Research Article

A Bio-Inspired Method for the Constrained Shortest Path Problem

Hongping Wang, 1 Xi Lu, 1 Xiaoge Zhang, 1 Qing Wang, 1 and Yong Deng 1,2

1 School of Computer and Information Science, Southwest University, Chongqing 400715, China
2 School of Engineering, Vanderbilt University, Nashville, TN 37235, USA

Correspondence should be addressed to Yong Deng; ydeng@swu.edu.cn

Received 2 March 2014; Revised 13 April 2014; Accepted 23 April 2014; Published 14 May 2014

Academic Editor: Wen-Chiung Lee

Copyright © 2014 Hongping Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The constrained shortest path (CSP) problem has been widely used in transportation optimization, crew scheduling, network routing and so on. It is an open issue since it is a NP-hard problem. In this paper, we propose an innovative method which is based on the internal mechanism of the adaptive amoeba algorithm. The proposed method is divided into two parts. In the first part, we employ the original amoeba algorithm to solve the shortest path problem in directed networks. In the second part, we combine the Physarum algorithm with a bio-inspired rule to deal with the CSP. Finally, by comparing the results with other method using an examples in DCLC problem, we demonstrate the accuracy of the proposed method.

1. Introduction

The shortest path problem (SPP) is one of the most classic problems, which is widely used in many fields, like network optimization [1, 2], road navigation [3, 4], and so on [5–7]. The constrained shortest path (CSP) problem aims at finding the shortest path from a predetermined source node to a predetermined destination node in a directed network under the constraint of being less than or equal to a given upper limit. Such constraints on the classical shortest path problem generally result in the problem becoming NP-hard. CSP is often considered as a variant of SPP and is frequently put into practice, like tail assignment problem in aircraft scheduling [8–10], the quality of service routing in communications networks [11, 12], and so forth [13].

Strategies of solving the CSP can be classified into three main categories: Lagrangian relaxation methods, dynamic programming (DP), and path ranking methods. The first method of CSP relies on the solution to a Lagrangian dual problem and uses different approaches to close the duality gap. Handler and Zang [14] proposed a method to solve the Lagrangian dual problem. They closed the duality gap by using the k shortest paths algorithm, terminating with the first path satisfying the constraint. They also proposed a method which searched in a direction. The method was based on combination of the original objective and the constraint. Carlyle et al. [15] presented a new algorithm for CSP. The improvement of this method is that the generated Lagrangian function was optimized and the optimality gap was closed by enumerating near-shortest paths. Recently, Zhang et al. [16] used Lagrangian relaxation combined with an adaptive amoeba algorithm to solve the CSP. In fact, if the constraint was relaxed, it made the CSP as the classical SPP, which could be solved repeatedly by plentiful computational efforts. What is more, the efficiency of this approach relied on the effectiveness of the underlying unconstrained shortest path algorithms.

A number of previous works were based on DP. All of these methods were based on label-setting or label-correcting algorithm. Starting from the the work of Joksch [17], the approaches of node labeling were deeply investigated. Dumitrescu and Boland [18] extended this idea and solved this problem by using preprocessing techniques. They [18] showed how the performance of the label-setting method could be further improved by making use of all Lagrange multiplier information collected in a Lagrangian relaxation’s first step. Recently, Likhyani and Bedathur [19] developed SkIt index structure, which supported a wide range of label
constraints on paths, and returned an estimation of the shortest path that satisfied the constraints. However, all of these methods mentioned above had a disadvantage: they might fail to well-estimate very large networks due to the "curse of dimensionality." As a result, it could not apply to real world with a large size of nodes.

A path ranking method was presented by Handler and Zang [14] by k shortest paths algorithm. Santos et al. [20] extended this idea by improving the way of sorting the k shortest paths. Its direction was based on the relative tightness of the constraint. Jia and Varaiya [21] presented two new methods based on the k-shortest-path approach. These heuristic methods, where one is centralized and the other is distributed, were both polynomial. In numerical experiments the proposed algorithms almost found the optimal paths. The main drawback of this method is that large value for k is required before a feasible solution of CSP is found.

