International Journal of Multimedia and Ubiquitous Engineering
Vol.12, No.2 (2017), pp.41-68
http://dx.doi.org/10.14257/ijmue.2017.12.2.04

Solving the Path Problem with Required Length

Zhi-xiong Su [1], Han-ying Wei [1] and Xue-min Yu [2]

[1]Business Administration College, Nanchang Institute of Technology, Nanchang, Jiangxi, China
[2]School of Government, Beijing Normal University, Beijing, China
suzhixiongbaner@126.com, happywhy@126.com, qjx640@126.com

Abstract

Path problems are core problems of graph theory, and the representative one is the shortest path problem. But the short path problem only is a special issue, and in many cases the most reasonable solution of a path problem maybe some paths with a required length rather than the shortest path or the longest path. Thence the path problem with required length has more wide application, but it is also more difficult to solve. For the problem, we propose an idea that "first simplifying and then solving". Following the idea, firstly, we found the shortest path model and the longest path model of connected graph, and reveal and summary their properties as theorems which reflect relationships between their parameters and lengths of paths in graph; secondly, base on the path models, we design a simple algorithm to simplify the connected graph for reducing difficulty of the path problem, and further design polynomial algorithms to search the k*th* *shortest path and the* k*th* *longest path for finding all paths with required length in graph. Finally, we use an illustration to show the effect of solving the path problem with required length by first simplifying and then searching the* k*th* *shortest (longest) path.*

***Keywords**: operations research, path problem with required length, path model, equivalent simplification, the* k*th* *shortest (longest) path, connected graph*

1. Introduction

The path problem is an important branch of operations research and a foundation problem for forming graph theory, and furthermore it is a practical problem having close relationship to the practical production and actual life. In additional to the actual path problem, many practical problems can be transformed into the corresponding path problems, and thus they can be simplified and then solved easily since the path problems can be represented clearly and intuitively by using graph.

The most familiar path problem is the shortest path problem. Although the problem has been highly concerned, it still contains many unresolved problems, for example, the optimal solution of the famous traveling salesman problem cannot be calculated until now by using any polynomial algorithm since the problem is *NP*-hard. However, the shortest path is special path after all, thence the shortest path problem only is special path problem, and similarly the corresponding longest problem also is special path problem. Searching the shortest path or the longest path is very valuable in theory and application, but just as the optimal schemes of many practical problems may be not the extreme ones, for example the optimal speed of car may be not the fastest or the most fuel-efficient one but the equilibrium of them, the most reasonable schemes of many path problems also may be not the shortest paths or the longest paths. For instance, for choosing the tour line, assume a tourist has no preference for attractions, generally he will choose a line with moderate length based on his own interesting rather than choose the shortest or the longest lines; for another instance, for racing in city or among cities, generally the organizer will also choose a path with required length based on the racing as the track rather than choose the shortest or the longest path; and so on. Thence, in the general sense, the path problem with required length contains the shortest path problem and the longest path problem, and further it can be applied more widely with higher value than the two problems.

The two frequently used approaches for this path problem are the dynamic programming approach with complexity $O(mn)$ [1] and determining the k^{th} shortest (longest) path. Dynamic programming is more complex than the Dijkstra algorithm and critical path method, particularly in large-scale networks. Li *et al.* [2] and Qi *et al.* [3] presented a simple algorithm to search the k^{th} critical path in time networks, which inspire Su *et al.* to propose an algorithm for the k^{th}-shortest path problem in acyclic networks (to be published) [4]. The complexities of these algorithms are k times larger than the complexity of the Dijkstra algorithm and the critical path method, and may be one of the simplest algorithms for the k^{th} shortest (longest) path problem in acyclic networks. However, for a path with a desired length, the relationship between the value of k and the value of the desired length cannot be determined a priori; thus, almost all paths may be searched. Therefore, searching the k^{th} shortest (longest) path is similar to the enumeration method, which may be inefficient, particularly in large-scale networks. Unfortunately, high computation times may be unavoidable even though advanced algorithms and computers are used. In an attempt to tackle this problem, we first propose a simplification that converts the problem into an equivalent and much simpler problem by eliminating non-desired paths, and then search the desired paths in the simplified graph. This simplification is conducive to obtaining the optimal solution.

