Fast Algorithms Inspired by Physarum Polycephalum for Node Weighted Steiner Tree Problem with Multiple Terminals

Yahui Sun
Department of Mechanical Engineering
University of Melbourne
Melbourne, Victoria 3010, Australia
Email: yahuis@student.unimelb.edu.au

Saman Halgamuge
Department of Mechanical Engineering
University of Melbourne
Melbourne, Victoria 3010, Australia
Email: saman@unimelb.edu.au

***Abstract*—Recently it has been shown that Physarum-inspired algorithms can solve some network optimization problems. However, it is not yet shown that Physarum-inspired algorithm can solve Node Weighted Steiner Tree Problem (NWSTP). Two new Physarum-inspired algorithms are proposed in this paper to solve NWSTP for the first time. Since all the existing NWSTP benchmark instances have an empty terminal set, new benchmark instances with non-empty terminal sets are generated to cover the shortage of existing benchmark instances. Both proposed algorithms are compared with Genetic Algorithm (GA) and Discrete Particle Swarm Optimization (DPSO) in these benchmark instances. Furthermore, an adapted Dijkstra's algorithm is proposed to provide the optimal solutions for part of these benchmark instances where there are two terminals and the node weights are negative. Simulation results show that our first proposed algorithm can find the optimal solutions for NWSTP with two terminals in graphs with negative node weights, and our second proposed algorithm can find close approximate solutions for NWSTP with multiple terminals in any node weighted graph. Both proposed algorithms provide faster and better NWSTP solutions than GA and DPSO.**

I. Introduction

Physarum polycephalum is a large amoeboid organism that inhabits shady, cool and moist areas [1], and it has displayed many remarkable intelligent behaviors, such as maze solving [2], periodic event anticipation [3] and efficient-network building [4]. This intelligence has been used to solve several network optimization problems. For example, Becker M used an agent based Physarum model to construct fault tolerant networks for the Tokyo rail system [5], and Zhang Z et al. used a Physarum-inspired algorithm to update the pheromone matrix in Ant Colony Optimization algorithm and then solved the traveling salesman problem [6].

Steiner tree problems are important network optimization problems which can be used in many areas, such as VLSI design [7], transportation system design [8] and system biology [9]. Therefore there are already some attempts to use Physarum-inspired algorithms to solve Steiner tree problems [10], [11]. However, all these researches are focusing on using Physarum-inspired algorithms to solve Steiner Tree Problem in Graphs (STPG) [10]–[12], while a more general version of Steiner tree problem, Node Weighted Steiner Tree Problem (NWSTP), has not been solved by Physarum-inspired algorithms so far.

The definition of NWSTP is given as follows. Let $G = (V, E, w, c)$ be a connected, undirected graph, where V is the set of vertices, E is the set of edges, w is a function which maps each vertex in V to a real number called the node weight, and c is a function which maps each edge in E to a positive number called the edge cost. Let $T \subseteq V$ be a (possibly empty) subset of V called terminals. The purpose of NWSTP is to find a connected subgraph $G' = (V', E'), V' \subseteq V, E' \subseteq E$ which connects all the terminals while minimizing the objective function $c(G') = \sum_{e \in E'} c(e) - \sum_{v \in V'} w(v)$, and the optimal solution is called Steiner Minimal Tree (SMT) in G for T. Other famous Steiner tree problems belonging to NWSTP are STPG and Prize-collecting Steiner Tree Problem (PCSTP). In STPG, the node weights are zero and there are at least two terminals, while in PCSTP, the node weights are nonnegative and there can be any number of terminals.

NWSTP was first proposed by Segev, A. in 1987 [13], and it arises in real-world telecommunication instances where access networks have to be built to provide services to customers. Positive node weights represent potential profits gained from each service point, negative node weights represent costs to set up each service point, and edge costs represent costs to connect different service points, either through wired or wireless technologies.

Many algorithms have been proposed to solve NWSTP in recent years. Nevertheless, most of these algorithms can only provide approximate solutions which may not be good enough [14]–[16], and other algorithms which promise to find the optimal solution cannot find it fast [17]. Moreover, there are still very few benchmark instances to test NWSTP algorithms so far, and all the existing node-weighted benchmark instances have an empty terminal set [18], regardless of the fact that many real-world NWSTP instances have non-empty terminal sets [19]. To cover these shortages, we propose two new Physarum-inspired algorithms to provide fast and good solutions for NWSTP with multiple terminals, and we also

978-1-5090-0623-6/16/$31.00

generate new NWSTP benchmark instances with non-empty terminal sets to test the proposed algorithms.