Recently, a slime mold called Physarum polycephalum has attracted many investigators [22–25]. The body of the plasmodium of Physarum polycephalum contained a network of tubular elements by means of which nutrients and chemical signals circulated through the organism in an effective manner [26, 27]. Since the tubes disassembled and reassembled within a period of a few hours in response to external conditions, this organism was very useful for studying the function and dynamics of natural adaptive networks [28–30]. It was used to find the shortest path [31–33], solve the amaze problem [34, 35], optimize the networks [36, 37], and structure the road networks [38–40] and other fields [41–45]. Due to the uncertainty in real application [46–53], this algorithm is also applied to solve shortest path under uncertain environment [54, 55].

Inspired by the intelligence of the amoeba, we combine the Physarum algorithm with a bioinspired rule to deal with the CSP. The main idea behind the approach is to employ Physarum algorithm to find the shortest path. Then a check procedure is processed to ensure it satisfies the constraint. Once it does not meet the constraint, a penalty function is applied to solve shortest path under uncertain environment [54, 55].

The paper is organized as follows. In Section 2, the basic theories, including CSP and the Physarum model, are introduced. In Section 3, the bioinspired method which is based on amoeba algorithm for CSP is proposed. In Section 4, an example is used to illustrate the proposed method. In comparison with other methods, the validity of the proposed method is demonstrated. In Section 5, the conclusion is given.

2. Basic Theories

In this section, we will introduce some basic theories, including the constrained shortest path problem and Physarum polycephalum model.

2.1. The Constrained Shortest Path Problem. Let $G = (V, E)$ be a directed network, where $V = 1, \ldots, n$ is the set of nodes and $E = (i, j) : i, j \in N, i \neq j$, is the set of directed edges. Each edge has two nonnegative weights $c_{ij}$ and $t_{ij}$, $c_{ij}$ and $t_{ij}$ represent the generalized cost and the constrained variable, respectively. The CSP can be expressed by the following mathematical formulation:

$$\min \sum_{ij} c_{ij} x_{ij},$$

subject to

$$\sum_{ij} x_{ij} - \sum_{ji} x_{ji} = \begin{cases} 1, & \text{for } j = 1, \\ -1, & \text{for } j = 2, \\ 0, & \text{for otherwise}, \end{cases}$$

$$\sum_{ij} t_{ij} x_{ij} \leq T,$$

$$x_{ij} \in \{0, 1\}, \forall i, j.$$  

where $x_{ij}$ is binary variable, which is defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if } x_{ij} \text{ is in optimal path}, \\ 0, & \text{others}. \end{cases}$$

The parameter $T$ indicates the maximum value allowed for the sum of $t_{ij}$. Equations (1), (2), and (4) define the shortest problem. The constraint (3) causes the CSP to be an NP-hard problem.

2.2. Physarum polycephalum Model. Physarum polycephalum is a single-celled amoeboid organism that can form a dynamic tubular network to link the discovered food sources during foraging. The physiological mechanism behind the phenomenon is that tubes thicken in a given direction when the flux through it persists in that direction for a certain time. By this mechanism, a mathematical model for path finding has been constructed to describe the adaptive dynamics of tubular network. In what follows, we will give a brief introduction to this path finding model [56].

Suppose the shape of the network formed by the Physarum is represented by a graph; the edge and the junction in the graph can be treated as a plasmodial tube and the node, respectively. Two special nodes labeled as $N_1$ and $N_2$ act as the source node and sink node, respectively. The other nodes are labeled as $N_3, N_4, N_5, N_6$, and so forth. The edge between nodes $N_i$ and $N_j$ is expressed as $M_{ij}$. The parameter $Q_{ij}$ denotes the flux through tube $M_{ij}$ from node $N_i$ to $N_j$. Assuming the flow along the tube is an approximate Poiseuille flow, the flux $Q_{ij}$ can be expressed as

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

where $p_i$ is the pressure at the node $N_i$, $D_{ij}$ is the conductivity of the tube $M_{ij}$, and $L_{ij}$ is its length.

By considering that the inflow and outflow must be balanced, we have

$$\sum Q_{ij} = 0 \quad (j \neq 1, 2).$$
As for the source node $N_1$ and the sink node $N_2$ the following two equations hold:

$$\sum_i Q_{i1} + I_0 = 0,$$

$$\sum_i Q_{i2} - I_0 = 0,$$

where $I_0$ is the flux flowing from the source node, which is set as a constant in the model.

Then the network Poisson equation for the pressure can be derived from (6)–(8) as follows:

$$\sum_i D_{ij} (p_i - p_j) = \begin{cases} -1, & \text{for } j = 1, \\ +1, & \text{for } j = 2, \\ 0, & \text{otherwise.} \end{cases}$$

By setting $p_2 = 0$ as a basic pressure level, all $p_i$ can be determined by solving (9) and $Q_{ij}$ can also be obtained according to (6).