Some authors studied path problems with desired paths, such as finding paths that meet desired vertex restrictions. Kawarabayashi and Kobayashi [5] studied the induced disjoint path problem, which finds desired disjoint paths that have neither common vertices nor adjacent vertices, and designed a linear time algorithm in planar graphs. Bolívar *et al.* [6], and Mokarami and Mehdi Hashemi [7] studied the constrained shortest path problems that find paths under desired conditions such as time constrained and weight constrained. Fiala *et al.* [8] and Bodlaender *et al.* [9] studied the k-path problem that tests whether a graph contains an induced path that spans k given vertices and solved the problem in polynomial time on claw-free graphs. For the path problem with desired length, many authors have tried to study similar problems, such as the next-to-shortest path problem and the k^{th} shortest path problems. Fox [10]-[12] and András and Péter [13] first studied the k^{th} shortest path problem. Lari *et al.* [14] considered the bounded length median path problem and compared the tabu search to the old bachelor acceptance. Kim [15] considered a tree with edge lengths and edge weights, and designed a linear time algorithm to find a maximum weight path that is less than a given length. Wu [16] designed a simpler and more efficient algorithm to find an st-path that has the shortest length among all st-paths and is strictly longer than the shortest path length in an undirected network with positive arc lengths and two vertices s and t. Current works for the k^{th}-shortest path problem include Cormen *et al.* [1], Wang[17], Feng [18], Lirov [19], Hiroshi [20], Chen and Tang [21], Li *et al.* [22], Gao *et al.* [23], Pascoal and Sedeno-Noda [24], Jiang *et al.* [25], Liu *et al.* [26], and Antonio and Sergio. [27]. Particularly, Gao *et al.* [23] designed an algorithm implemented with a bi-directional search, and Cormen *et al.* [1] provided a dynamic programming approach to find the k^{th}-shortest paths. Li *et al.* [22] designed a free float approach for the k^{th}-longest path problem in acyclic networks.

These methods are conducive to the path problem with desired length; however, they are computationally complex in large-scale graphs because k may be large. The problem is much more difficult than the shortest and longest path problems, especially in the current and future circumstances that we will have to deal with more and more difficult practical problems. For a large-scale problem with complex graph, it is hard to avoid the super-large computation of solving the problem even by using the most advanced computer and algorithm, and it is also difficult to overcome the bottleneck by designing algorithm and improving computer technology. Therefore, in this study, we propose the idea of simplification that reducing scale of the problem by removing unwanted paths, rather than directly solve the problem by searching the required paths directly. Little research has been done on simplifying path problems. Su *et al.* [28] simplified a similar path problem in acyclic networks, and we will improve the approach to apply to other types of connected graphs. After simplifying the problem, the required paths as final solutions of the problem can be found more easily, and we propose simple algorithms for the simplified problem.

2. Path Models of Connected Graph

According to the concept of connected graph, it is either directed graph or undirected graph. The undirected graph can be transformed into the directed one by representing each edge e_{ij} as one arc v_iv_j and another reversed arc v_jv_i with the same lengths, hence we can regard all connected graphs as directed ones for uniformly describing them.

For conveniently simplifying and solving the path problem in connected graph, we found the corresponding path models which contain the shortest path model and the longest path model. Su *et al.* [28] founded the path models of acyclic networks, and now we further improve it for other types of connected graphs. Founding the path models of connected graph mainly contains that, setting parameters of nodes and arcs based on the structure of graph, designing algorithms of computing the parameters, and revealing their properties. These parameters can be used to analyze the property of graph especially the properties of paths in graph, which help to solve practical problems.

Set a start node v_s and a terminal node v_t in connected graph, and the length of each arc v_iv_j is w_{ij}. The main idea for founding path model in the paper is that, considering paths from the start node v_s to each node v_j and from each node v_j to the terminal node v_t, and further revealing the property and law of the graph's structure by setting parameters. We design the algorithm of computing each parameter firstly in this section, and discourse the property of each parameter in next section. For different kinds of graphs, the algorithms of searching the longest and the shortest paths are also different. We design all connected graphs into following three kinds ones and discourse them respectively.

2.1. Path Models of Connected Graph without Cycle

If there is no cycle in a connected graph, we can reference the critical path method of the network planning technology to found the path models.

2.1.1. Found the Shortest Path Model

Step 1. Set a parameter α_j of each node v_j, and compute it from the start node v_s as follows:

$$\begin{cases}\alpha_s = 0 \\ \alpha_j = \min\limits_{i}\left\{\alpha_i + w_{ij}\right\}\end{cases} \tag{1}$$

Step 2. Set a parameter β_j of each node v_j, and compute it from the terminal node v_t as follows:

$$\begin{cases}\beta_t = \alpha_t \\ \beta_j = \max\limits_{k}\left\{\beta_k - w_{jk}\right\}\end{cases} \tag{2}$$

Step 3. Set a parameter N_t of each node v_j and parameters A_{ij}^{α}, A_{ij}^{β} and $A_{ij}^{\alpha\beta}$ of each arc v_iv_j, and compute them as follows:

$$
\begin{aligned}
N_j &= \alpha_j - \beta_j \\
A_{ij}^{\alpha} &= \alpha_i + w_{ij} - \alpha_j \\
A_{ij}^{\beta} &= \beta_i + w_{ij} - \beta_j \\
A_{ij}^{\alpha\beta} &= \alpha_i + w_{ij} - \beta_j
\end{aligned} \tag{3}
$$

