Abstract—Maximum flow problem is one of the most fundamental network optimization problems. Recently, experimental observations showed that an amoeboid organism, Physarum polycephalum, contains a tube network by means of nutrients and signals circulating through the body. The tube network can sense and adopt to local shear stress difference in its own body until the shortest tubes of connecting two food sources (placed at two exits of a maze) keep alive while longer tubes vanish eventually. Unlike to the global optimization algorithm Physarum solver, we develop a mathematical model of a dynamical system which capture the local control behavior of Physarum when searching for foods. With this new model, a novel adaptive amoeba algorithm for maximum flow problem is proposed, which can be used to solve shortest path problem as Physarum solver. Furthermore, we firsts apply this model to solve maximum flow problem, where its convergence is proved in this paper as well. Additionally, numerical results demonstrate the validity and efficiency of the proposed algorithm to solve maximum flow problem.

Index Terms—maximum flow problem, bio-inspired algorithm, network optimization, physarum solver.

I. INTRODUCTION

NETWORK flow problems form a large class of optimization problems and are central problems in operations research, computer science, civil engineering and combinatorial optimization [1], [2], [3], [4], [5]. Among them, one of the most famous problems is maximum flow problem, which has many applications in transportation, logistics, telecommunications, and scheduling etc. The maximum flow problem was first formulated by Ted Harris and F. S. Ross when they studied a simplified model of Soviet railway traffic flow in 1954 [6] as follows. Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, the flow is maximal from one given city to other this moment.

To the best of our knowledge, the existing algorithms for maximum flow problem can be classified in two major classes. One is the classical augmenting path method. Ford and Fulkerson devised the first algorithm of feasible flow by working with augmenting path incrementing the flow at every iteration [7]. This algorithm is based on the fact that a flow is maximum if and only if there is no augmenting path. It repeatedly finds an augmenting path and augments along it until no augmenting path exists anymore. The other is the preflow algorithm. A preflow is a flow that seems to violate the restriction that the incoming flow and outgoing flow should be balanced. The push-relabel algorithm is a representative of this type which is efficient both theoretically and empirically [8]. A comprehensive discussion of such algorithms and applications can be found in [9]. Note that max flow problem is a fundamental problem. Numerous efficient algorithms for this problem have been proposed to reduce the runtime or time complexity. However, for general cases, only small improvements have been made. In this paper, we try our best to solve maximum flow problem from other perspective, that is, is there a different type of method for maximum flow problem other than the the two classical algorithms?

Recently, studies of Physarum polycephalum (P. polycephalum) have attracted huge attention. The plasmodium of P. polycephalum is a large amoeba-like organism with great intelligence. Studies have shown that it has the ability to solve a complex maze [10], [11] and other graph theoretical problems [12], [13], [14], [15]. When food sources (FSs) placed at two exits are presented to a starved P. polycephalum, it covers FSs to absorb nutrients and constructs a tube network by means of which nutrients and signals circulate through the body. Later, only a few short tubes remain which means effectiveness for transportation. Tero et al. develop a mathematical model Physarum solver to describe the above adaptation process of the tube network [16]. The insight essence of Physarum solver is rhythmic oscillation which is exactly described by positive feedback mechanism between the thickness of each tube and internal protoplasmic flow. Physarum solver has been applied in many fields [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. In the dissertation [28], Wang is the first to apply physarum solver to solve maximum flow problem. Wang claims that her developed algorithm is easy to implement, and more importantly, it can obtain maximum flow and minimum cut sets simultaneously.

Even though significant progress has been made, it remains challenging to achieve novel methods to solve maximum flow problem. In this paper, we are motivated to develop a biologically inspired algorithm for maximum flow problem based on Physarum solver. As we know, it is the first time that Physarum solver is developed to solve maximum flow problem. It is very interesting to see how this new amoeba model works when solving a classical network flow problem. Second, we also note that the adaptation principle of Physarum solver follows the property of collaborating simultaneously and globally. However, it is observed in experiment that P. polycephalum adapts tube diameters in
response to wall shear stress [16], [18]. Based on these facts, we simulate the adaptation process of *Physarum* and develop a similar but novel mathematical model where a dynamical system works quite like *Physarum* but responses to local information. Third, the proposed model is proved to efficiently solve maximum flow problem by testing on instances with relatively large size when compared with *Physarum solver* [16].