With the mechanism between the flux and tube thickness (described as the conductivity of the tube), the adaptation equation for conductivity is constructed as follows:

$$\frac{d}{dt} D_{ij} = f \left( |Q_{ij}| \right) - r D_{ij},$$

where $r$ is a decay rate of the tube. It can be obtained from the equation that the conductivity will finally vanish if there is no flux along the edge, while it is enhanced by the flux. The $f$ is monotonically increasing continuous function satisfying $f(0) = 0$.

### 3. Proposed Method

The original amoeba algorithm can only find the shortest path in undirected networks. In order to solve the CSP in directed networks, we have to solve two problems. The first one is how to find the shortest path in directed networks. The second one is how to efficiently solve CSP by modified amoeba algorithm.

#### 3.1. Modified Amoeba Model for the Shortest Path Problem in Directed Networks

Let $G = (N, E, W)$ be a directed network, where $N$ represents the set of $n$ nodes, $E$ represents the set of directed edges, and $W$ represents the weight set of $E$. We suppose that there is only one direction between node $i$ and node $j$ in the network $G$. Assume nodes and node $t$ are source node and sink node, respectively. The shortest path problem in the network can be defined as how to find a path from node $s$ to node $t$, which only consists of directed edges of $E$, with the minimum sum of weights on the edges.

In the original amoeba model, every arc is bidirectional. It means that the flux can flow from node $i$ to node $j$. They can also flow in the opposite direction. It is no different for original amoeba model, while the direction should be strictly distinguished in the directed networks. In particular, there is only one direction between two nodes in network $G$. It means that if the flux can flow from node $i$ to node $j$, they cannot flow in the opposite direction. In order to take the factor of direction into account, each edge is regarded as two tubes with opposite direction and equal weight. Here, we assume the weight represents the length between two nodes represented by $L$. If there is a link between nodes $i$ and $j$, then there are two tubes with equal length $L_{ij}$ and $L_{ji}$, respectively, denoting two opposite directions from node $i$ to node $j$ and from node $j$ to node $i$. The initial value of conductivity along with each tube is used to imply whether the tube is accessible or not. If arbitrary tube satisfies $L_{ij} \in E$, then the corresponding initial value of conductivity $D_{ij}$ is assigned an arbitrary value among $[0.5, 1]$. Besides, $D_{ji}$ is assigned as 0. The matrix $D$ implies not only the conductivity but also the direction of each edge. By this way, the factor of direction is considered. In order to make original amoeba model that can adapt the directed networks, (9) is improved as follows:

$$\sum_i \left( \frac{D_{ij}}{L_{ij}} - \frac{D_{ji}}{L_{ji}} \right) (p_i - p_j) = \begin{cases} -1, & \text{for } j = s, \\ +1, & \text{for } j = t, \\ 0, & \text{otherwise.} \end{cases}$$

With the purpose of keeping the validity of conductivity, the flux along with each tube needs to be adjusted. Equation (10) is modified as follows:

$$\frac{D_{ij}^{n+1} - D_{ij}^n}{\delta t} = \begin{cases} Q_{ij} - D_{ij}^{n+1}, & Q_{ij} > 0, \\ -D_{ij}^{n+1}, & \text{otherwise.} \end{cases}$$

Amoeba model has a positive feedback mechanism: the higher the pressure difference is, the more the flux will be. The more the flux is, the higher the pressure difference will be. When the iteration continues, the shortest path gradually appears through this mechanism. In directed networks, the positive feedback mechanism also exists. The method for finding the shortest path in directed network is described as follows.

1. Initialize values of each parameter. According to the weight of the edge in $G$, the length of $L_{ij}$ is initialized. According to the direction of each edge, the conductivity is initialized. Here, the initial value is assigned equally as 0.5. If there is an edge from node $i$ to node $j$, $D_{ij} = 0.5$. Otherwise, $D_{ij} = 0$. The pressure of the node $i$ is represented by $p_i$. The basic information of this directed network is initialized.

2. According to current conductivity, the pressure of each node can be calculated using (11).

3. According to (6), we can obtain the flux of each edge.

4. According to (12), we can calculate the conductivity of the next moment.

5. If the conductivity $D_{ij}$ of each edge is no longer changed, the termination criterion is met. The directed shortest path consists of the edges with conductivity approximately equaling 1. Otherwise, go back to Step 2 and repeat until convergence.

