Fast Shortest Path Optimization Inspired by Shuttle Streaming of Physarum Polycephalum

Jayantha Siriwardana  
Department of Mechanical Engineering  
School of Engineering  
The University of Melbourne  
Parkville, Victoria, Australia  
Email: jsat@pgrad.unimelb.edu.au

Saman K. Halgamuge  
Department of Mechanical Engineering  
School of Engineering  
The University of Melbourne  
Parkville, Victoria, Australia  
Email: saman@unimelb.edu.au

Abstract—The plasmodium of the slime mold Physarum polycephalum, a large amoeboid organism, displays remarkable intelligent behaviors such as solving mazes, shuttle streaming and event anticipation. These amoeboid behaviors are results of the dynamics of the viscoelastic protoplasm and its biochemical rhythms. Having inspired by the intelligence shown by this primitive organism without a nerve system to solve mazes, we proposed mathematical models to mimic the intelligent foraging behavior that can be used to find the shortest path between two points of a graph. In result, we found that the convergence of the proposed two versions, Physarum Optimization with Shuttle Streaming (POSS) and POSS with mutation, are 40-11650 times faster when compared with the currently available Physarum Solver (PS) method and the results obtained are comparable.

I. INTRODUCTION

The true slime mold Physarum polycephalum is a large amoeboid organism that inhabits shady, cool and moist areas. The vegetative phase of the life cycle of Physarum polycephalum is the plasmodium, which consists of protoplasmic tube-like plasmodial veins and many nuclei. Usually during the search of food, plasmodium spreads out its network of tubes to fill the total available area that permits it to grow. It sometimes grows to a size of several square meters, while separated segments as small as 1 mm² can survive as individuals. On the other hand, two plasmodia can coalesce spontaneously to form one larger plasmodium when they encounter each other.

While being a unicellular organism, it has been reported that the plasmodium can display remarkable smart behavior: solving mazes [1], [2], [3], periodic event anticipation [4], maintaining an effective communication system within the cell [5] and moving towards nutrients and humidity [6]. These intriguing behaviors are considered as a result of information processing at intracellular level [7] although there is no widely accepted mechanism to explain it. The intelligent foraging behavior that gives rise to solving mazes has inspired researchers to suggest a new model to solve shortest path problems [8].

The intelligent behaviors of the plasmodium also inspired researchers to exploit and utilize them in achieving variety of non-classical computational schemes. Adamatzky demonstrated realization of Kolmogorov-Uspensky machines [9] and computation of spanning trees [10] on an appropriately cultured substrate of plasmodium. Tsuda, Aono and colleagues implemented Boolean gates that are fundamental to digital computing on the plasmodium to mark possibility of biological computing devices [11], [12], [13]. Aono and colleagues demonstrated a solution to the traveling salesman problem by inducing plasmodium’s photosensitive branches to grow or degenerate [14]. In another occasion, Tsuda and colleagues experimented controlling a hexapod robot using a biological control circuit made of plasmodium of Physarum polycephalum [15].

In this paper, we focus on the back and forth movement of protoplasmic flow in the tubular network of the plasmodium and propose a model that mimic the shortest path finding behavior. We also propose a modification to a previously proposed model called ‘Physarum Solver’ in order to achieve fast and sustained convergence. We incorporate all the variants of models to find the shortest path of graphs with various complexities and compare their performance with Ant Colony Optimization (ACO) [16], [17].

II. INTELLIGENT BEHAVIOR OF Physarum polycephalum

The plasmodium of Physarum polycephalum is essentially a unicellular organism with many nuclei. Hence, it does not possess a central information processing mechanism such as a brain to process information intelligently nor a sophisticated information communication system such as a nerve system to communicate information throughout its body structure. Therefore, it is indeed remarkable to witness aforementioned intelligent behavior. There has been numerous research work that attempt to answer the question: where does this intelligence originate?

