\}$ and $B = \{4, 5, 6\}$. Each edge $E_{ab}, a \in A, b \in B$, has length $L_{ab} = 1$ and flow capacity $C_{ab} = 1$.

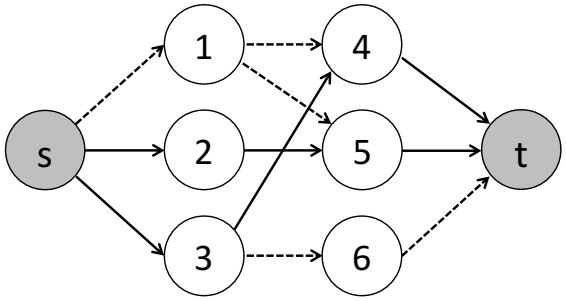

Fig. 2: Transform maximum matching problem to maximum flow problem: add two dummy nodes s and t and connect A and B respectively. This new graph G' has $L_e = 1, C_e = 1$, for every edge $e \in E$. A feasible flow is assigned to two paths of solid lines while dash lines indicate zero flow. The corresponding matches for this feasible flow is m_{25} and m_{34}.

current matching value is larger than the maximum flow, current feasible flow is not optimal since the matching will provide a better feasible flow. Therefore, solving MFP equals to solving maximum matching problem on the same instance.

In order to apply the iterative physarum model to solve MFP such that maximum matching problem can be conquered equivalently, we implement the following equivalent transformation: Given a bipartite graph (see Fig. 1), two dummy vertices s and t are added to the network, respectively. Connect s with all vertices in the first set A and connect t with all vertices of the second set B (see Fig. 2). Image two units of flows, $F = \{s \to 3 \to 4 \to t, s \to 2 \to 5 \to t\}$, are pushed from s to t. Apparently, it is not the maximum flow (3 units), thereby current matching $M = \{m_{34}, m_{25}\}$ is suboptimal.

B. Solving Maximum Flow Problem

In this section, we apply the iterative physarum model to solve maximum flow problem according to the recipe [10]. First, a new network G^* is constructed by adding a dummy node d to connect the source s and sink t via two virtual edge E_{sd} and E_{dt} on G'. By setting $L_{sv} = L_{vt} = nl_m/2$, where l_m is the length of the longest edge among M, the additional path $s \to v \to t$ must be longer than any other simple path (connected and acyclic) from s to t. The capacities of edges E_{sd} and E_{dt} are $C_{sv} = C_{vt} = \min(\sum_{e \in E_s} C_e, \sum_{e \in E_t} C_e)$. The total inflow, denoted by I_0, is equal to C_{sv}. By running

the iterative physarum model on G^*, the flow will converge to the edges in G' as much as possible by complying with some certain constraint conditions below. For an unsaturated edge, the flow should follow Eq. (5). The maximum flow of a saturated edge should be lower than C_{ij}. Hence, the flow equation (5) can be rewritten as

$$F_{ij} = \begin{cases} \frac{D_{ij}}{L_{ij}}(\Phi_i - \Phi_j) & \text{if } F_{ij} < C_{ij} \\ C_{ij} & \text{if } F_{ij} = C_{ij} \end{cases} \tag{10}$$

Consequently, the conductivity of edge E_{ij} evolves according to the equation as

$$\frac{d}{d\tau}D_{ij} = \begin{cases} F_{ij} - D_{ij} & \text{if } 0 \le F_{ij} \le C_{ij} \\ C_{ij} - D_{ij} & \text{if } C_{ij} < F_{ij} \end{cases} \tag{11}$$

To ensure most flow is tracing the virtual edges mostly in the first iteration, the conductivity $D_{ij}(\tau = 0)$ of each edge in G' initialize as

$$D_{ij}^{\tau=0} = \begin{cases} C_{sv}^{\infty} & \text{if } i = v \text{ and } j = v \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

where C_{sv}^{∞} is sufficiently large number.

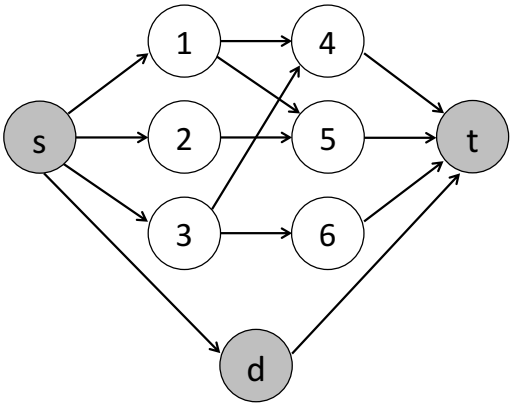

Fig. 3: solve maximum flow problem by iterative physarum model on a new graph G^* – add another dummy node d and connect it with s and t. The length and capacity for two new edges E_{sd} and E_{dt} are specified in Section III-B.