#### 3.2. Bioinspired Method

In this section, we will introduce some nomenclatures and phenomena about amoeba. Then we will introduce the proposed method.
3.2.1. Nomenclatures and Phenomena

- **Segment**: the tubular pseudopodia of amoeba linking node \( i \) to node \( j \).
- **Growing segment**: the segments with continuous thickening.
- **Potential segment**: the growing segments with growing times exceeding given threshold.
- **Threshold**: the critical value of judging whether growing segment is potential segment.
- **Important segment**: the potential segments which can make up an unbroken path.
- **Important path**: the path which is made up by potential segments.

**Conductivity**: indicating the thickness of the segment.

As the iteration continues, some segments become thicker while others degenerate. Simultaneously, the corresponding conductivity tends to increase or decrease. When the conductivity value in the current iteration is greater than the previous one, it is called thickening segment. When a segment is thickening, it means that the segment will continue to thicken. The segment becomes a growing segment. If the growing times of the growing segment exceed the threshold, it becomes a potential segment. If the potential segment belongs to the important path, it becomes an important segment.

Through many trials, we find that the segment which belongs to the shortest paths continuously thickens from the beginning to the end. At the beginning, growing segments may be dispersed in the network. Then, potential segments and important segments occur. At the end, there are only important segments left, which make up a shortest path. In other words, we think some of those important segments are the final part of the shortest path and the important path is the shortest path on the great probability.

3.2.2. Proposed Method. As shown in Figure 1, an example employed in [16] is used to illustrate the general flow of the proposed method. Each arc has two attributes: the length \( L_{ij} \) and the cost \( C_{ij} \). The specific values can be obtained according to Table 1. For instance, the numbers (120 and 60) along the arc between node 1 and node 2 mean the length along the arc is 120, and the cost along the arc is 60. We want to find a length shortest path from source node 1 to sink node 20 with the cost no more than 200.

Inspired by the important segment, we propose a bioinspired method based on the internal mechanism of the amoeba model. Steps are introduced in detail as follows.

1. **On the basis of Step 1 in Section 3.1, we need to do some extra operations as follows.**

   According to the information of the given CSP, we initialize the constraint \( \theta \). \( \kappa \) is a parameter which is used to judge whether the growing segment is the potential segment or not. \( \gamma \) is the divisor of the punishment. When the important path does not satisfy the constraint conditions, \( \gamma \) is used to decrease the conductivity of those important segments so that the amoeba can converge to the path satisfying the constraint. Besides, we set a variable \( \text{flag} \) representing the end mark and assign 0, which means the end mark does not satisfy the termination criterion.

2. **Step 2 to Step 5 are the same as in Section 3.1.**

<table>
<thead>
<tr>
<th>Edge</th>
<th>Length</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 → 2</td>
<td>120</td>
<td>60</td>
</tr>
<tr>
<td>1 → 5</td>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>2 → 3</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>2 → 6</td>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>3 → 7</td>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>4 → 8</td>
<td>70</td>
<td>40</td>
</tr>
<tr>
<td>5 → 4</td>
<td>120</td>
<td>60</td>
</tr>
<tr>
<td>5 → 9</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>6 → 10</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>6 → 12</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>7 → 12</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>7 → 13</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>8 → 15</td>
<td>70</td>
<td>40</td>
</tr>
<tr>
<td>8 → 16</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>9 → 10</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>9 → 16</td>
<td>90</td>
<td>40</td>
</tr>
<tr>
<td>10 → 11</td>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>10 → 17</td>
<td>70</td>
<td>40</td>
</tr>
<tr>
<td>11 → 18</td>
<td>70</td>
<td>60</td>
</tr>
<tr>
<td>12 → 11</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>13 → 18</td>
<td>140</td>
<td>40</td>
</tr>
<tr>
<td>14 → 13</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>15 → 20</td>
<td>150</td>
<td>50</td>
</tr>
<tr>
<td>15 → 16</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>16 → 20</td>
<td>80</td>
<td>60</td>
</tr>
<tr>
<td>17 → 20</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>18 → 19</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>19 → 20</td>
<td>50</td>
<td>40</td>
</tr>
</tbody>
</table>
According to the current and the previous conductivity, the growing segments can be found and marked. The growing times of each growing segment are also saved.