II. Related Works

A. Algorithms for Node Weighted Steiner Tree Problem

NWSTP is an important network optimization problem which has been studied intensively in recent years [8], [9]. The algorithms for NWSTP can be divided into 2 groups: exact algorithms and approximation algorithms. Exact algorithms can find SMT, but it takes a long time to do so in large-size graphs [17]. On the contrary, approximation algorithms can find solutions faster, but they may only find close approximations to SMT [16]. Due to the fact that large-size graphs widely exist in real-world NWSTP instances [9], the use of approximation algorithms may be necessary.

Many approximation algorithms have been proposed to solve NWSTP, such as GW algorithm (named by Michel X Goemans and David P Williamson) [14], Primal-dual algorithm [15] and other algorithms which can find close approximate solutions [16]. However, most of these state-of-the-art algorithms are specially designed to solve STPG or PCSTP, and they cannot be used in all the NWSTP instances. For example, GW algorithm is designed to solve PCSTP and it cannot be used in instances with negative node weights. Genetic Algorithm (GA) [20] and Discrete Particle Swarm Optimization (DPSO) [21] are approximation algorithms which can be used in all the NWSTP instances. Moreover, GA and DPSO have shown the ability to find SMTs in many benchmark instances in a reasonable time [20], [21]. Thus it is recommended to use GA and DPSO to solve NWSTP in some cases. Therefore we compare our proposed algorithms with GA and DPSO in this paper.

There are two ways to solve NWSTP using GA and DPSO. One way is to use GA and DPSO to select Steiner Vertices, which are non-terminal vertices in SMT. The other way is to use GA and DPSO to select the predecessor of each vertex. Kapsalis A et al. [20] first used GA to select Steiner Vertices. After the selection, SMT can be found easily by finding the Minimal Spanning Tree of the subgraph constructed by Steiner Vertices and terminals. In their simulations, GA has successfully found SMTs in many benchmark instances. Different from the method of selecting Steiner Vertices, Qu R et al. [21] have used DPSO to select the predecessor of each vertex, and the solution tree can be represented by the predecessor array. However, the method of selecting predecessors does not show a good performance in their simulation. Thus we use both GA and DPSO to solve NWSTP by selecting Steiner Vertices.

B. Physarum-inspired Algorithms for Seiner Tree Problems

Physarum-inspired algorithms are nature-inspired algorithms which can solve Steiner tree problems. In 2006, Tero A et al. [10] proposed a Physarum-inspired algorithm, Physarum Solver, to solve the Shortest Path Problem, which can be seen as STPG with only two terminals. Physarum Solver treats the graph as the tubular structure of the plasmodium of Physarum polycephalum, and it finds the shortest path by imitating Physarum's behavior to connect two food points. Beyond the Shortest Path Problem, Tero A et al. [12] further adapted Physarum Solver in 2008 to connect more than two terminals, and they have found close approximations to SMT in graphs without node weight. Recently, Liu L et al. [11] developed a new Physarum-inspired algorithm, Physarum Optimization (PO), to solve STPG. PO has low complexity and high parallelism, and it has a similar performance with several other algorithms in graphs with equal edge length. Therefore Physarum-inspired algorithms have already shown the ability to solve STPG.

STPG can be seen as a special case of NWSTP where all node weights are zero. Therefore NWSTP is a more general version of Steiner tree problem than STPG [9]. Nevertheless, no attempts have been made to use Physarum-inspired algorithms to solve NWSTP so far. Notably, no existing Physarum-inspired algorithm can even be applied to node weighted graphs. To cover this shortage, we propose two new Physarum-inspired algorithms in this paper and show their ability to provide faster and better NWSTP solutions than other algorithms.

III. Proposed Algorithms

A. Lowest-cost Path Physarum Optimization

In a special case of NWSTP where there are only two terminals in the graph, SMT is the lowest-cost path between two terminals. We define the Lowest-cost Path Problem (LPP) as a problem of finding a path $P = (V', E'), V' \subseteq V, E' \subseteq E$ to connect two terminals while minimizing the objective function $c(P) = \sum_{e \in E'} c(e) - \sum_{v \in V'} w(v)$. LPP can be seen as NWSTP with two terminals.

To our knowledge, there is no algorithm specially designed to solve LPP so far, and the lowest-cost path in node weighted graphs can only be found using NWSTP algorithms. However, NWSTP algorithms may not be able to solve LPP fast. In this section, a Physarum-inspired algorithm Lowest-cost Path Physarum Optimization (LPPO) is proposed to solve LPP fast, and it is the basis of the second proposed algorithm for NWSTP with multiple terminals.

The proposed algorithm LPPO is adapted from Physarum Solver [10]. Same as Physarum Solver, the two terminals in LPPO are also called the source node and the sink node. But the difference between LPPO and Physarum Solver is that LPPO is used in node weighted graphs, while Physarum Solver can only be used in graphs without node weight.

In LPPO, the graph is also considered as the tubular structure of the plasmodium of Physarum polycephalum. There is protoplasmic flow in every edge, and each edge has a conductivity value. In Physarum Solver, the flux Q_{ij} in edge (v_i, v_j) is given by Equation (1) and (2).