2.1.2. Found the Longest Path Model

Step 1. Set a parameter α'_j of each node v_j, and compute it from the start node v_s as follows:

$$
\begin{cases}
\alpha'_s = 0 \\
\alpha'_j = \max\limits_{i} \left\{ \alpha'_i + w_{ij} \right\}
\end{cases} \tag{4}
$$

Step 2. Set a parameter β'_j of each node v_j, and compute it from the terminal node v_t as follows:

$$
\begin{cases}
\beta'_t = \alpha'_t \\
\beta'_j = \min\limits_{k} \left\{ \beta'_k - w_{jk} \right\}
\end{cases} \tag{5}
$$

Step 3. Set a parameter N'_j of each node v_j and parameters $A_{ij}^{\prime\alpha}$, $A_{ij}^{\prime\beta}$ and $A_{ij}^{\prime\alpha\beta}$ of each arc $v_i v_j$ and compute them as follows:

$$
\begin{aligned}
N'_j &= \beta'_j - \alpha'_j \\
A_{ij}^{\prime\alpha} &= \alpha'_j - w_{ij} - \alpha'_i \\
A_{ij}^{\prime\beta} &= \beta'_j - w_{ij} - \beta'_i \\
A_{ij}^{\prime\alpha\beta} &= \beta'_j - w_{ij} - \alpha'_i
\end{aligned} \tag{6}
$$

2.2. A Path Model of the Graph with Cycles but without Negative Length Arc

If there are cycles in a connected graph, the method in Section 2.1 is unfeasible to found the path model of the graph. In addition, for the connected graph with negative length arc and the one without negative length arc, the methods of founding the path models of them are also quite different. Now we consider the graph with cycles but without negative length arc. In the graph, searching the longest path is *NP*-hard, therefore it is difficult to found the longest path model. We mainly found the shortest path model.

The key of founding the shortest path model is to compute the length of the shortest path from the start node to any node, and from any node to the terminal node. For the connected graph with cycles but without negative length arc, the best algorithm of computing length of the shortest path between nodes is the Dijkstra algorithm. We reference the algorithm to found the shortest path model as follows:

Step 1. Set and compute a parameter α_j of each node v_j.

Step 1-1. For the start node v_s, let

$$
\alpha_s = 0 \tag{7}
$$

Step 1-2. Assume $v_i \in X$ represents that the parameter α_i of the node v_i is calculated, and $v_j \in \bar{X}$ represents that the parameter α_j of the node v_j is not calculated, find all arcs $v_i v_j \in X\bar{X}$ which meet $v_i \in X$ and $v_j \in \bar{X}$.

Step 1-3. If $X\bar{X} = \varnothing$, turn to Step 2; and if $X\bar{X} \neq \varnothing$, compute

$$\min_{v_i v_j \in X\bar{X}} \{\alpha_i + w_{ij}\} = \alpha_{i_x} + w_{i_x j_x}$$

and get the parameter α_{j_x} of the node $v_{j_x} \in \bar{X}$ that

$$\alpha_{j_x} = \alpha_{i_x} + w_{i_x j_x} \tag{8}$$

Turn to Step 1-2. If the parameter α_{v_t} of the terminal node v_t is calculated, turn to Step 2.

Step 2. Set and compute a parameter β_{v_j} of each node v_j.

Step 2-1. For the terminal node v_t, let

$$\beta_t = \alpha_t \tag{9}$$

Step 2-2. Assume $v_k \in Y$ represents that the parameter β_k of the node v_k is calculated, and $v_j \in \bar{Y}$ represents that the parameter β_j of the node v_j is not calculated, find all arcs $v_j v_k \in \bar{Y}Y$ which meet $v_j \in \bar{Y}$ and $v_k \in Y$.

Step 2-3. If $\bar{Y}Y = \varnothing$, turn to Step 3; and if $\bar{Y}Y \neq \varnothing$, compute

$$\max_{v_j v_k \in \bar{Y}Y} \{\beta_k - w_{jk}\} = \beta_{k_y} - w_{j_y k_y}$$

and get the parameter β_{j_y} of the node $v_{j_y} \in \bar{Y}$ that

$$\beta_{j_y} = \beta_{k_y} - w_{j_y k_y} \tag{10}$$

turn to Step 2-2. If the parameter β_s of the terminal node v_t is calculated, turn to Step 3.

Step 3. Set a parameter N_j of each node v_j and parameters A_{ij}^{α}, A_{ij}^{β} and $A_{ij}^{\alpha\beta}$ of each arc $v_i v_j$, and compute them as follows:

$$\begin{aligned} N_j &= \alpha_j - \beta_j \\ A_{ij}^{\alpha} &= \alpha_i + w_{ij} - \alpha_j \\ A_{ij}^{\beta} &= \beta_i + w_{ij} - \beta_j \\ A_{ij}^{\alpha\beta} &= \alpha_i + w_{ij} - \beta_j \end{aligned} \tag{11}$$

2.3. A Path Model of the Graph with Cycles and Negative Length Arcs

Now we consider the connected graph with cycles and negative length arcs. If there are cycles and negative length arcs in a connected graph, the Dijkstra algorithm is also unfeasible to compute the length of the shortest path. The Bellman-Ford algorithm is an effective algorithm for the shortest path problem in the graph, hence we reference it to found path model. Assume lengths of the cycles are positive, the longest path problem in the graph also is *NP*-hard. We mainly found the shortest path model as follows:

Step 1. Set and compute a parameter α_j of each node v_j.