According to the saved times and the threshold \( \kappa \), we can judge whether the growing segment exceeds the threshold. If it exceeds it, it is the potential segment. Save these potential segments.

Judge whether these potential segments can make up a complete path from the source node to the sink node. If they can, we mark these important segments and then go to Step 6. Otherwise, go to Step 7.

According to these important segments, the important path can be obtained. Judge whether the important path satisfies the constraint \( \theta \). If it can, the original data such as \( D_{ij}, Q_{ij} \) is kept and makes the flag equal 1, which means the flag satisfies the termination criterion. In other words, the constrained shortest path is found. Then go to Step 7. Otherwise, do the following. The conductivities of each important segment (the segment who belongs to this important path) are decreased. The updated conductivity is equal to the original value divided by the divisor of the punishment \( \gamma \). The marks including times and important segments are initialized.

Check whether the termination criterion is met or not. The termination criterion is that the value of flag is 1. If it is satisfied, tubes with maximal conductivity make up the constrained shortest path. Otherwise, go to Step 2 and repeat until termination is satisfied.

More detailed procedures are showed in Algorithm 1. The basic idea behind this method is that we take full advantage of *Physarum* algorithm to find the path. After finding the path successfully, the check procedure is conducted to insure the path satisfies the constraint. If it does not meet the constraint, the penalty mechanism will be processed. By this way, the successive path will fade out. The program will continue to execute until the path satisfying the constraint is found. As can be seen in Algorithm 1, it is necessary for us to solve the linear equations shown in (9). The time complexity of solving the linear equations is \( O(N^3) \), where \( N \) is the number of equations. In the original amoeba model, the number of equations and nodes in the network are equal. Hence, the time complexity of the bioinspired method is \( O(N^3) \), where the \( N \) is the number of nodes in the network. According to Figure 1, we let \( s \) be 1, \( t \) 20, and \( \theta \) 200. Parameters \( \kappa \) and \( \gamma \) are experimental values. Here, we let them be 3 and 10, respectively. As shown in Figure 2, the solution 1-5-9-16-20 to this problem is obtained easily. The length of the path is 340 and the cost of the path is 200.

It can be noted that the solution obtained by the proposed method is the same as the result presented by Zhang et al. [16]. Both of them can find the optimal solution. Therefore, the accuracy of the presented method is demonstrated. Different from the method proposed by Zhang et al. [16], the bioinspired method combines the *Physarum* algorithm and the penalty mechanism by modifying the internal mechanism of the amoeba model while Zhang et al. [16] combine the *Physarum* algorithm with the Lagrangian relaxation method.

### 4. Examples

In this section, an example is introduced in order to prove the validity of the proposed method. The results are compared with the algorithm presented by Handler and Zang [14].

It is well known that the delay constrained least cost problem (DCLC) is a typical problem of CSP, which is widely used in network routing [57, 58]. The DCLC problem is to find the least cost path in a graph while keeping the path delay below a specified value. Let \( G = (V, E) \) be a directed network, where \( V = 1, \ldots, n \) is the set of nodes and \( E = (i, j) : i, j \in n, i \neq j \) is the set of edges. The edge \((i, j)\) represents the direction from node \( i \) to node \( j \). Each edge has two nonnegative attributes: \( c_{ij} \) and \( d_{ij} \) represent cost and transmission delay from node \( i \) to node \( j \), respectively.

Given a source node \( s \in V \), a sink node \( t \in V \). \( \Delta_{\text{delay}} \) is the given delay constraint. Functions \( D(p) \) and \( C(p) \) represent delay and cost of the path \( p \). The DCLC problem can be formulated as follows:

\[
\min_{p \in P_s(t)} C(p). \quad (13)
\]
// L is the distance matrix, \( L_{ij} \) represents the edge length from node \( i \) to node \( j \).
// C is the cost matrix, \( C_{ij} \) represents the edge cost from node \( i \) to node \( j \).
// s is the source node, \( t \) is the sink node
// \( M : M_{ij} \) represents the increasing times for the segment with flux continuing to grow in the network (growing segment).
// \( U : U_{ij} \) represents the \( U_{ij} \) whose value is beyond the threshold (potential segment).
// \( \theta \) is the constraint, \( \kappa \) is the threshold, \( \gamma \) is the divisor of punishment

\[
D_{ij} \leftarrow \begin{cases} 
[0.5, 1] & (\forall i, j = 1, 2, \ldots, N \text{ and } L_{ij} \neq 0) \\
0 & (\forall i, j = 1, 2, \ldots, N) 
\end{cases} \\
p_i \leftarrow 0 (\forall i = 1, 2, \ldots, N) \\
Q_{ij} \leftarrow 0 (\forall i = 1, 2, \ldots, N)
\]