The rest of the paper is organized as follows. In Section 2, some definitions and preliminary descriptions are introduced. Section 3 describes the proposed mathematical model and applies it to solve maximum flow problem. Section 4 gives the proposed algorithm, in Section 5. Finally, concluding remarks and several large-size instances to test the efficiency of the proposed algorithm, in Section 6. Finally, concluding remarks and several large-size instances to test the efficiency of the proposed algorithm, in Section 6.

II. PRELIMINARIES

In this section we introduce some basic knowledge about maximum flow problem and the well-known theorem, “Max-flow Min-cut Theorem”.

A. Maximum Flow Problem

The maximum flow problem is defined on a capacitated directed network $G = (N, M, L, C)$ with a set of $n$ nodes and $m$ directed edges. Specifically, $M_{ij}$ denotes the edge with a direction from node $i$ to node $j$. Its length is written as $L_{ij}$, $C_{ij}$ denotes the capacity of edge $M_{ij}$, which is always nonnegative. For edge $M_{ij}$, the lower bound of the feasible flow is zero and an upper bound on flow is $C_{ij}$. In such a network, the maximum flow problem is to send maximum possible flow from a source $s$ to a sink $t$. Let $f$ represents the amount of flow in the network. Then, the maximum flow problem may be expressed as follows,

$$
\text{maximize } f
$$

subjecting to

$$
x_{ij} - \sum_{j=1}^{n} x_{ji} = \begin{cases} 
  f & \text{if } i = s, \\
  0 & \text{if } i \neq s \text{ or } t, \\
  -f & \text{if } i = t,
\end{cases}
$$

where $0 \leq x_{ij} \leq C_{ij}$, $i, j = 1, 2, ..., n$. The sums and inequalities are taken over all edges in the network. Every feasible flow $f$ must satisfy the above capacity constraint and flow conservation constraint.

B. Max-flow Min-cut Theorem

Let two mutually exclusive subsets $S, T$ in $N$, that is, $S, T \subset N$, $S \cup T = N$ and $S \cap T = \emptyset$. Considering any two nodes $p \in S$ and $q \in T$, the cut $[S, T]$ is defined as

$$
[S, T] = \{ e_{pq} \in M | p \in S, q \in T \}
$$

and the capacity of cut $[S, T]$ is defined as

$$
C(S, T) = \sum_{e \in M} C_e.
$$

The following is the well-known max-flow min-cut theorem.

**Theorem 2.1:** The maximum value of $|f|$ is equal to the minimum capacity of cut $[S, T]$, that is, $|f| = C(S, T)$.

III. AN ADAPTIVE AMOEBA ALGORITHM

Given a resistance network $G$, a particle is assumed with one unit of energy which enters at the starting node $s$. This particle traverses through the network and leaves from the sink node $t$. Each node has an ability of storing particles temporarily. The number of particles stored in a node indicates the energy level of this node’s current state. These particles flow to nodes with lower energy levels and leave from the sink node $t$. Furthermore, we define energy flow $E_{ij}$ transferred from node $i$ to node $j$ as

$$
E_{ij} = \Phi_i - \Phi_j = \frac{D_{ij}}{L_{ij}}(\Phi_i - \Phi_j),
$$

where $\Phi_i$ is the energy level of node $i$, $L_{ij}$ length of edge $M_{ij}$, and $D_{ij}$ is the conductivity of edge $M_{ij}$, $L_{ij}/D_{ij}$ can be regarded as the energy resistance. Due to non-negativity of both $L_{ij}$ and $D_{ij}$, the direction of the energy flow $E_{ij}$ is determined by the sign of $\Phi_i - \Phi_j$. If $\Phi_i > \Phi_j$, the energy flows from node $i$ to node $j$; otherwise it goes to the opposite direction.