Step 1-1. For the start node v_s, let

$$\alpha_s = 0 \tag{12}$$

Step 1-2. For each hinder adjacent arc $v_s v_j$ of the start node v_s, let

$$d''^{(1)}\left(v_s,v_j\right)=w_{sj}$$

Step 1-3. For $x=2,3,\cdots$,

$$d''^{(x)}\left(v_s,v_j\right)=\min_i\left\{d''^{(x-1)}\left(v_s,v_i\right)+w_{ij}\right\}$$

and there cannot be a node v_j on the path $v_s\cdots v_i$ which corresponding to $d''^{(x-1)}\left(v_s,v_i\right)$. If $x=y$ and each node v_j meet

$$d''^{(y)}\left(v_s,v_j\right)=d''^{(y-1)}\left(v_s,v_j\right)$$

then let

$$\alpha_j=d''^{(y)}\left(v_s,v_j\right) \tag{13}$$

Step 2. Set and compute a parameter β_j of each node v_j.

Step 2-1. For the terminal node v_t, let

$$\beta_t=\alpha_t \tag{14}$$

Step 2-2. For each former adjacent arc v_jv_t of the terminal node v_t, let

$$d'''^{(1)}\left(v_j,v_t\right)=\beta_t-w_{jt}$$

Step 2-3. For $x=2,3,\cdots$,

$$d'''^{(x)}\left(v_j,v_t\right)=\max_k\left\{d'''^{(x-1)}\left(v_k,v_t\right)-w_{jk}\right\}$$

and there cannot be a node v_j on the path $v_k\cdots v_t$ which corresponding to $d'''^{(x-1)}\left(v_k,v_t\right)$. If $x=y$ and each node v_j meet

$$d'''^{(y)}\left(v_j,v_t\right)=d'''^{(y-1)}\left(v_j,v_t\right)$$

then let

$$\beta_j=d'''^{(y)}\left(v_j,v_t\right) \tag{15}$$

Step 3. Set a parameter N_j of each node v_j and parameters A_{ij}^{α}, A_{ij}^{β} and $A_{ij}^{\alpha\beta}$ of each arc v_iv_j, and compute them as follows:

$$\begin{aligned}N_j&=\alpha_j-\beta_j\\A_{ij}^{\alpha}&=\alpha_i+w_{ij}-\alpha_j\\A_{ij}^{\beta}&=\beta_i+w_{ij}-\beta_j\\A_{ij}^{\alpha\beta}&=\alpha_i+w_{ij}-\beta_j\end{aligned} \tag{16}$$

3. Properties of the Parameters in the Path Model

This paper studies the relationships between the parameters of the path model and the length of the path in graph, viz. the properties of the parameters, and summaries them as related theorems. These theorems are main basis for simplifying and solving the path problem. For discoursing easily, we name the path from the start node to the terminal node of graph as a "path", and name the other kind of path as a "path section".

Theorem 1. In a connected graph, for any node v_j, the parameter α_j is equal to the length of the shortest path section $\mu_{s\to j}^{\min}$ from the start node v_s to the node v_j, that is

$$\alpha_j=L\left(\mu_{s\to j}^{\min}\right) \tag{17}$$

The parameter β_j is equal to the difference between the length of the shortest path $\mu^{\min}$ of the graph and the length of the shortest path section $\mu_{j\to t}^{\min}$ from the node v_j to the terminal node v_t, that is

$$\beta_j = L\left(\mu^{\min}\right) - L\left(\mu_{j\to t}^{\min}\right) \tag{18}$$

The parameter α'_j is equal to the length of the longest path section $\mu_{s\to j}^{\max}$ from the start node v_s to the node v_j, that is

$$\alpha'_j = L\left(\mu_{s\to j}^{\max}\right) \tag{19}$$

And the parameter β'_j is equal to the difference between the length of the longest path $\mu^{\max}$ of the graph and the length of the longest path section $\mu_{j\to t}^{\max}$ from the node v_j to the terminal node v_t, that is

$$\beta'_j = L\left(\mu^{\max}\right) - L\left(\mu_{j\to t}^{\max}\right) \tag{20}$$