// flag = 0 
while \( flag \neq 0 \) do 
\[
p_t \leftarrow 0 \quad // \text{The pressure at the ending node } t
\]
Calculate the pressure of every node using (11)

\[
\sum_{j \in V} \left( \frac{D_{ij}}{L_{ij}} + \frac{D_{ji}}{L_{ji}} \right) (p_i - p_j) = \begin{cases} 
+1 & \text{for } i = s \\
-1 & \text{for } i = t \\
0 & \text{otherwise}
\end{cases}
\]

\[
Q_{ij} \leftarrow D_{ij} \times (p_i - p_j) / L_{ij} \quad // \text{Using (6)}
\]
if \( Q_{ij} < 0 \) then
\[
Q_{ij} = 0; // \text{Using (12)}
\]
\[
D_{ij}^{n+1} = D_{ij}^n / 2;
\]
else 
\[
D_{ij}^{n+1} = (D_{ij}^n + Q_{ij}) / 2;
\]
end if 
if \( D_{ij}^{n+1} - D_{ij}^n > \varepsilon (\varepsilon \rightarrow 0^+) \) then 
\[
M_{ij} = M_{ij} + 1
\]
else 
\[
M_{ij} = 0
\]
end if 
if \( M_{ij} > \kappa \) then 
\[
U_{ij} = 1
\]
else 
\[
U_{ij} = 0
\]
end if 
if judgePath(\( U, s, t \)) // To judge whether there is a path, the result is boolean then
\[
\text{path} = \text{findPath}(U, s, t) // \text{Find the path and return. The nodes (important segment) making up the important path are saved.}
\]
if calculateCost(path, \( C, s, t \)) \( \geq \theta \) // To judge whether the cost of the path is beyond the constraint then 
for \( i = 1 \) to (\( \text{length(path)} - 1 \)) do 
\[
M(\text{paths}(i), \text{paths}(i + 1)) = 0
\]
\[
M(\text{paths}(i + 1), \text{paths}(i)) = 0
\]
\[
D(\text{paths}(i), \text{paths}(i + 1)) = \max(D(\text{paths}(i), 3)) / \gamma
\]
\[
D(\text{paths}(i + 1), \text{paths}(i)) = D(\text{paths}(i), \text{paths}(i + 1))
\]
end for 
else 
\[
\text{flag} = 1
\]
end if 
end if 
end while

Algorithm 1: A bioinspired method for the constrained shortest path problem.

Here, \( P'(s, t) \) meets the following constraint:

\[
\Delta_{\text{delay}} \geq D(p), \quad p \in P(s, t), \quad (14)
\]
where \( P(s, t) \) represents the set of all paths from node \( s \) to node \( t \).

As shown in Figure 3, it is a real network topological structure with source node 1 and sink node 23. It was frequently an example in fuzzy shortest path problem [54, 59, 60]. The delay and cost of each link in the network are randomly generated. The costs of links are randomly generated by normal distribution with mean value equaling 10 and standard deviation equaling 3. The delays of links are randomly generated using normal distribution with mean value equaling 10 and standard deviation equaling 5. The cost and delay of each link randomly generated by normal distribution are shown in Table 2.
Table 3: Paths and their attributes.