For the source $s$, particles enter at a rate of $I_0$ and disappear at the sink node with equal rate. The total particle flow $I_0$ is a fixed constant in our model. In order to describe the positive feedback behavior of the system, the conductivity $D_{ij}$ varies in response to energy drop with time according to the following equation,

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

where $f(E_{ij})$ is the driving power that incurs in energy propagation and satisfies $f(0) = 0$. In this paper, we set $f(E) = E$. Then, the semi-implicit scheme of evolution equation can be expressed as

$$
\Phi^t_{i+1} - \Phi^t_i = \frac{E^t_{ij}}{\delta t} - D^t_{ij}
$$

where $\delta t$ is a time mesh size and the upper index $t$ denotes a time step. As energy inflow and outflow incur in each time step, the amount of energy stored in each node is updated as

$$
\Phi^t_i = \Phi^t_{i-1} + \sum_{e \in M_i} E^t_e
$$

where $M_i$ is the set of adjacent edges of node $i$.

In a word, the network full of particles can be viewed as a system of energy propagation on the basis of aforementioned defined rules. So far, we describe a mathematical model for a dynamical system in response to local information (node energy difference). Similar to *Physarum solver*, the proposed model can solve shortest path problem [26]. However, our model adopts to energy level difference between each pair of nodes while the flows in *Physarum solver* are computed by solving a systems of linear equation. The time complexity of solving it is $O(n^3)$, but our model computes the flows in $O(m)$ at each iteration.

IV. SOLVING MAXIMUM FLOW PROBLEM

In this section, we apply the above mathematical model to solve maximum flow problem. First, a new network $G'$ is constructed by adding a dummy node $v$ and two virtual edge $M_{sv}$ and $M_{vt}$. Such additional path $s\rightarrow v\rightarrow t$ is longer than any other possible path between $s$ and $t$ by setting $L_{sv} = L_{vt} =$

(Advance online publication: 10 February 2018)
\[ n \max(L)/2, \text{ where } \max(L) \text{ is the length of the longest edge among } M. \text{ The capacities of edges } M_{st} \text{ and } M_{ut} \text{ are } C_{sv} = C_{vt} = \max(\sum_{e \in \mathcal{E}} C_e). \text{ The total inflow } I_0 \text{ is equal to } C_{sv} \text{ and } C_{vt}. \text{ On the basis of the positive feedback between the energy flow } E_{ij} \text{ and the energy conductivity } D_{ij}, \text{ we want to the flow converges to the edges in } G \text{ 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}. \text{ As a consequence, the energy flow Eq. (5) is rewritten as}

\[ E_{ij} = \begin{cases} \frac{D_{ij}}{E_{ij}}(\Phi_i - \Phi_j), & M_{ij} < C_{ij}, \\ C_{ij}, & M_{ij} = C_{ij}. \end{cases} \quad (9) \]

Consequently, the convertibility of edge \( M_{ij} \) evolves according to the equation as

\[ \frac{dD_{ij}}{dt} = \begin{cases} E_{ij} - D_{ij}, & 0 \leq E_{ij} < C_{ij}, \\ C_{ij} - D_{ij}, & E_{ij} > C_{ij}. \end{cases} \quad (10) \]

In order to ensure flow tracing the virtual edges mostly at each edge in \( G' \), initialize as

\[ D_{ij}(0) = \begin{cases} \frac{2L_{ij}C_i}{\min(e \in M L_{ie}C_e)}, & i = v \text{ or } j = v, \\ 1, & \text{otherwise}, \end{cases} \quad (11) \]

where \( \min(e \in M L_{ie}C_e) \) is the minimum value of the length of each edges in \( M \) times its capacity, respectively. Later, the flow will converge to the path from \( s \) to \( t \) in the network \( G \) as much as possible. When the flow in the \( G' \) is steady, the maximum flow is obtained.

**V. PROOF**

In this section, we prove that the flow in the network \( G \) is the maximum flow when the dynamical system is stable. Let \( \tau \) be the stabilization time. The stabilization of the system means \( \forall t > \tau, dD_{ij}^t/\text{d}t = 0 \).