In the plasmodium, streaming of the protoplasm through its tubular veins plays an important role in chemical signal generation. This protoplasmic streaming is known as shuttle streaming because the direction of flow changes back and forth periodically [7], [18]. Shuttle streaming is a result of hydrostatic pressure produced by the active rhythmic contraction throughout the cell [7]. The protoplasm stream flows through the tubular veins with speed circa 1 – 3 mm/sec [19] and changes the direction of flow every 1 – 3 minutes [9].

The rhythmic contractions occur synchronously throughout the organism also plays a key role in forming the cell shape.
Researchers have reported that the contraction pattern of the plasmodium is affected by the cell shape and conversely the cell shape can be affected by the contraction pattern [18]. It has also been observed that the rate of contraction can be changed artificially by changing the concentration of suitable attractants and repellents [20], [21]. These results collectively suggest that the intelligence shown by the plasmodium of Physarum polycephalum is closely related to the rhythmic contractions of the membrane of the cell and shuttle streaming of protoplasm through the plasmodal veins.

The model we propose to find the shortest path of a graph is inspired by shuttle streaming of the protoplasm caused by the rhythmic contractions. We associate the amount of nutrients carried through the tubular vein network by shuttle streaming with the optimization behavior plasmodium shows when two separated food sources presented.

III. RELATED WORK

A mathematical model named Physarum Solver (PS) was proposed by Tero et al. [8] to simulate the intelligent foraging behavior of the plasmodium of Physarum polycephalum. This model considers the tubular vein network as a network of liquid flow with two special nodes, i.e., $v_S$ and $v_F$ in Fig. 1. Node $v_S$ always acts as the source node while the node $v_F$ always acts as the sink node of the flux of protoplasmic flow. Assuming approximate Poiseuille flow, i.e. the steady flow of incompressible protoplasm parallel to the axis of circular tube produced by a pressure gradient along the tubular vein, the flux $Q_{ij}$ through the edge $(v_i, v_j)$ can be expressed as

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

and

\[
D_{ij} = \frac{\pi r_{ij}^4}{8\xi},
\]

where, $p_i$ is the pressure at node $v_i$, $L_{ij}$ is the length of the edge $(v_i, v_j)$, $D_{ij}$ is its conductivity, and $\xi$ and $r_{ij}$ are the viscosity coefficient and radius of the tube, respectively.

By considering the law of conservation of flux at each node, we have $\sum_j Q_{ij} = 0$, where $j \neq S, F$. For source node $v_S$ and sink node $v_F$, $\sum_i Q_{iS} + I_0$ and $\sum_i Q_{iF} - I_0$, where $I_0$ is the flux flowing into the source node and out of the sink node. In this model $I_0$ is considered as a constant.

Therefore, the network Poisson equation for the pressure, derived from the above equation becomes

\[
\sum_i \frac{D_{ij}}{L_{ij}}(p_i - p_j) = \begin{cases} -I_0 & \text{for } j = S, \\ I_0 & \text{for } j = F, \\ 0 & \text{otherwise.} \end{cases}
\]

By setting $p_F = 0$ as the basic pressure level at sink node $v_F$, one can solve Eq. (2) to obtain all $p_i$ values uniquely.

This model assumes the tube thickness changes dynamically according to the the flux and therefore the conductivity $D_{ij}$ changes in time according to the flux $|Q_{ij}|$,

\[
\frac{d}{dt}D_{ij} = \alpha |Q_{ij}| - \sigma D_{ij},
\]

Eq. (3) can be approximated as follows:

\[
D_{ij}(t + \delta t) = \alpha \delta t|Q_{ij}(t)| + (1 - \sigma \delta t)D_{ij}(t).
\]

This relation implies that the conductivity tends to diminish in the edges with low fluxes. The tube length $L_{ij}$ remains constant throughout the adaptation process, and changes in the conductance therefore result in changes in tube radius.

At the end of each time step, the new conductivity values of each edges are calculated and edges with negligible conductivity are considered as died out.

In the following section, we propose a new model to simulate the intelligent behavior of the plasmodium.