<table>
<thead>
<tr>
<th>Order</th>
<th>k shortest path</th>
<th>Cost</th>
<th>Important path</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-411-17-20-23</td>
<td>54.1798</td>
<td>1-4-17-11-20-23</td>
<td>54.1798</td>
</tr>
<tr>
<td>2</td>
<td>1-3-8-13-19-22-23</td>
<td>54.1798</td>
<td>1-3-8-13-19-15-18-23</td>
<td>56.3484</td>
</tr>
<tr>
<td>3</td>
<td>1-4-11-17-21-23</td>
<td>57.8418</td>
<td>1-4-7-10-17-21-23</td>
<td>64.6681</td>
</tr>
<tr>
<td>4</td>
<td>1-4-11-14-21-23</td>
<td>58.1237</td>
<td>1-2-6-9-16-20-23</td>
<td>64.6681</td>
</tr>
<tr>
<td>5</td>
<td>1-4-11-17-21-23</td>
<td>57.8418</td>
<td>1-3-8-13-19-22-23</td>
<td>44.0553</td>
</tr>
<tr>
<td>6</td>
<td>1-4-7-10-16-20-23</td>
<td>52.0383</td>
<td>1-3-8-13-19-22-23</td>
<td>44.0553</td>
</tr>
<tr>
<td>7</td>
<td>1-4-7-11-17-20-23</td>
<td>61.3472</td>
<td>1-3-8-13-19-22-23</td>
<td>44.0553</td>
</tr>
<tr>
<td>8</td>
<td>1-4-7-11-17-20-23</td>
<td>61.3472</td>
<td>1-3-8-13-19-22-23</td>
<td>44.0553</td>
</tr>
<tr>
<td>9</td>
<td>1-4-7-11-17-20-23</td>
<td>61.3472</td>
<td>1-3-8-13-19-22-23</td>
<td>44.0553</td>
</tr>
<tr>
<td>10</td>
<td>1-3-8-13-19-22-23</td>
<td>49.6080</td>
<td>1-3-8-13-19-22-23</td>
<td>44.0553</td>
</tr>
</tbody>
</table>

Table 4: Setting parameter values.

<table>
<thead>
<tr>
<th>No.</th>
<th>BA model</th>
<th>Constraint</th>
<th>Cost</th>
<th>Delay</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>std</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m</td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean</td>
<td>std</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean</td>
<td>std</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean</td>
<td>std</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean</td>
<td>std</td>
<td></td>
</tr>
</tbody>
</table>

In order to get significative constraint, the delay constraint is got as follows [20]:

\[ \Delta_{\text{delay}} = p \ast (p_{dc} - p_{ld}) + p_{ld} \]  \hspace{1cm} (15)

where \( p_{ld} \) is the least delay and \( p_{dc} \) is the delay of the path which has the least cost. They can be obtained by amoeba algorithm. The parameter \( p \) is a problem specific ratio that measures the “tightness” of constraint (3) such that \( 0 < p < 1 \). The more constraining (or “tight”) the value of \( T \) in constraint (3) is, the lower the value of \( p \) will be. For simplicity, here, we let \( p = 0.1 \).

Here, the least delay path is 1-3-8-13-19-22-23 and its delay is 44.0553. The least cost path is 1-4-11-17-20-23 and its delay is 54.1798 and its cost is 48.7660. The constraint 45.0680 can be obtained by using (15): \( \Delta_{\text{delay}} = 0.1 \ast (54.1798 - 44.0553) + 44.0553 = 45.0680 \).

Let \( \theta \) be 45.0680, \( \kappa 2 \), and \( \gamma 30 \). Finally, we can get the solution: 1-3-8-13-19-22-23. The cost of the solution is 74.5885.

The solution is exactly the same as the method [14] as well. Table 3 lists these k shortest paths [14] and important paths.

As can be seen in Table 3, our searching order is different from Handler and Zang’s. The reason is that the order of important path depends on the specific values of the parameters \( \kappa \) and \( \gamma \). Generally speaking, smaller \( \kappa \) and larger \( \gamma \) lead to a bigger step when it searches for the following path in our method. Otherwise, the step will be small. It will search the alternative paths one by one. The parameters setting is in association with the scale of network and the parameter \( p \). In the case of a large scale of network, different parameters result in a big difference in executing time. The problem of how to set specific parameters will be further studied in the future work.

5. Computational Comparisons of the Solution Algorithms

In order to demonstrate the accuracy of proposed method, a number of experiments have been conducted on different networks. In addition, the CPU run-time has been compared between our method and the method proposed by Handler and Zang [14]. The main idea of the algorithm proposed by Handler and Zang is to use the k shortest paths algorithm to find the next shortest path terminating with the first path satisfying the constraint. This algorithm requires solving the k shortest path problem. The method finding the k shortest loopless paths in a network proposed by Yen [61] and the Dijkstra algorithm for finding the shortest path are used to solve the k shortest path problem.

Sixty-five networks were randomly generated. There are 5 different networks sizes based on the number of nodes in the networks. For each network in the same size, four networks are randomly generated. For each of the four
random networks, three different attributes, but the same topologies networks, are generated. These attributes are randomly generated by normal distribution.