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

$$D_{ij} = \frac{\pi r_{ij}^4}{8\xi} \tag{2}$$

TABLE I
LOWEST-COST PATH PHYSARUM OPTIMIZATION ALGORITHM

Algorithm:	**Lowest-cost Path Physarum Optimization (LPPO)**
Input:	graph $G = (V, E, w, c)$, two terminals
Output:	path P, which is the lowest-cost path between two terminals
Initialize graph G, the edge conductivities, and the conductivity update time k	
while	$k \leq K$
	$k = k + 1$
	Calculate the pressure at each vertex using Equation (6)
	Calculate the flux through each edge using Equation (3)
	Update the conductivities using Equation (5)
	Cut edges with conductivity smaller than the threshold value ϵ
end	
The remaining network is path P	

where D_{ij} is the edge conductivity, c_{ij} is the edge cost, p_i and p_j are the pressures at vertex i and j, r_{ij} is the edge radius, and ξ is the viscosity coefficient. It can be seen that only the edge cost is considered in Equation (1), while the node weight is not. Therefore Physarum Solver cannot find the lowest-cost path in node weighted graphs.

A new flux equation is proposed in this paper to consider both the edge cost and the node weight, and it is described by Equation (3) and (4).

$$Q_{ij} = \frac{D_{ij}}{C_{ij}}(p_i - p_j) \tag{3}$$

$$C(i,j) = c_{ij} - \frac{w_i}{d_i} - \frac{w_j}{d_j} + 2M \tag{4}$$

where $C(i,j)$ is a composite cost for edge (v_i, v_j), w_i is the weight of node i, d_i is the degree of node i, and $M = max(w_x), x \subseteq V$. The node weight in Equation (4) can be both positive and negative.

After the calculation of flux, edge conductivities can be updated using the conductivity update equation below.

$$D_{ij}(k+1) = D_{ij}(k) + f(|Q_{ij}(k)|) - \mu D_{ij}(k) \tag{5}$$

where k is the number of conductivity update times, μ is a constant, and $f(|Q_{ij}(k)|)$ is an increasing function with $f(0) = 0$. In this paper, we consider $f(|Q_{ij}(k)|) = \alpha|Q_{ij}(k)|$, where α is a positive number. The conductivity update equation implies that conductivities tend to decrease in edges with small flux. By considering the conservation law of flux at each vertex, the network Poisson equation is described below.

$$\sum_{i \in V(j)} \frac{D_{ij}}{C_{ij}}(p_i - p_j) = \begin{cases} -I_0, & j = source \\ +I_0, & j = sink \\ 0, & otherwise \end{cases} \tag{6}$$

where $V(j)$ is the set of vertices linked to vertex j, and I_0 is the flux flowing into the source node and out of the sink node. Let the pressure at the sink node be 0, and other pressures can be calculated by solving the network Poisson equation. Then the new fluxes can be obtained and the conductivities will be updated again using the new fluxes. Edges with conductivities smaller than the threshold value ϵ will be cut from the network. In LPPO, K is used to signify the upper limit of k. After K times of conductivity update, the lowest-cost path between the source node and the sink node will be found. The steps of LPPO are described in Table I.

B. Lowest-cost Network Physarum Optimization

The major difference between LPP and NWSTP is that there may be more than two terminals in NWSTP. However, in the first proposed algorithm LPPO, there are only one source node and one sink node in the model. Therefore the biggest challenge of using Physarum-inspired algorithms to solve NWSTP is how to choose multiple terminals to be source node or sink node. The Lowest-cost Network Physarum Optimization algorithm (LNPO) is proposed in this section to solve NWSTP, and there is one sink node and many source nodes in it.

Let $l(i)$ be the total cost of edges linked to terminal i. Name the terminals in such a way that $l(1) \leq l(2) \leq \cdots \leq l(T)$, where T is the terminal number. Then the probability of choosing terminal i as the sink node can be obtained using Equation (7).

$$P(i) = \frac{l(T-i+1)}{\sum_{j=1}^{T} l(j)} \tag{7}$$

If terminal i is chosen to be the sink node, other terminals will become source nodes. With one sink node and $T-1$ source nodes, the network Poisson equation can be adapted as

$$\sum_{i \in V(j)} \frac{D_{ij}}{C_{ij}}(p_i - p_j) = \begin{cases} -I_0, & j = source \\ +(T-1)I_0, & j = sink \\ 0, & otherwise \end{cases} \tag{8}$$

Let the pressure at the sink node be 0, all the other pressures can be calculated using Equation (8). Ultimately, a NWSTP solution can be found by iterating the conductivity-update process in the same way as LPPO.