IV. PROPOSED MODEL FOR NETWORK OPTIMIZATION BY SHUTTLE STREAMING

A. Observations

When presented two or more food sources to a starved plasmodium of Physarum polycephalum, one can observe changes of its tubular vein network [3]. It has been observed two empirical rules are primarily responsible for this change in the network. First, open-ended tubular veins are likely to disappear gradually. Second, when there exist two or more veins to connect the same two food sources, the longer veins tend to disappear. Another important observation is that the external perturbations such as addition of nutrients can cause changes to the rhythmic contractions and shuttle streaming [18]. Therefore, we can reasonably assume that the geometry of the tubular vein network changes depending on the external stimulations.

B. Rationale behind the Algorithm

When presented with a food source, nutrients absorbed into the protoplasm start to flow through the tubular vein network. Due to the observations mentioned in the Section IV-A, we assume that the changes to the geometry of the tubular vein network after presenting with the food source is influenced by the chemical signals generated by the amount of nutrients introduced into it.

When two or more food sources are presented, the shuttle streaming of the protoplasm caused by the rhythmic contractions can play a vital role in changing the geometry of the tubular vein network. We utilize the strength of nutrient
concentrations carried by the protoplasmic flow towards each direction to reproduce the observations of changes to the tubular vein network when presented with food sources. As the result of local optimizations in the network, we achieve, as the global result, the shortest path between two food sources in the tubular vein network which is equivalent to an undirected graph.

C. The Proposed Algorithm

Let us assume that we require to find the shortest path from node \( v_S \) to node \( v_F \) in the undirected graph in Fig. 1. We assume this graph as a tubular vein network of plasmodium with two food sources placed at node \( v_S \) and node \( v_F \). In order to simulate the shuttle streaming of protoplasmic flow, we consider two different flows originate from node \( v_S \) and node \( v_F \) respectively with flow on one direction at a time. The time interval between the change of the direction of flow is \( \delta t \).

These flows bring important nutrients absorbed from food sources into the tubular vein network. The more the concentration of nutrients in the flow through a tubular vein, the more likely it can absorb nutrients into its walls and hence the more likely that it can survive. We use this concept to select the tubular veins that are most likely to survive and thereby let the shortest path between the two food sources emerge as the global result.

We identify the protoplasmic flow from node \( v_S \) to node \( v_F \) as forward flow and flow from node \( v_F \) to node \( v_S \) as backward flow.

D. Forward flow

Let us consider the protoplasmic flow in the direction of \( v_S \) to \( v_F \) at the node \( v_j \) (see Fig. 1) at time \( T = t \). The amount of nutrients received, \( R_f((v_j, v_j), t) \), from node \( v_j \) through the edge \((v_i, v_j)\) during the forward flow is given by

\[
R_f((v_i, v_j), t) = S_f((v_i, v_j), t) - \eta L_{ij}, \tag{5}
\]

where \( S_f((v_i, v_j), t) \) is the amount of nutrients sent through the edge \((v_i, v_j)\) at node \( v_i \), \( L_{ij} \) is the length of the edge \((v_i, v_j)\) and \( \eta \) is the absorption rate of nutrients into the tubular vein walls per unit length. Here, we assume the nutrient absorption rate \( \eta \) is uniform throughout the tubular network.

Next we consider the amount of nutrients sent out from the node \( v_f \) through the edges that go out from it. Here, we assume that each node has a memory of the amount of nutrients it received through each edge in concern in the previous backward flow. The node \( v_f \) distributes the total amount of nutrients it received in the current forward flow among the edges that go out from it in proportion to the amount it received in the previous backward flow. Therefore, the amount of nutrients sent out through the edge \((v_j, v_k)\) by the node \( v_j \) can be expressed as

\[
S_f((v_j, v_k), t) = \frac{\text{tot}R_f(v_f, T = 0)}{1} \sum_{\forall v_k \in (v_j, v_k)} R_f((v_k, v_j), t - \delta t), \tag{6}
\]

where,

\[
\text{tot}R_f(v_j, t) = \sum_{\forall v_i \in (v_j, v_k)} R_f((v_i, v_j), t),
\]

\[
\text{tot}R_b(v_j, t - \delta t) = \sum_{\forall v_k \in (v_j, v_k)} R_b((v_k, v_j), t - \delta t).
\]

Note that, \( \text{tot}R_f(v_j, t) \) is the total amount of nutrients received by node \( v_j \) in the current forward flow and \( \text{tot}R_b(v_j, t - \delta t) \) is the total amount of nutrients received by node \( v_j \) in the previous backward flow.