**Lemma 5.1:** In an equilibrium point, if \( E_{ij} = 0 \), then \( \Phi_i - \Phi_j < L_{ij} \); if \( 0 < E_{ij} < C_{ij} \), then \( \Phi_i - \Phi_j = L_{ij} \); if \( E_{ij} = C_{ij} \), then \( \Phi_i - \Phi_j \geq L_{ij} \).

**Proof:** If \( \forall t > \tau, E_{ij}^t = 0 \) and \( D_{ij}^t \neq 0 \), by Eq. (10), \( D_{ij}^t \) decreases with time, which is inconsistent with the default assumption. As a consequence, \( dD_{ij}^t/\text{d}t < 0 \). According to Eqs. (9) and (10), we get \( dD_{ij}^t/\text{d}t = E_{ij}^t - D_{ij}^t = (\Phi_i^t - \Phi_j^t)/L_{ij} - 1)D_{ij}^t < 0 \). Thus, we get \( \Phi_i^t - \Phi_j^t > L_{ij} \). If \( 0 < E_{ij}^t < C_{ij} \), there is \( D_{ij}^t/\text{d}t = 0 \). Based on Eqs. (9) and (9), it can be seen that \( E_{ij}^t = D_{ij}^t \). Then we get \( \Phi_i^t - \Phi_j^t > L_{ij} \). By solving the lower differential equation of (10) when \( E_{ij}^t = C_{ij} \), we have \( \lim_{t \to \infty} D_{ij}^t = \lim_{t \to \infty}(C_{ij} + (D_{ij}^0 - C_{ij}) \exp(-t)) = C_{ij} \). By Eq. (9), there is \( \Phi_i^t - \Phi_j^t > L_{ij} \).

**Lemma 5.2:** In an equilibrium point, \( \Phi_s - \Phi_t = L_{sv} + L_{vt} \).

**Proof:** According to Lemma 5.1, if the total flow \( I_0 \) is large enough to ensure the flow on the virtual edges satisfy \( 0 < E_{ij} < C_{ij} \), we will have \( \Phi_s - \Phi_t = (\Phi_s - \Phi_v) + (\Phi_v - \Phi_t) = L_{sv} + L_{vt} \). Generally speaking, the maximum flow in the network \( G \) is lower than the value of total incoming flow \( I_0 \) we set above.

**Lemma 5.3:** Assume a path from \( s \) to \( t \) \( (s, N_k, s_2, ..., N_k, t) \) in \( G \) in an equilibrium point. Then, there exists at least one saturated edge.

**Proof:** According to Lemma 5.2, we have \( \Phi_s - \Phi_t = (\Phi_s - \Phi_k_1) + (\Phi_k_2 - \Phi_k_3) + ... + (\Phi_k_n - \Phi_t) \). Due to \( L_{sv} + L_{vt} \) is larger than any path from \( s \) to \( t \) in \( G \), so \( \Phi_s - \Phi_t > L_{sv} + L_{vt} \). Therefore, there must be at least one edge that satisfies \( \Phi_s - \Phi_j > L_{ij} \). It means that there is a saturated edge \( M_{ij} \), that is, \( E_{ij} = C_{ij} \). For edge \( M_{ij} \), in turn, there is \( \Phi_j - \Phi_t \leq -L_{ij} \). By Eq. (10), we get \( E_{ij} = 0 \).

Let \( [S, T] \) is a cut of \( G \) from a source node \( s \) to a sink node \( t \). Then, we have

\[ [S, T] = \{e_{ij} \mid e_{ij} \in M, i \in S, j \in T\}, \]

\[ [T, S] = \{e_{ij} \mid e_{ij} \in M, i \in T, j \in S\}. \]

**Theorem 5.4:** In an equilibrium point, for \( e_{ij} \in [S, T] \), each edge is saturated; for \( e_{ij} \in [T, S] \), \( e_{ij} \) is empty.