As for the random networks, we use the BA model [62] to generate a randomly directed network. In BA model, there are two assumptions: firstly, the network is formed by the continuous addition of new vertices to the system. Therefore, the number of vertices $N$ increases throughout the lifetime of the network. Secondly, when there is a new node added in the network, it tends to be connected with the vertex who has more connections.

Under this assumption, we use the following steps to construct the network.

(i) Beginning with a small network $G_0$, it has $N_0$ nodes and $E_0$ edges. At each time step, we add only one new node.

(ii) Assume there are $n$ nodes ($s_1, s_2, \ldots, s_n$). When adding a new node $s_{n+1}$, it has $m$ ($\leq N_0$) edges linking $m$ different already existing nodes in the network.

(iii) $d_i$ indicates the degree of node $s_i$ ($1 \leq i \leq n$). The linking probability $P_i$ between nodes $s_i$ and $s_{n+1}$ is defined as follows:

$$P_i = \frac{d_i}{\sum_{j=1}^{n} d_j}.$$  

(16)

Because the network is generated by this model, its degree distribution follows the power-law distribution with $\gamma = 2.9 \pm 0.1$. In this paper, we let $N_0, E_0$, and $m$ be equal to 3.

All parameter values are listed in Table 4.

The tests are executed on Intel(R) Core(TM) i3-2120 processor (3.30GHz) with 2 GB of RAM under Windows Seven. Each instance is run for 3 times, and we compute the average executing time and average accuracy. Results of 65 test problems are summarized in Table 5.

After experiment, the accuracy is 95.38% in the proposed method. In all the 65 networks, there are only 3 networks we cannot get optimal solution when $\kappa = 2, \gamma = 30$. However, we have verified that once these two parameters are adjusted, the optimal solution can be obtained. As can be seen in Table 5, the method proposed by Handler and Zang [14] is faster than the approach proposed by this paper. However, the method combining amoeba model and penalty mechanism has three potential advantages. First of all, the CSP problem is an NP-hard problem and it is difficult for us to solve this problem. The method proposed by this paper is to contribute an innovative idea and to enrich the solution strategies for solving this NP-hard problem. Secondly, the reason why the proposed method is relatively slow is that the amoeba model in this paper is implemented in a tradition way. In fact, there are already many investigations on parallel amoeba algorithm which have dramatically decreased the computational time. For example, worthwhile works [63, 64] have been achieved. Amoeba algorithm is constantly improving and it has a great potential for the development of the bioinspired artificial intelligence. It is certain that the computational time of the proposed method will be greatly reduced if there was a sophisticatedly parallel amoeba algorithm. At the current stage, we just aim at its methodology. Finally, the proposed method is easy to combine with other algorithms. By combining with other approaches, we can take the advantages of both of them. To sum up, although the executing time of the proposed method is relatively slow, it can be improved with the implementation of the parallel amoeba algorithm. What is more, the proposed method has many advantages over the existing algorithms.
6. Conclusion

In this paper, we propose an innovative method to solve the CSP, which employs the internal mechanism of the amoeba model. The solution to the CSP consists of two parts. Firstly, we extend Physarum algorithm into the shortest path problem in directed networks. Secondly, we combine the Physarum algorithm with a bioinspired rule to deal with the CSP. In addition, we have shown the validity of the proposed method by comparing with other existing approaches using an example in the DCLC problem.

In the future, on the one hand, we will further investigate how to determine the associated parameters, such as $\kappa$ and $\gamma$. Currently, these parameters are given according to the many experimental results. In the near future, we will try to provide theoretical analysis to these parameters and to provide an interval to these parameters. On the other hand, we will explore how to implement the proposed method in parallel environment, which will greatly reduce the executing time of the proposed method.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Authors’ Contribution

Hongping Wang and Xiaoge Zhang contributed equally.

Acknowledgments

The work is partially supported by National Natural Science Foundation of China (Grant no. 61174022), Specialized Research Fund for the Doctoral Program of Higher Education (Grant no. 20131102130002), R&D Program of China (2012BAH07B01), National High Technology Research and Development Program of China (863 Program) (Grant no. 2013AA013801), the open funding project of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (Grant no. BUAA-VR-14KF-02), the Fundamental Research Funds for the Central Universities, Grant no. XDJK2013D010, and Southwest University Research Funds for Undergraduate Students in School of Computer and Information Science, Grant no. SWUJX2013-18.

References