As shown in Fig. 3, we add one dummy vertex d to connect source s and sink t. Initially, all the flows are assigned on this additional path. As the process continues, more flux will flow back to the original graph until reaching no more flow can be pushed back to G'.

IV. Simulation Experiment

In this section, we examine the efficiency of the proposed model on different sizes of instances. We adopt two types of instances: one is "sparse graph" case and the other is "complete graph" case. The sparse graphs are generated by the following setting: a node $a \in A$ connects to a node $b \in B$ with a certain probability pr. For complete bipartite graphs, every node $a \in A$ must be connected with every node $b \in B$. The computational experiments are performed on a PC with Intel Core I7-8850HQ 6 Core Processor with $vPro^{tm}$ and 16 GB of memory, running Windows 10. All algorithms adopted for performance comparison are implemented in Matlab programming language of version 2017a.

In Section IV-A, we will first compare our proposed iterative physarum solver (IPM) with linear programming model (LP), called simplex method, and Ford-Fulkerson (FF) algorithm. Section IV-B will further examine the effectiveness between iterative physarum model and the classical physarum solver from the perspectives of the number of iterations and computation time.

A. Linear Programming and Ford-Fulkerson Algorithm

The statistical information of instances is reported Table I. The first column reports instance name. For example, instance "S-01" represents the first sparse graph we generate. The cardinalities of A and B are reported in second column and third column, respectively. The forth column reports the number of edges m in the corresponding sample according to the connection probability pr in the fifth column. Additionally, complete graphs are also provided, which are reported in last three columns.

TABLE I: The basic topological features of two sets of graphs – 10 sparse graphs and 10 complete graphs. m is number of edges between node set A and node set B and pr is the probability that a node $a \in A$ is connected with $b \in B$.

Sparse graphs					Complete graphs		
Instance	$\|A\|$	$\|B\|$	m	pr	Instance	$\|A\|$	$\|B\|$
S-01	10	20	70	0.2	C-01	10	20
S-02	50	60	798	0.05	C-02	50	60
S-03	100	140	3423	0.01	C-03	100	140
S-04	200	250	3345	0.01	C-04	200	250
S-05	400	500	8631	0.01	C-05	400	500
S-06	500	650	43923	0.01	C-06	500	650
S-07	800	880	45623	0.01	C-07	800	880
S-08	1000	1160	69893	0.01	C-08	1000	1160
S-09	1500	1560	173345	0.01	C-09	1500	1560
S-10	2000	2000	578321	0.01	C-10	2000	2000

The computational results for IMP, LP and FF are reported in Table II. Our method outperforms the simplex method for most sparse and complete instances, especially on graphs S-07 and S-08. Note that, the simplex method usually requires extra time for building the model, i.e., preprocess some redundant constraints or variables for acceleration. Hence, for larger graphs, simplex method could do better since the "pure" time (total CPU time minus preprocessing time) is actually smaller than what we expect. Additionally, the traditional Ford-Fulkerson algorithm outperforms the IMP and LP on all sparse and complete graphs.

B. Iterative Physarum model vs. Physarum Solver

In this section, we generate five sets of complete graphs, where in each graph $|A|$ and $|B|$ are equal. The sizes of each graph can be found from the graph names (first column of Table III). For example, graph C-250-01 means the first (-01) graph of complete graph (C) with 500 nodes ($|A| = |B| = 250$). Columns 2 and 4 report the total number of iterations for either solving the linear equations system

TABLE II: The CPU time(second) of iterative physarum model (IPM), linear program (LP) and Ford-Fulkerson (FF) methods on both sparse and complete graphs

Sparse graphs				Complete graphs			
Instance	IPM	LP	FF	Instance	IPM	LP	FF
S-01	0.01	0.02	0.01	C-01	0.01	0.02	0.01
S-02	0.31	0.22	0.01	C-02	0.52	0.32	0.77
S-03	0.42	0.87	0.33	C-03	0.34	0.56	0.99
S-04	1.13	2.21	1.32	C-04	1.91	2.02	1.71
S-05	1.32	3.32	1.71	C-05	2.35	2.24	1.91
S-06	1.57	4.22	2.11	C-06	2.54	10.19	3.31
S-07	2.32	10.14	2.34	C-07	3.53	15.31	3.32
S-08	7.46	14.52	4.11	C-08	9.32	18.72	6.43
S-09	16.72	15.33	5.23	C-09	18.43	16.02	6.47
S-10	20.17	19.32	5.32	C-10	21.54	20.02	8.84

(in PS) or update the flow level for all nodes (in IPM) for convergence. As we can see, no matter what the graph size is, the number of iterations in PS varies insignificantly. For IPM, the number of iterations increases with the graph closely. However, the IPM outperforms PS significantly with respect to computation time on all graphs. This is because larger size of graphs implies solving large-size linear equations system, even though total iterations are less than IPM's.