LNPO may find different NWSTP solutions for one instance because there is randomness in choosing the sink node. Therefore it is necessary to get as many solutions as possible and then choose the best one as the final solution. Nevertheless,

TABLE II
LOWEST-COST NETWORK PHYSARUM OPTIMIZATION ALGORITHM

Algorithm:	**Lowest-cost Network Physarum Optimization (LNPO)**
Input:	graph $G = (V, E, w, c)$, terminal set T
Output:	graph G', which is SMT or a close approximation to SMT in G for T
while	$n \leq N$ (the outer iteration process)
	$n = n + 1$
	Initialize graph G, the edge conductivities, and the conductivity update time k
	while $k \leq K$ (the inner iteration process)
	$k = k + 1$
	Choose the sink node and the source nodes using Equation (7)
	Calculate the pressure at each vertex using Equation (8)
	Calculate the flux through each edge using Equation (3)
	Update the conductivities using Equation (9)
	Cut edges with conductivity smaller than the threshold value ϵ
	end
	Calculate the total cost of the network found in the last inner iteration process
	Update the saved network
end	
The saved network is graph G'	

it is impossible to obtain more than one solution without retrieving the edges that have already been cut from the initial graph G. Thus the graph and edge conductivities need to be initialized after a certain number of conductivity update times. Same as LPPO, K is also used as the upper limit of k in LNPO. After K times of conductivity update, the graph and its edge conductivities will be initialized. Another parameter N is used as the upper limit of n, which is the number of initialization times. In LNPO, the iteration of conductivity update is called the inner iteration process, and the iteration of graph initialization is called the outer iteration process.

A new conductivity update equation is used in LNPO, and it is described as follows.

$$D_{ij}(k+1) = \alpha_{ij} * [D_{ij}(k) + f(|Q_{ij}(k)|) - \mu D_{ij}(k)] \quad (9)$$

where α_{ij} is a positive number attached to edge (v_i,v_j). LNPO can find different networks through different outer iteration processes, and the network with the smallest total cost is saved as the best network found so far. If edge (v_i,v_j) belongs to the saved network, α_{ij} will be set bigger than 1, which will make edge (v_i,v_j) have a bigger conductivity and then survive longer, or α_{ij} will be set smaller than 1, which will make edge (v_i,v_j) have a smaller conductivity and then survive shorter. The new parameter α_{ij} accelerates the optimization process of LNPO. The steps of LNPO are described in Table II.

IV. SIMULATIONS

A. LPPO Applied to Benchmark Instances

The first proposed algorithm LPPO is designed to solve LPP, which is NWSTP with two terminals. Other nature-inspired algorithms such as GA and DPSO can also provide good solutions for LPP. Therefore LPPO is compared with GA and DPSO in this section. Moreover, an Adapted Dijkstra's Algorithm (ADA) is proposed in this section to provide the optimal solutions for LPP in graphs with negative node weight. These ADA solutions are used to prove that LPPO can find the optimal solutions for LPP in all the benchmark instances with negative node weight.

Dijkstra's algorithm was proposed in 1959 to solve the Shortest Path Problem [22], and it cannot be applied to node weighted graphs directly. In the proposed algorithm ADA, the node weighted graph $G = (V, E, w, c)$ is first transformed to a graph without node weight $G'' = (V, E, c'')$. Assume the terminal set is T, the edge cost in G'' can be calculated using the equation below.

$$c''_{ij} = \begin{cases} c_{ij} - w_i/2 - w_j/2, & i, j \notin T \\ c_{ij} - w_i - w_j/2, & i \in T, j \notin T \\ c_{ij} - w_i - w_j, & i, j \in T \end{cases} \quad (10)$$

where c''_{ij} is the edge cost in G'', c_{ij} is the edge cost in G, and w_i is the node weight in G. It can be proved by Equation (11) that for any path $P(t_1, m_1, \ldots, m_k, t_2)$, the cost of P in G'' is always equal to the cost of P in G. Therefore the Lowest-cost Path Problem in G is equal to the Shortest Path Problem in G''.

$$\begin{aligned} C_{P\,in\,G''} = \; & (c_{t_1 m_1} - w_{t_1} - w_{m_1}/2) + \\ & (c_{m_1 m_2} - w_{m_1}/2 - w_{m_2}/2) + \ldots + \\ & (c_{m_k t_2} - w_{m_k}/2 - w_{t_2}) \\ = \; & (c_{t_1 m_1} + c_{m_1 m_2} + \ldots + c_{m_k t_2}) - \\ & (w_{t_1} + w_{m_1} + \ldots + w_{m_k} + w_{t_2}) \\ = \; & C_{P\,in\,G} \end{aligned} \quad (11)$$

After the transformation from G to G'', Dijkstra's algorithm can be used to find the shortest path in G'', which is the lowest-cost path in G. As Dijkstra's algorithm can only be used in graphs with positive edge costs, ADA can only find the lowest-cost path in graphs with negative node weight, which ensures the new edge cost c''_{ij} is positive. Nevertheless,