**Proof:** According to Lemma 5.3, both saturated edges and empty ones can be cut off. So these edges include a cut of \( G \). Furthermore, all saturated edges are from \( S \) to \( T \) and all the empty edges are in the opposite direction.

According to Theorem 2.1, the flow in the network \( G \) is obviously equal to the maximum flow when the dynamical system is at equilibrium state.

**VI. SIMULATION EXPERIMENT**

**A. A simple example**

For Fig. 1, node 1 and node 11 are set as the source \( s \) and the sink \( t \), respectively. The dummy node \( v \) is added to squeeze the flow into the original network. The total input flow is set as \( I_0 = C_{sv} = C_{vt} = 25 \) and the length of the virtual edge is \( L_{sv} = L_{vt} = 5.5 \). As illustrated in Fig. 1(b), most part of the flow traces among the virtual path \( s \rightarrow v \rightarrow t \) at the beginning. Afterwards, the value of flow on virtual path decreases and converges to 2 in the equilibrium point. Flows on edges \( M_{s2}, M_{s3}, M_{s4} \) are increasing and converge to some fixed value after a certain amount of time. As a result, the value of maximum flow in Fig. 1(a) is obtained \( I_0 - 2 = 23 \), which is the correct maximum flow that the network could have.

**B. Large-scale Instances**

In order to test the efficiency of the proposed algorithm, we adopt large max flow problems instances. The instances were generated by G. Skorobohatyi using the program RMFGEN.
In table II, the computational results are listed. Clearly, for instances elist96 and elist96d, the PS algorithm reach to the optimal solution without any gap with even less computational time. Note that, instances with same number of nodes but different number of edges could lead to an significant different computation time. More edges could affect the performance of the PS in a large extent. From the other instances’ computational results, we can generally say that more edges lead to more runtime. This is because the particles traverses the networks and need more time to reach the end. FF and LP methods perform much better than PS and AA in the computational speed. We are not surprised by the longer runtime of PS because solving the flow conservation constraint is truly to solve a system of linear equations. It can be solved by $O(n^3)$ by using Gaussian elimination method.

**VII. CONCLUSIONS**

In this paper, we have proposed a novel adaptive amoeba algorithm for maximum flow problem. The insight essence of this algorithm is the positive feedback mechanism between the particle flow $E$ and the conductivity $D$: greater conductivity results in great flow, and this increases conductivity in turn. Distinct from preflow algorithms, the algorithm observes the restriction on the balance of the incoming flow and the outgoing flow into each internal node (other than source and sink nodes). In addition, this model has computational complexity $O(m)$, lower than $O(n^3)$ in Physarum Solver [16]. In terms of space complexity, it takes $O(m)$ in our model, also lower than $O(n^2)$ in Physarum Solver [16]. We will do research on the time complexity of the iterations in the near future.

As the numerical results illustrate, the proposed algorithm obtains the maximum flow in a continuous manner, which is quite consistent with actual situation. When working with dynamically changing environment, i.e., the cost of transferring goods may change dynamically because of weather or other unexpected factors. The continuous process allow the model to quickly adapt to external variation and recompute the experimental results. In a word, the algorithm is flexible and quite effective.

When comparing with the other two traditional methods for maximum flow problem, the runtime is quite longer. That’s because Physarum solver is modeled based on the solving a systems of linear equations on every iteration. It takes definitely more time to solve. But as we mention from beginning, the adaptivity of the proposed algorithm has never been shown or developed to solve maximum flow problem. Maybe parallel computing would help accelerate the algorithm.

**ACKNOWLEDGMENT**

We thank Dr. Qing Wang at Southwest University and PHD Cai Gao at University at Buffalo for helpful conversations. The work is partially supported by National Natural Science Foundation of China (No. 61364030 and 61763009),
the Scientific Research Foundation of Educational Commission of Hubei Province of China (No. D20151902), the Training Programs of Innovation and Entrepreneurship for Undergraduates of Hubei Province (No. 201410517018), The Training Programs of innovation and entrepreneurship for undergraduates of Hubei University for Nationalities (No. 2016CX033).

REFERENCES