Proof. According to the algorithm of computing the parameter α_j in each kind of graph, for the start node v_s,

$$\alpha_s = 0 = L\left(\mu_{s\to s}^{\min}\right)$$

And for any other node v_j except the start node v_s, we summarize that

$$\alpha_j = \min_i\left\{\alpha_i + w_{ij}\right\},\ \alpha_i = \min_h\left\{\alpha_h + w_{hi}\right\}, \ldots, \alpha_a = \min_s\left\{\alpha_s + w_{sa}\right\}$$

and further deduce by iterating that

$$\begin{aligned}
\alpha_j &= \min_i\left\{\alpha_i + w_{ij}\right\} \\
&= \min_i\left\{\min_h\left\{\alpha_h + w_{hi}\right\} + w_{ij}\right\} \\
&\ \cdots\cdots \\
&= \min_i\left\{\min_h\left\{\min_g\left\{\cdots\min_s\left\{\alpha_a + w_{sa}\right\} + w_{ab}\right\} + \cdots + w_{hi}\right\} + w_{ij}\right\} \\
&= \min_{s\to j}\left\{w_{sa} + w_{ab} + \cdots + w_{hi} + w_{ij}\right\} \\
&= L\left(\mu_{s\to j}^{\min}\right)
\end{aligned}$$

Therefore

$$L\left(\mu_{s\to j}^{\min}\right) = \alpha_j$$

and the Equation (17) is correct.

Then according to the algorithm of computing the parameter β_j in each kind of graph, for the terminal node v_t,

$$\beta_t = \alpha_t = L\left(\mu_{s\to t}^{\min}\right) = L\left(\mu^{\min}\right)$$

And for any other node v_j except the terminal node v_t, we can summarize that

$$\begin{aligned}
&\beta_j = \max_k \left\{\beta_k - w_{jk}\right\} \\
&\beta_k = \max_l \left\{\beta_l - w_{kl}\right\} \\
&\cdots \\
&\beta_y = \max_t \left\{\beta_t - w_{yt}\right\}
\end{aligned}$$

and further deduce by iterating that

$$\begin{aligned}
\beta_j &= \max_k \left\{\beta_k - w_{jk}\right\} \\
&= \max_k \left\{\max_l \left\{\beta_l - w_{kl}\right\} - w_{jk}\right\} \\
&\cdots\cdots \\
&= \max_k \left\{\max_l \left\{\cdots\left\{\max_y \left\{\beta_t - w_{yt}\right\} - w_{xy}\right\} - \cdots - w_{kl}\right\} - w_{jk}\right\} \\
&= \max_{j\to t} \left\{\beta_t - w_{jk} - w_{kl} - \cdots - w_{xy} - w_{yt}\right\} \\
&= \beta_t - \min_{j\to t} \left\{w_{jk} + w_{kl} + \cdots + w_{xy} + w_{yt}\right\} \\
&= L\left(\mu^{\min}\right) - L\left(\mu^{\min}_{j\to t}\right)
\end{aligned}$$

therefore

$$L\left(\mu^{\min}_{j\to t}\right) = L\left(\mu^{\min}\right) - \beta_j$$

and the Equation (18) is correct.

Similarly, we can prove that the Equations (19) and (20) are correct. This completes the proof.

Corollary 1. In a connected graph, the length of the shortest path $\mu^{\min}$ is

$$L\left(\mu^{\min}\right) = \alpha_t = \beta_t \tag{21}$$

and the length of the longest path $\mu^{\max}$ is

$$L\left(\mu^{\max}\right) = \alpha'_t = \beta'_t \tag{22}$$

Theorem 2. In a connected graph, for any node v_i, the parameter N_i is equal to the difference between the length of the shortest path $\mu_i^{\min}$ passing the node and the length of the shortest path $\mu^{\min}$ of the graph, that is

$$\begin{aligned}
N_i &= L\left(\mu_i^{\min}\right) - L\left(\mu^{\min}\right) \\
&= L\left(\mu_i^{\min}\right) - \alpha_t \\
&= L\left(\mu_i^{\min}\right) - \beta_t
\end{aligned} \tag{23}$$

And the parameter N'_i is equal to the difference between the length of the longest path $\mu^{\max}$ of the graph and the length of the longest path $\mu_i^{\max}$ passing the node v_i, that is

$$\begin{aligned}
N'_i &= L\left(\mu^{\max}\right) - L\left(\mu_i^{\max}\right) \\
&= \alpha'_t - L\left(\mu_i^{\max}\right) \\
&= \beta'_t - L\left(\mu_i^{\max}\right)
\end{aligned} \tag{24}$$

Proof. (1) In a graph, the shortest path $\mu_i^{\min}$ passing any node v_i is composed by two sections, the one section is the shortest path section $\mu_{s\to i}^{\min}$ from the start node v_s to the node v_i, and the other one is the shortest path section $\mu_{i\to t}^{\min}$ from the node v_i to the terminal node v_t, that is

$$\mu_i^{\min} = \mu_{s\to i}^{\min} + \mu_{i\to t}^{\min}$$

and its length is

$$L\left(\mu_i^{\min}\right) = L\left(\mu_{s\to i}^{\min}\right) + L\left(\mu_{i\to t}^{\min}\right)$$