E. Backward flow

Now we consider backward flow nutrients in the direction of \( v_F \) to \( v_S \) at node \( v_q \) (see Fig. 1) at time \( T = t + \delta t \). We can express the amount of nutrients received, \( R_b((v_p, v_q), t + \delta t) \), from node \( v_p \) in the current backward flow similar to Eq. (5) as

\[
R_b((v_p, v_q), t + \delta t) = S_b((v_p, v_q), t + \delta t) - \eta L_{pq}, \tag{7}
\]

where \( S_b((v_p, v_q), t + \delta t) \) is the amount of nutrients sent through the edge \((v_p, v_q)\) at node \( v_p \).

We use the same rationale as in Eq. (6) to obtain the amount of nutrients sent, \( S_b((v_q, v_r), t + \delta t) \), through the edge \((v_q, v_r)\). The equation then becomes

\[
S_b((v_q, v_r), t + \delta t) = \frac{\text{tot}R_b(v_q, t + \delta t)}{\text{tot}R_f(v_q, t)}, \tag{8}
\]

where,

\[
\text{tot}R_b(v_q, t + \delta t) = \sum_{\forall v_p \in (v_q, v_q)} R_b((v_p, v_q), t + \delta t),
\]

\[
\text{tot}R_f(v_q, t) = \sum_{\forall v_r \in (v_q, v_q)} R_f((v_r, v_q), t).
\]

F. Start Node, End Node, Initialization and Termination

Start and end nodes (i.e. \( v_S \) and \( v_F \)), are special nodes where food sources located. Both \( v_S \) and \( v_F \) act as source of protoplasmic flow and as sink depending on the direction nutrient of flow. Here, we assume that the amount of nutrients absorbed into the tubular vein network per one forward flow session or backward flow session is \( N_{in} \) and it does not change over the time.

The amount of nutrients received and sent from nodes \( v_S \) and \( v_F \) both in forward and backward flows can be derived from Eqs. (5) – (8) by substituting \( N_{in} \) as necessary.

At \( T = 0 \) the amount of nutrients sent from node \( v_j \) is calculated as follows:

\[
S_f((v_j, v_k), T = 0) = \frac{\text{tot}R_f(v_j, T = 0)}{1} \sum_{\forall v_k \in (v_j, v_k)}, \tag{9}
\]

i.e. total amount of nutrients divided by the number of edges. When the nutrients carried by the edges reaches zero or below, we can discard them as disappeared since without nutrients...
Algorithm 1 Physarum Optimization with Shuttle Streaming (POSS)