TABLE III
COMPARISON OF LPPO, GA, DPSO AND ADA IN BENCHMARK INSTANCES

Instance	V	E	T	NW Range	EC Range	ADA Sol	LPPO Sol	GA Sol	DPSO Sol	Fitness Evaluation Times		
										LPPO	GA	DPSO
LP1	100	120	2	(1, 10)	(5, 50)	N/A	6	**-81**	-35	1000000	**491054**	1000000
LP2	100	120	2	(1, 10)	(10, 100)	N/A	122	**79**	472	1000000	**30923**	1000000
LP3	100	120	2	(1, 10)	(20, 200)	N/A	**29**	29	799	**3**	4000	1000000
LP4	100	300	2	(1, 10)	(5, 50)	N/A	-6	**-106**	-53	1000000	**26955**	1000000
LP5	100	300	2	(1, 10)	(10, 100)	N/A	54	**41**	196	1000000	**3030**	1000000
LP6	100	300	2	(1, 10)	(20, 200)	N/A	**97**	153	382	**86**	1000000	1000000
LP7	100	500	2	(1, 10)	(5, 50)	N/A	-16	**-126**	-73	1000000	**142439**	1000000
LP8	100	500	2	(1, 10)	(10, 100)	N/A	**1**	1	71	**5**	112871	1000000
LP9	100	500	2	(1, 10)	(20, 200)	N/A	**98**	123	369	**43**	1000000	1000000
LP10	100	750	2	(1, 10)	(5, 50)	N/A	7	**-89**	-47	1000000	**93232**	1000000
LP11	100	750	2	(1, 10)	(10, 100)	N/A	40	**31**	128	1000000	**2220**	1000000
LP12	100	750	2	(1, 10)	(20, 200)	N/A	**43**	43	322	**24**	3511	1000000
LP13	100	1000	2	(1, 10)	(5, 50)	N/A	6	**-62**	-56	1000000	**122416**	1000000
LP14	100	1000	2	(1, 10)	(10, 100)	N/A	**28**	40	110	**205**	1000000	1000000
LP15	100	1000	2	(1, 10)	(20, 200)	N/A	107	**89**	314	1000000	**188969**	1000000
LP16	100	120	2	(-10, -1)	(5, 50)	90	**90**	121	804	**23**	1000000	1000000
LP17	100	120	2	(-10, -1)	(10, 100)	207	**207**	212	1261	**89**	1000000	1000000
LP18	100	120	2	(-10, -1)	(20, 200)	249	**249**	249	2248	**4**	10723	1000000
LP19	100	300	2	(-10, -1)	(5, 50)	42	**42**	42	177	**63**	2063	1000000
LP20	100	300	2	(-10, -1)	(10, 100)	95	**95**	147	418	**20**	1000000	1000000
LP21	100	300	2	(-10, -1)	(20, 200)	78	**78**	78	433	**28**	245180	1000000
LP22	100	500	2	(-10, -1)	(5, 50)	43	**43**	43	161	**61**	950	1000000
LP23	100	500	2	(-10, -1)	(10, 100)	85	**85**	90	328	**142**	1000000	1000000
LP24	100	500	2	(-10, -1)	(20, 200)	135	**135**	135	560	**13**	1664	1000000
LP25	100	750	2	(-10, -1)	(5, 50)	19	**19**	19	175	**4**	1835	1000000
LP26	100	750	2	(-10, -1)	(10, 100)	58	**58**	58	179	**7**	1024	1000000
LP27	100	750	2	(-10, -1)	(20, 200)	76	**76**	76	342	**12**	2937	1000000
LP28	100	1000	2	(-10, -1)	(5, 50)	61	**61**	61	158	**136**	30225	1000000
LP29	100	1000	2	(-10, -1)	(10, 100)	67	**67**	67	231	**19**	1279	1000000
LP30	100	1000	2	(-10, -1)	(20, 200)	129	**129**	129	380	**875**	907	1000000

the ADA solutions are always the optimal solutions of LPP. Therefore it is still meaningful to use ADA in the comparison to see whether LPPO, GA or DPSO has found the optimal solutions or not.

There is no existing benchmark instances for LPP so far. Thus new benchmark instances have been generated in this section. The generation method is the most popular graph generation method proposed by Aneja, Y. P in 1980 [23]. LPPO, GA, DPSO and ADA are compared in these new benchmark instances, and the comparison results are listed in Table III, in which V is the number of vertices, E is the number of edges, T is the number of terminals, *NW Range* is the range of node weights, *EC Range* is the range of edge costs, and *ADA Sol*, *LPPO Sol*, *GA Sol*, *DPSO Sol* are respectively the total costs of the best solutions found by ADA, LPPO, GA and DPSO in a given running time.