According to the Theorem 1,

$$\begin{aligned} L\left(\mu_i^{\min}\right) &= L\left(\mu_{s\to i}^{\min}\right) + L\left(\mu_{i\to t}^{\min}\right) \\ &= \alpha_i + L\left(\mu^{\min}\right) - \beta_i \\ &= L\left(\mu^{\min}\right) + \left(\alpha_i - \beta_i\right) \\ &= L\left(\mu^{\min}\right) + N_i \end{aligned}$$

and then according to the Equation (21),

$$\begin{aligned} N_i &= L\left(\mu_i^{\min}\right) - L\left(\mu^{\min}\right) \\ &= L\left(\mu_i^{\min}\right) - \alpha_t \end{aligned}$$

therefore the Equation (23) is correct.

(2) Similarly, according to the Equations (6)~(9), we can prove that the Equation (24) is correct. Thence the theorem is correct. This completes the proof.

Theorem 3. In a connected graph, for any arc v_iv_j, the parameter $A_{ij}^{\alpha\beta}$ is equal to the difference between the length of the shortest path $\mu_{ij}^{\min}$ passing the node and the length of the shortest path $\mu^{\min}$ of the graph, that is

$$\begin{aligned} A_{ij}^{\alpha\beta} &= L\left(\mu_{ij}^{\min}\right) - L\left(\mu^{\min}\right) \\ &= L\left(\mu_{ij}^{\min}\right) - \alpha_t \\ &= L\left(\mu_{ij}^{\min}\right) - \beta_t \end{aligned} \tag{25}$$

And the parameter $A_{ij}^{\prime\alpha\beta}$ is equal to the difference between the length of the longest path $\mu^{\max}$ of the graph and the length of the longest path $\mu_{ij}^{\max}$ passing the arc v_iv_j, that is

$$\begin{aligned} A_{ij}^{\prime\alpha\beta} &= L\left(\mu^{\max}\right) - L\left(\mu_{ij}^{\max}\right) \\ &= \alpha_t' - L\left(\mu_{ij}^{\max}\right) \\ &= \beta_t' - L\left(\mu_{ij}^{\max}\right) \end{aligned} \tag{26}$$

Proof. It is similar to the proof for the Theorem 2.

Theorem 4. In a connected graph, the difference between the length of any path μ and the length of the shortest path $\mu^{\min}$ of the graph is equal to the sum of the parameters A_{ij}^{α} or the sum of the parameters A_{ij}^{β} of all arcs v_iv_j on the path μ, that is

$$L(\mu) - L\left(\mu^{\min}\right) = \sum_{\mu} A_{ij}^{\alpha} = \sum_{\mu} A_{ij}^{\beta} \tag{27}$$

And the difference between the length of the longest path $\mu^{\max}$ of the graph and the length of the path μ is equal to the sum of the parameters $A_{ij}^{\prime\alpha}$ or the sum of the parameters A'^{β}_{ij} of all arcs v_iv_j on the path μ, that is

$$L\left(\mu^{\max}\right)-L(\mu)=\sum_{\mu}A'^{\alpha}_{ij}=\sum_{\mu}A'^{\beta}_{ij} \tag{28}$$

Proof. Assume any path $\mu=v_sv_av_b\cdots v_zv_t$, according to the Equations (3), (11) and (16),

$$\begin{aligned}\sum_{\mu}A^{\alpha}_{ij}&=\sum_{\mu}\left(\alpha_i+w_{ij}-\alpha_j\right)\\&=\left(\alpha_s+w_{sa}-\alpha_a\right)+\left(\alpha_a+w_{ab}-\alpha_b\right)+\cdots+\left(\alpha_z+w_{zt}-\alpha_t\right)\\&=\alpha_s+\left(w_{sa}+w_{ab}+\cdots+w_{zt}\right)-\alpha_t\\&=\alpha_s+L(\mu)-\alpha_t\\&=L(\mu)-L\left(\mu^{\min}\right)\end{aligned}$$

Similarly, we can prove

$$\sum_{\mu}A^{\beta}_{ij}=L(\mu)-L\left(\mu^{\min}\right)$$

therefore the Equation (27) is correct.

And similarly, we also can prove the Equation (28) is correct. Therefore the theorem is correct. This completes the proof.

4. An Algorithm for Simplifying Path Problem with Required Length

4.1. Description of the Algorithm

Assume a connected graph *G*, and require to finding out the path with the length λ from the start node v_s to the terminal node v_t in *G*. For solving the path problem easily, we simplify the graph *G*, and ensure all paths with the length λ are still in the remaining graph after simplification that the solution result of the problem will not be affected.

An algorithm for the simplification is following:

Step 1. Compute the parameters α_i, β_i and N_i of each node v_i in the *G*, and then remove the node v_k with $N_k>\lambda-\alpha_t$ and its adjacent arcs to obtain a graph *G*1.