1: procedure POSS($G(n, v), S, F$)
2:     // Find the shortest path from $S$ to $F$ of graph $G$ with
3:     // $n$ nodes and $v$ edges.
4:     $\eta :=$ Nutrient absorption rate
5:     $N_{in} :=$ Rate of nutrient input at $S$ and $F$
6:     $t := 0$
7:     $\delta t :=$ Time between change of flow direction
8:     $P :=$ Termination condition
9:     while $P$ NOT satisfied do
10:         // Nutrient flow $S \rightarrow F$ (Forward flow)
11:             for $j \leftarrow 1 : n$
12:                 $t \leftarrow t + 1$
13:                 for all edges $v_{ij} \in v$ come into node $j$ do
14:                     // Calculate amount of nutrients coming in
15:                     // from each edge
16:                     if $Re(v_{ij}, t) \leq 0$ then
17:                         $Re(v_{ij}, t) = 0$ // Discard edge $v_{ij}$
18:                     end if
19:                 end for
20:                 for all edges $v_{jk} \in v$ go out of node $j$ do
21:                     // Calculate amount of nutrients going out
22:                     // through each edge
23:                 end for
24:     end for
25:     // Nutrient flow $F \rightarrow S$ (Backward flow)
26:     for $q \leftarrow 1 : n$
27:         $t \leftarrow t + 1$
28:         for all edges $v_{pq} \in v$ come into node $q$ do
29:             // Calculate amount of nutrients coming in
30:             // from each edge
31:             if $Re(v_{pq}, t) \leq 0$ then
32:                 $Re(v_{pq}, t) = 0$ // Discard edge $v_{pq}$
33:             end if
34:         end for
35:         for all edges $v_{qr} \in v$ go out of node $q$ do
36:             // Calculate amount of nutrients going out
37:             // through each edge
38:         end for
39:     end for
40:     // Return the graph with shortest path
41:     return $G(n, v)$
42: end procedure

they cannot survive. In the subsequent protoplasmic flows we
do not consider these edges as candidates to direct the flow.

The amount of nutrients receiving at the sink node increases
with the progress of the optimization process. Therefore, the
termination condition can be defined as the situation where
the amount of nutrients at the sink node does not change with
time.

In the next section, we propose one modification each for
above POSS model and PS model to study the convergence
properties.

V. VARIANTS OF POSS AND PS

Nature inspired optimization algorithms such as Ant Colony
Optimization (ACO), Genetic Algorithms (GA), Particle
Swarm Optimization (PSO), etc. utilize stochastic sampling
to reduce the problem space and to avoid local minima in
multimodal problem spaces. However, despite the fact that PS
and the POSS are nature inspired, they do not employ this
technique for their advantage. In this section, we address this
concern by introducing stochastic control parameters to these
algorithms.

A. POSS with Mutation

In POSS method, we relied upon Eq. (6) in the forward flow
and Eq. (8) in the backward flow to calculate the propagation
of nutrients flowed through a given edge to the direction
of a given node. In these equations, more nutrient flow is
directed to the edges that have less probability to be absorbed.
However, in a natural system, there can be exemptions to this
rule in the form of mutations where abrupt changes happen
to the existing ongoing process. In our case, this can be
considered as a random distribution of nutrient propagation.

We introduce a new control parameter, probability of mu-
tation, which is a constant across the tubular vein network as
the parameter that control the occurrences of mutation. If the
node was selected mutation to happen, it would distribute the
amount of received nutrients randomly among the edges that
convey nutrients out.

It should be noted that the amount of nutrients came into
a node has to go out from the node. Therefore, at occasions
where mutation occur, the following conditions must be satis-
fied:

$$\sum_{\forall v_k \in (v_j, v_k)} Sc_{rand}((v_j, v_k), t) = totRe(v_j, t), \quad (10)$$

where $Sc_{rand}((v_j, v_k), t)$ is the amount of nutrients flow out
through the edge $(v_j, v_k)$.

The mutation can help fast convergence at times where there
is a competition between edges with similar lengths. However,
as with other nature inspired algorithms that employ mutation, the probability of mutation should be determined carefully with empirical trials.

B. PS with Time Varying Coefficients

Conductivity update equation of PS method, Eq. (3) or Eq. (4) shows that the larger $\alpha$ and $\sigma$ values emphasize on the current flux through an edge (tubular vein) over its previous value. On the other hand, smaller $\alpha$ and $\sigma$ values give more emphasis on the conductivity of the previous time step. We investigated the impact of the values of $\alpha$ and $\sigma$ for the convergence behavior.

Fig. 2 shows the labyrinth traditionally that was used to observe the shortest path finding by the plasmodium of true slime mold Physarum Polycephalum. We used this labyrinth to investigate the convergence behavior of PS method. In this investigation, we set the lengths of the edges in such a way that the shortest path becomes $\{(v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_6), (v_6, v_7), (v_7, v_20), (v_20, v_22)\}$.