Besides the total cost of solutions, the running time of LPPO, GA and DPSO are also compared with each other. In the comparison, the running time is measured by the fitness evaluation times, which is the number of conductivity update times in LPPO, the number of crossover times in GA, or the number of particle position update times in DPSO. The upper limit of fitness evaluation times is 1 million. If an algorithm has not found the best solution found by all four algorithms, the recorded fitness evaluation times of this algorithm will be 1 million, or the fitness evaluation times needed by this algorithm to find the best solution will be recorded. In LPPO, GA and DPSO, an algorithm is the best algorithm when it needs a shorter running time to find the best solution than the other two algorithms, or when its solution has a smaller cost than the solutions of the other two algorithms. The solution and fitness evaluation time of the best algorithm in each instance have been highlighted in Table III. The parameter settings are: in LPPO, $I_0 = 0.5$, $\epsilon = 0.001$, $\alpha = 0.222$, $\mu = 1$; in GA, the population size is 10, the mutation probability of chromosomes is 0.3; in DPSO, the population size is 10, the neighborhood radius is 2, and the mutation probability of the best position is 0.2.

There are 30 benchmark instances in Table III (LP1-30). Node weights are positive in the first half of instances (LP1-

TABLE IV
COMPARISON OF LNPO, GA AND DPSO IN BENCHMARK INSTANCES

Instance	V	E	T	NW Range	EC Range	LNPO Sol	GA Sol	DPSO Sol	Fitness Evaluation Times		
									LNPO	GA	DPSO
NG1	100	120	5	(1, 10)	(5, 50)	8	**-32**	18	1000000	**22050**	1000000
NG2	100	120	10	(1, 10)	(5, 50)	-3	**-72**	-23	1000000	**16293**	1000000
NG3	100	120	5	(1, 10)	(10, 100)	**78**	78	450	**84**	2453	1000000
NG4	100	120	10	(1, 10)	(10, 100)	**137**	146	391	**177905**	1000000	1000000
NG5	100	120	5	(1, 10)	(20, 200)	**353**	353	1271	**360**	2787	1000000
NG6	100	120	10	(1, 10)	(20, 200)	**595**	739	1394	**2380**	1000000	1000000
NG7	100	500	5	(1, 10)	(5, 50)	-11	**-103**	-71	1000000	**46884**	1000000
NG8	100	500	10	(1, 10)	(5, 50)	1	**-80**	-52	1000000	**58201**	1000000
NG9	100	500	5	(1, 10)	(10, 100)	**86**	91	144	**225**	1000000	1000000
NG10	100	500	10	(1, 10)	(10, 100)	**114**	119	151	**171490**	1000000	1000000
NG11	100	500	5	(1, 10)	(20, 200)	**192**	211	409	**216**	1000000	1000000
NG12	100	500	10	(1, 10)	(20, 200)	**453**	457	530	**267731**	1000000	1000000
NG13	100	1000	5	(1, 10)	(5, 50)	6	**-92**	-59	1000000	**2563**	1000000
NG14	100	1000	10	(1, 10)	(5, 50)	-17	**-90**	-72	1000000	**1711**	1000000
NG15	100	1000	5	(1, 10)	(10, 100)	**56**	56	117	**26983**	362387	1000000
NG16	100	1000	10	(1, 10)	(10, 100)	**128**	138	145	**53160**	1000000	1000000
NG17	100	1000	5	(1, 10)	(20, 200)	**174**	198	401	**23900**	1000000	1000000
NG18	100	1000	10	(1, 10)	(20, 200)	297	**297**	382	752896	**21857**	1000000
NG19	100	120	5	(-10, -1)	(5, 50)	**130**	130	775	**38**	1891	1000000
NG20	100	120	10	(-10, -1)	(5, 50)	**281**	281	815	**40**	1072	1000000
NG21	100	120	5	(-10, -1)	(10, 100)	**167**	167	1266	**8**	2876	1000000
NG22	100	120	10	(-10, -1)	(10, 100)	**371**	390	643	**2132**	1000000	1000000
NG23	100	120	5	(-10, -1)	(20, 200)	**419**	431	2187	**349569**	1000000	1000000
NG24	100	120	10	(-10, -1)	(20, 200)	**959**	1048	2177	**12529**	1000000	1000000
NG25	100	500	5	(-10, -1)	(5, 50)	**114**	114	177	**99**	116235	1000000
NG26	100	500	10	(-10, -1)	(5, 50)	**205**	206	255	**122**	1000000	1000000
NG27	100	500	5	(-10, -1)	(10, 100)	**198**	198	343	**22**	333508	1000000
NG28	100	500	10	(-10, -1)	(10, 100)	324	**324**	417	230332	**11527**	1000000
NG29	100	500	5	(-10, -1)	(20, 200)	**321**	382	633	**32**	1000000	1000000
NG30	100	500	10	(-10, -1)	(20, 200)	**583**	583	675	**4342**	115101	1000000
NG31	100	1000	5	(-10, -1)	(5, 50)	**74**	74	127	**2184**	5192	1000000
NG32	100	1000	10	(-10, -1)	(5, 50)	**180**	180	226	**61768**	349499	1000000
NG33	100	1000	5	(-10, -1)	(10, 100)	**164**	164	292	**56**	517	1000000
NG34	100	1000	10	(-10, -1)	(10, 100)	280	**280**	331	374473	**7557**	1000000
NG35	100	1000	5	(-10, -1)	(20, 200)	**214**	214	433	**810**	11317	1000000
NG36	100	1000	10	(-10, -1)	(20, 200)	**488**	488	610	**2970**	56709	1000000