Step 2. Compute the parameter $A^{\alpha\beta}_{ij}$ of each arc v_iv_j in the *G*1, and remove the arc v_xv_y with $A^{\alpha\beta}_{xy}>\lambda-\alpha_t$ to obtain a graph *G*2.

Step 3. Compute the parameters α'_i, β'_i and N'_i of each node v_i in the *G*2, and then remove the node v_k with $N'_k>\alpha'_t-\lambda$ and its adjacent arcs to obtain a graph *G*3.

Step 4. Compute the parameter $A'^{\alpha\beta}_{ij}$ of each arc v_iv_j in the *G*3, and then remove the arc v_xv_y with $A'^{\alpha\beta}_{xy}>\lambda-\alpha'_t$ to obtain a graph *G*4.

The graph *G*4 still contains all paths with the length λ in the original graph *G*, and it has much less nodes, arcs and especially paths than the *G*. Thence, it is much more convenient to find the paths with the length λ in the *G*4, which can be completed generally only by searching a few paths such as the longest path, the 2nd longest path, the shortest path or the 2nd shortest path, and so on.

4.2. Correctness of the Algorithm

(1) For any node v_i in the connected graph *G*, according to the Theorem 2 and the

Equation (23),

$$N_{v_i} = L\left(\mu_{v_i}^{\min}\right) - \alpha_{v_t}$$

If $N_i > \lambda - \alpha_t$, put above Equation into it that

$$L\left(\mu_i^{\min}\right) - \alpha_t > \lambda - \alpha_t \Rightarrow L\left(\mu_i^{\min}\right) > \lambda$$

It means that the shortest path passing the node v_i is longer than λ, thence all paths passing the node must be longer than λ, which are all not required ones. Removing the node and its adjacent arcs is equivalent to removing all these paths. After removing all this kind of nodes and their adjacent arcs, we mark the remaining part of the graph *G* as the G_1.

(2) Similarly, for any arc $v_i v_j$ in the G_1, according to the Theorem 3 and the Equation (25), if $A_{xy}^{\alpha\beta} > \lambda - \alpha_t$, all paths passing the arc must be longer than λ, which are also not required ones. Removing the arc is equivalent to removing all these paths. After removing all this kind of arcs, we mark the remaining part of the G_1 as the G_2.

Thence, the paths which are removed by using Steps 1 and 2 of the algorithm in Section 4.1 are all longer than λ.

(3) For any node v_i in the connected graph G_2, according to the Theorem 2 and the Equation (24),

$$N_i' = \alpha_t' - L\left(\mu_i^{\max}\right)$$

If $N_i' > \alpha_t' - \lambda$, put above Equation into it that

$$\alpha_t' - L\left(\mu_i^{\max}\right) > \alpha_t' - \lambda \Rightarrow L\left(\mu_i^{\max}\right) < \lambda$$

It means that the longest path passing the node v_i is shorter than λ, thence all paths passing the node must be shorter than λ, which are all not required ones. Removing the node and its adjacent arcs is equivalent to removing all these paths. After removing all this kind of nodes and their adjacent arcs, we mark the remaining part of the G_2 as the G_3.

(4) Similarly, for any arc $v_i v_j$ in the G_3, according to the Theorem 3 and the Equation (26), if $A_{xy}'^{\alpha\beta} > \lambda - \alpha_t'$, all paths passing the arc must be shorter than λ, which are also not required ones. Removing the arc is equivalent to removing all these paths. After removing all this kind of arcs, we mark the remaining part of the G_3 as the G_4.

Thence, the paths which are removed by using Steps 3 and 4 of the algorithm in Section 4.1 are all shorter than λ.

According to above analysis, when removing the paths which are longer or shorter than λ, all paths with the length λ are reserved. It means that, although the G_4 is simpler than the original graph *G*, it still contains all paths with the length λ and can be used to solve the path problem instead of the *G*. Thence the algorithm for simplification in Section 4.1 is correct.

4.3. Complexity of the Algorithm

Assume there are *n* nodes and *m* arcs in the connected graph *G*, and $m \geq n$ based on the relationship between the node and the arc in the graph. The complexity of the algorithm in Section 4.1 is analyzed as follows:

Step 1 is for computing the parameters α_i, β_i and N_i of each node v_i. Thereinto, for different kinds of connected graphs, the algorithms of computing the parameters α_i and

β_i are also different.

(1) For the graph without cycle, the complexities of computing the parameter α_i and β_i are $O(m)$.

(2) For the graph with cycles but without negative length arc, the algorithm of computing α_i is similar to the Dijkstra algorithm, and its complexity is $O(m+n\log n)$; but the algorithm of computing β_i is similar to the Bellman-Ford algorithm, and its complexity is $O(mn)$, thence the total complexity of computing the two parameter is $O(mn)$.