TABLE III: The comparison between iterative physarum model (IPM) and classical physarum solver (PS)

Instance	IPM		PS	
	Iterations	Times(s)	Iterations	Times(s)
C-50-01	127	1.13	242	2.31
C-50-02	142	1.28	322	4.18
C-50-03	85	0.78	187	1.97
C-50-04	101	1.03	410	3.99
C-50-05	123	1.12	331	3.73
C-250-01	157	1.45	241	5.42
C-250-02	167	2.56	302	6.12
C-250-03	151	3.01	287	5.89
C-250-04	182	2.69	295	5.42
C-250-05	153	1.74	303	6.01
C-450-01	220	3.76	321	10.21
C-450-02	215	4.01	283	12.12
C-450-03	334	5.92	304	11.21
C-450-04	267	2.34	321	10.53
C-450-05	214	3.62	286	7.33
C-650-01	350	7.09	289	14.21
C-650-02	358	8.62	334	15.46
C-650-03	335	6.12	201	14.83
C-650-04	389	7.52	402	16.26
C-650-05	310	7.32	334	17.15

V. Conclusions

In this paper, we apply an novel bio-inspired physarum model, to solve maximum matching problem on bipartite graphs. First, we transform the maximum matching problem into maximum flow problem by adding a dummy node to the original bipartite graph. Second, the maximum flow problem on the new graph can be solved by using the iterative physarum model. Numerical experiments demonstrate that iterative physarum model can solve the problem precisely and reach the optimality definitely.

Meanwhile, we also notice that solving the maximum matching problem in "general" graphs is not easy. Current physarum models are still not capable of solving it. In the near future, more works should be focused on how to generalize the model and apply to solve a broad range of general matching problems. Moreover, there is still room for improving the computational efficiency of iterative physarum model. For example, we can do parallel computation when updating the flow level at each node, which will significantly reduce the computation time.

Acknowledgment

C. Qi wrote the manuscript. Both C. Qi and J. Diao carried out the experiment. The work is partially supported by Shandong Vocational Education Reform Project (No. 2017121) and a Project of Shandong Province Higher Educational Science and Technology Program (No. J17RB149).

References

[1] T. Nakagaki, H. Yamada, and A. Tóth, "Intelligence: Maze-Solving by an Amoeboid Organism," *Nature*, vol. 407, no. 6803, 2000.

[2] A. Tero, S. Takagi, T. Saigusa, K. Ito, D.P. Bebber, M.D. Fricker, K. Yumiki, R. Kobayashi and T. Nakagaki, "Rules for Biologically Inspired Adaptive Network Design," *Science*, vol. 327, no. 5964, pp. 439–442, 2010.

[3] A. Tero, R. Kobayashi and T. Nakagaki, "Physarum solver: A Biologically Inspired Method of Road-network Navigation," *Physica A*, vol. 363, pp. 115–119, 2006.

[4] A. Tero, R. Kobayashi and T. Nakagaki, "A Mathematical Model for Adaptive Transport Network in Path Finding by True Slime Mould," *Journal of Theoretical Biology*, vol. 244, pp. 553–564, 2007.

[5] C. Gao, C. Yan, A. Adamatzky and Y. Deng, "A Bio-inspired Algorithm for Route Selection in Wireless Sensor Networks," *IEEE Communications Letters*, vol. 18, no. 11, pp. 2019-2022, 2014.

[6] Q. Liu, "On Maximal Incidence Energy of Graphs with Given Connectivity," *IAENG International Journal of Applied Mathematics*, vol. 48, no.4, pp. 429–433, 2018.

[7] X. Chen and S. Liu, "Adjacent Vertex Distinguishing Proper Edge Colorings of Bicyclic Graphs," *IAENG International Journal of Applied Mathematics*, vol. 48, no.4, pp. 401–411, 2018.

[8] X. Zhang, C. Gao, Y. Deng and Z. Zhang, "Slime Mould Inspired Applications on Graph-optimization Problems," *Advances in Physarum Machines*, Springer, Cham, 2016, pp. 519–562.

[9] T. Huang, L. Zuo, and C. Shang, "The Linear k-Arboricity of Cartesian Product of Multipartite Balanced Complete Graphs," *IAENG International Journal of Applied Mathematics*, vol. 48, no.3, pp. 362-367, 2018.

[10] Z. Wang and D. Wei, "Solving the Maximum Flow Problem by a Modified Adaptive Amoeba Algorithm," *IAENG International Journal of Computer Science*, vol. 45, no.1, pp. 130–134, 2018.

[11] L. R. Ford Jr and D. R. Fulkerson, "Flows in Networks," *Princeton university press*, 2015.

[12] E. A. Dinic, "Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation," *Soviet Math Doklady*, vol. 11, 1970.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," *MIT Press*, 2009