15), and LPPO has a better performance than GA and DPSO in 6 of these instances (LP3, 6, 8, 9, 12, 14). On the contrary, node weights are negative in the second half of instances (LP16-30), and LPPO has a better performance than GA and DPSO in all of them. Moreover, it can be seen from the ADA solutions that LPPO can find the optimal solution in all the instances with negative node weights, while GA can only find the optimal solution in part of these instances (LP18, 19, 21, 22, 24-30), and DPSO cannot find the optimal solution in any instance. Furthermore, in all the instances where both LPPO and GA can find the best solution (LP3, 8, 12, 18, 19, 21, 22, 24-30), LPPO finds the solution much faster than GA. Thus it can be concluded that LPPO can find the optimal solutions for NWSTP with two terminals in graphs with negative node weights, and its performance is better than GA and DPSO.

B. LNPO Applied to Benchmark Instances

NWSTP has important applications in the real world [8], [9], but there are very few NWSTP benchmark instances at present [18]. Moreover, all the existing benchmark instances have an empty terminal set, even though many real-world NWSTP instances have non-empty terminal sets [19]. Therefore new benchmark instances with non-empty terminal sets are generated in this section to cover the shortage of existing benchmark instances. The generation method is also the method proposed by Aneja, Y. P in 1980 [23]. After the generation, the second proposed algorithm LNPO is compared with GA and DPSO in these benchmark instances. The comparison results are

listed in Table IV, in which the solution and fitness evaluation time of the best algorithm in each instance have also been highlighted. The parameter settings are: in LNPO, $I_0 = 1$, $\epsilon = 0.001$, $\alpha = 0.4$, $\mu = 1$, $\alpha_{ij} = 1.2/0.8$; in GA and DPSO, the settings are the same as the settings in the above section.

It can be seen from Table IV that there are 36 benchmark instances in total (NG1-36). In 18 instances with positive node weights (NG1-18), LNPO can find the best solution in 12 instances (NG3-6, 9-12, 15-18), while GA can only find the best solution in 10 instances (NG1-3, 5, 7, 8, 13-15, 18), and DPSO cannot find the best solution in any instance. In 18 instances with negative node weights (NG19-36), LNPO can find the best solution in all the instances, while GA can only find the best solution in 13 instances (NG19-21, 25, 27, 28, 30-36), and DPSO cannot find the best solution in any instance. Furthermore, in the 17 instances where both LNPO and GA can find the best solution (NG3, 5, 15, 18-21, 25, 27, 28, 30-36), LNPO finds the solution faster than GA in 14 instances (NG3, 5, 15, 19-21, 25, 27, 30-33, 35, 36). Hence it can be concluded that LNPO can provide faster and better NWSTP solutions than GA and DPSO.

V. Conclusion

Two new Physarum-inspired algorithms are proposed in this paper to solve NWSTP. The first proposed algorithm LPPO can solve NWSTP with two terminals. The second proposed algorithm LNPO can solve NWSTP with multiple terminals. Since all the existing NWSTP benchmark instances have an empty terminal set, new benchmark instances with non-empty terminal sets are generated to cover the shortage of existing benchmark instances. After the generation of benchmark instances, two proposed Physarum-inspired algorithms are compared with GA and DPSO in these benchmark instances. Moreover, an adapted Dijkstra's algorithm is proposed to find the optimal solutions for NWSTP with two terminals in graphs with negative node weights. Simulation results show that our first proposed algorithm LPPO can also find the optimal solutions for NWSTP with two terminals in graphs with negative node weights, and our second proposed algorithm LNPO can find close approximate solutions for NWSTP with multiple terminals in any node weighted graph. Both proposed algorithms provide faster and better NWSTP solutions than GA and DPSO. This paper proves for the first time that Physarum-inspired algorithms can solve NWSTP, and it is recommended to research further on Physarum-inspired algorithms in the future to solve Steiner tree problems more efficiently.

Acknowledgment

This work is partially supported by University of Melbourne PhD Scholarship and Australian Research Council Grant DP150103512. Moreover, the authors would like to thank Dulini Mendis from University of Melbourne for her valuable comments on this work.