Fig. 3 reveals that the larger the values of $\alpha$ and $\sigma$, the faster the convergence is. These parameters also determine the smoothness of convergence behavior where the smaller the parameters the smoother the convergence curve. For instance, Fig. 4 compares the change of conductance value of the edge with iterations for conditions $\alpha, \sigma = 18$ and $\alpha, \sigma = 1$ during the optimization process.

The oscillation evident at the beginning of optimization process when larger parameters used may not be acceptable in certain situations such as there is a limitation to the flow rate through the tube. Therefore, starting with smaller values for parameters $\alpha, \sigma$ and dynamically change them as the
optimization process progress can yield a situation where smooth yet fast convergence.

Here, we define a function to set parameters $\alpha$ and $\sigma$ dynamically as the optimization process progresses. The requirements are to suggest smaller parameter values at times where conductivity change is large (at the beginning of optimization), larger parameters values at times where conductivity change is small (towards the end of the optimization) and obtain a smooth convergence curve and fast convergence. We defined an exponential function as the method of changing these parameters dynamically as follows:

$$\alpha = \sigma = \alpha_{max}e^{-\beta \Delta D},$$

where, $\alpha_{max}$ is the maximum value that $\alpha$ takes (user defined), $\Delta D$ is the change in conductivity and $\beta$ is a scaling parameter which is also user defined. We employed this technique to change $\alpha$ and $\sigma$ dynamically during the optimization process. The resulting convergence behavior of conductance for the edge $(v_i, v_j)$ of the labyrinth in Fig. 2 is given in Fig. 5. It can be seen from the Fig. 5 that we can achieve a fast convergence as in the case of $\alpha = \sigma = 18$ (top graph of Fig. 4) and smooth convergence as in the case of $\alpha = \sigma = 1$ (bottom graph of Fig. 4) simultaneously. The bottom graph of Fig. 5 shows the dynamic change of $\alpha$ and $\sigma$ values during the convergence process.

VI. EXPERIMENTATION WITH DIFFERENT GRAPH COMPLEXITIES

We use PS, POSS and their derivations on graphs with different complexities in order to assess their performance as algorithms to find the shortest path of a graph. We also use ACO algorithm to compare the performance of the mathematical models as optimization algorithms. The performance assessment is conducted by applying each method to find the shortest path from starting node to end node of graphs with different complexities. Here, when the graph $\Gamma$ has a set of nodes $V(\Gamma) \subseteq \{v_1, \cdots , v_n\}$, $v_1$ becomes the starting node and $v_n$ becomes the ending node. We use the definition of linear complexity of graphs described in [22] in order to construct benchmark graphs.

A. Linear Complexity of a Graph

The linear complexity of a graph is measured as the linear complexity of any one of graph’s adjacency matrix. If $A$ is any matrix, then the linear complexity of $A$ is the minimum number of additions, subtractions, and scalar multiplications required to compute $AX$, where $X$ is an arbitrary column vector of appropriate size.

If $A$ is the adjacency matrix of the graph $\Gamma$ with $V(\Gamma) \subseteq \{v_1, \cdots , v_n\}$ is its nodes set, $A$ can be defined as:

$$A = (a_{ij}) \in \{0, 1\}^{n \times n}$$

where, $a_{ij} = 1$ if $v_i$ and $v_j$ are connected and is 0 otherwise. For every node $v_i \in \Gamma$, one can associate the indeterminant $x_j$ and the linear form

$$f_i = \sum_{j=1}^{n} a_{ij}x_j.$$  

Note that $f_i$ depends only on the neighbors of $v_i$. Therefore, the linear complexity of $f_i$ can be upper bounded as $L(f_i) \leq \text{deg}(v_i) - 1$. Thereby, one can obtain an upper bound for a connected graph $\Gamma$ as

$$L(\Gamma) \leq 2|E(\Gamma)| - |V(\Gamma)|,$$

where, $E(\Gamma)$ is the set of edges of $\Gamma$. The upper bound of the linear complexity is where the graph is complete. Therefore, according to Eq. (14) the linear complexity of a complete graph with $n$ nodes becomes $n(n - 2)$.