References

[1] J. Siriwardana and S. K. Halgamuge, "Fast shortest path optimization inspired by shuttle streaming of physarum polycephalum," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2012.

[2] T. Nakagaki, H. Yamada, and Á. Tóth, "Intelligence: Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 407, pp. 470–470, 2000.

[3] T. Saigusa, A. Tero, T. Nakagaki, and Y. Kuramoto, "Amoebae anticipate periodic events," *Physical Review Letters*, vol. 100, p. 018101, Jan. 2008.

[4] T. Nakagaki, H. Yamada, and M. Hara, "Smart network solutions in an amoeboid organism," *Biophysical chemistry*, vol. 107, pp. 1–5, Jan. 2004.

[5] M. Becker, "Design of fault tolerant networks with agent-based simulation of physarum polycephalum," in *2011 IEEE Congress on Evolutionary Computation (CEC)*, pp. 285–291, 2011.

[6] Z. Zhang, C. Gao, Y. Liu, and T. Qian, "A universal optimization strategy for ant colony optimization algorithms based on the physarum-inspired mathematical model," *Bioinspiration & biomimetics*, vol. 9, no. 3, p. 036006, 2014.

[7] J. Dong, H. Zhu, M. Xie, and X. Zeng, "Graph steiner tree construction and its routing applications," in *2013 IEEE 10th International Conference on ASIC (ASICON)*, pp. 1–4, Oct. 2013.

[8] E. Álvarez-Miranda, A. Candia-Véjar, X.-j. Chen, X.-d. Hu, and B. Li, "Risk models for the prize collecting steiner tree problems with interval data," *Acta Mathematicae Applicatae Sinica, English Series*, vol. 30, no. 1, pp. 1–26, 2014.

[9] A. Sadeghi and H. Fröhlich, "Steiner tree methods for optimal sub-network identification: an empirical study," *BMC bioinformatics*, vol. 14, no. 1, p. 144, 2013.

[10] A. Tero, R. Kobayashi, and T. Nakagaki, "Physarum solver: a biologically inspired method of road-network navigation," *Physica A: Statistical Mechanics and its Applications*, vol. 363, pp. 115–119, Apr. 2006.

[11] L. Liu, Y. Song, H. Zhang, H. Ma, and A. V. Vasilakos, "Physarum optimization: A biology-inspired algorithm for the steiner tree problem in networks," *IEEE Transactions on Computers*, vol. 64, pp. 819–832, Mar. 2015.

[12] A. Tero, K. Yumiki, R. Kobayashi, T. Saigusa, and T. Nakagaki, "Flow-network adaptation in physarum amoebae," *Theory in Biosciences*, vol. 127, pp. 89–94, Apr. 2008.

[13] A. Segev, "The node-weighted steiner tree problem," *Networks*, vol. 17, no. 1, pp. 1–17, 1987.

[14] M. X. Goemans and D. P. Williamson, "A general approximation technique for constrained forest problems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 296–317, 1995.

[15] P. Feofiloff, C. G. Fernandes, C. E. Ferreira, and J. C. de Pina, "Primal-dual approximation algorithms for the prize-collecting steiner tree problem," *Information Processing Letters*, vol. 103, no. 5, pp. 195–202, 2007.

[16] S. Gutner, "Elementary approximation algorithms for prize collecting steiner tree problems," *Combinatorial Optimization and Applications*, vol. 5165, pp. 246–254, 2008.

[17] I. Ljubic, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti, "Solving the prize-collecting steiner tree problem to optimality," *ALENEX/ANALCO*, pp. 68–76, 2005.

[18] A. S. da Cunha, A. Lucena, N. Maculan, and M. G. Resende, "A relax-and-cut algorithm for the prize-collecting steiner problem in graphs," *Discrete Applied Mathematics*, vol. 157, no. 6, pp. 1198–1217, 2009.

[19] N. Tuncbag, A. Braunstein, A. Pagnani, S.-S. C. Huang, J. Chayes, C. Borgs, R. Zecchina, and E. Fraenkel, "Simultaneous reconstruction of multiple signaling pathways via the prize-collecting steiner forest problem," *Journal of Computational Biology*, vol. 20, no. 2, pp. 124–136, 2013.

[20] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith, "Solving the graphical steiner tree problem using genetic algorithms," *Journal of the Operational Research Society*, vol. 44, pp. 397–406, Apr. 1993.

[21] R. Qu, Y. Xu, J. P. Castro, and D. Landa-Silva, "Particle swarm optimization for the steiner tree in graph and delay-constrained multicast routing problems," *Journal of Heuristics*, vol. 19, pp. 317–342, Apr. 2013.

[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[23] Y. P. Aneja, "An integer linear programming approach to the steiner problem in graphs," *Networks*, vol. 10, no. 2, pp. 167–178, 1980.