B. Methodology of Generating Graphs with Arbitrary Complexities

We consider 9 graphs with 100 nodes and with linear complexities 500, 1000, 2200, 3400, 4600, 5800, 7000, 8400 and 9800. Note that the linear complexity of a complete graph with 100 nodes is 9800. We denote these graphs as $\Gamma^k$, where, $L(\Gamma^k) = k$, and $k = \{500, 1000, 2200, 3400, 4600, 5800, 7000, 8400, 9800\}$. Each of these graphs has a set of nodes $V(\Gamma^k) \subseteq \{v_1^k, \cdots , v_{100}^k\}$ where $v_1^k$ is the starting node and $v_{100}^k$ is the ending node.

The number of edges connected to node $\text{deg}(v_i^k)$ where $i = 1, \cdots , 100$ was determined randomly for each graph in such a way that each graph fulfill the condition $L(\Gamma^k) = k$. Values from the uniform distribution on the set $[0, 70]$ were assigned as the length of each edge $(v_i^k, v_j^k)$ that connects nodes $v_i$ and $v_j$ of graph $\Gamma^k$. We generated three graphs for each linear complexity to have three sets of graphs to analyze the performance of each technique.

C. Performance Analysis

The Table I through Table III show the convergence properties of each algorithm when applied to find the shortest path of each graph set. Here, ACO was used to compare the performance since it has long been around as an efficient nature inspired sub-optimal optimization algorithm. The results show that PS with time varying coefficients converges to the same result as the original version of PS except for graphs $\Gamma^{8400}$, $\Gamma^{9800}$ in Table II and $\Gamma^{7000}$, $\Gamma^{8400}$, $\Gamma^{9000}$ in Table III. PS with time varying coefficients also converge to the solution 1 - 72 times faster except for graph $\Gamma^{8400}$ in Table II and graph $\Gamma^{8400}$ Table III where it takes more time than PS.

The convergence of the proposed POSS and POSS with mutation methods are the fastest among the algorithms considered. When compared with PS, POSS and POSS with mutation are between 40 - 11650 times faster and when compared with ACO, they are between 15 - 142 times faster. The solutions generated are within 3.5% of the best result generated by either ACO or PS at higher computation time cost. When there are two competing paths such as in the case of graph $\Gamma^{9000}$ in all tables, these algorithms manage to select a path with minimum effort compared to PS and PS with time varying coefficients.
where take large amount of time to select one path between the two candidates. The convergence time for ACO increases as a rule of thumb with linear complexity of graphs. It also exhibits the convergence to sub-optimal solutions behavior. Both POSS and POSS with mutation achieve similar results in terms of the convergence to sub-optimal solutions behavior. Both POSS and POSS with mutation exhibit similar results.

VII. CONCLUSIONS

In this paper, we proposed a mathematical model to simulate the intelligent foraging behavior of the plasmodium of slime mold Physarum Polycephalum using shuttle streaming. We also proposed modifications to both current mathematical model and proposed method to analyze the performance as optimization techniques to find shortest path of a given graph. We compared our results with ACO and found that all the models inspired by Physarum Polycephalum perform sub-optimally when applied as a shortest path finding algorithm. The convergence of the proposed POSS and POSS with mutation are 40 - 11650 times faster when compared with PS and 15 - 142 times faster when compared with ACO. The results obtained are within 3.5% of the best results generated by PS or ACO. We can conclude that the proposed POSS based methodologies should be the choice for on-line applications that requires acceptable solutions quickly.
In future work, we will focus on applying these models to solve computationally hard problems such as traveling salesman problem to analyze the performance. We also work on applying these techniques to on-line optimization tasks that arise in real-world engineering problems.

**REFERENCES**