(3) For the graph with cycles and negative length arcs, the complexity of computing the α_i and β_i are $O(mn)$.

(4) The parameter N_i is equal to the difference between α_i and β_i, thence the complexity of computing it is $O(n)$.

Step 2 is for computing the parameter $A_{ij}^{\alpha\beta}$ of each arc v_iv_j by using the parameters α_i and β_i of its nodes, and there are *m* arcs at most, thence the complexity of the step is $O(m)$.

Steps 3 and 4 are similar to Steps 1 and 2.

Therefore, for the connected graph without cycle, the complexity of the algorithm is $O(m)$; and for the connected graph with cycles, the complexity of the algorithm is $O(mn)$.

5. An algorithm for Searching the k^{th} Shortest Path

In a connected graph, the path with the required length λ is generally not the shortest path or the longest path, and it is very difficult to be searched directly, thence if wanting to find the path effectively, the idea of searching indirectly need be used. The idea of searching indirectly can be realized from two approaches, the one is that searching individually from the shortest path to long ones until to find the k^{th} shortest path with the length λ, and the other one is that searching individually from the longest path to short ones until to find the k^{th} longest path with the length λ. Su *et al.* (to be published) [4] proposed an algorithm for searching the k^{th} shortest path in acyclic networks. In this paper, we also consider other types of connect graphs. In this section, We design an algorithm for searching the k^{th} shortest path in any types of connect graphs, and further design an algorithm for searching the k^{th} longest path in next section.

In order to discourse the algorithm conveniently, we first propose concepts and calculations of some new parameters, and then discourse the algorithm in detail.

5.1. A Short Feature Parameter of the Terminal Node of Connected Graph

The short feature parameter of the terminal node v_t of a graph is defined as a value of the node to an arc, and it is also named that the terminal node v_t has a short feature parameter to an arc.

(1) We mark the 1st rank short feature parameter of the terminal node v_t as θ^1, and define its value as zero, that is

$$\theta^1 = 0 \tag{29}$$

(2) The 2nd rank short feature parameter of the terminal node v_t.

If there is a path composed by arcs with the parameters $A^\alpha = 0$ and from a node v_r to the terminal node v_t, and the parameter A^α_{qr} of an arc $v_q v_r$ is not zero in the former adjacent arcs of the node v_r, then the terminal node v_t has the 2nd rank short feature parameter to the arc $v_q v_r$. We mark the short feature parameter as θ^2_{qr}, and define its value as

$$\theta^2_{qr} = \theta^1 + A^\alpha_{qr} = A^\alpha_{qr} \tag{30}$$

(3) The *k*th rank short feature parameter of the terminal node v_t.

Assume the terminal node v_t has the $k-1$th rank short feature parameter θ^{k-1}_{de} to an arc $v_d v_e$, and $k \geq 2$, if there is a path composed by arcs with the parameters $A^\alpha = 0$ and from a node v_b to v_d, and the parameter A^α_{ab} of an arc $v_a v_b$ is not zero in the former adjacent arcs of the node v_b, then the terminal node v_t has the *k*th rank short feature parameter to the arc $v_a v_b$. We mark the short feature parameter as θ^k_{ab}, and define its value as

$$\theta^k_{ab} = \theta^{k-1}_{de} + A^\alpha_{ab} \tag{31}$$

5.2. An Algorithm for Searching the *k*th Shortest Path

5.2.1. Description of the Algorithm

For finding the path with the length λ in a connected graph, we design an algorithm for searching the *k*th shortest path.

Step 1. Search the 1st shortest path μ^1.

Step 1-1. Compute the parameter α_i of each node v_i and the parameter A^α_{ij} of each arc $v_i v_j$.

Step 1-2. Find connected arcs with the parameters $A^\alpha = 0$ forward from the terminal node v_t to the start node v_s to get a $\mu^1 = \mu^{\min}$ with the length $L(\mu^1) = \alpha_t$. This step cannot be stopped repeating until no other new μ^1.

Step 2. Form short feature sets Ω_1 and Ω_2.

Step 2-1. Form a $\Omega_1 = \varnothing$, then compute the parameters $A^\alpha_{a_2 b_2}$ of all former adjacent arcs $v_{a_2} v_{b_2}$ of each node on the μ^1, and get $\theta^2_{a_2 b_2} = A^\alpha_{a_2 b_2} > 0$.

Step 2-2. Form the Ω_2 by putting $\theta^2_{a_2 b_2}$ into the Ω_1.

Step 3. Search the *i*th ($i \geq 2$) shortest path μ^i.

Step 3-1. Find all minimum values $\theta^j_{a_j b_j}$ from the Ω_i.

Step 3-2. According to the Equation (31) (the Equation (30) if $i = 2$), find $\theta^{j-1}_{a_{j-1} b_{j-1}}$ (assume it corresponds to a μ^m) and an arc $v_{a_j} v_{b_j}$ with the parameter $A^\alpha_{a_j b_j} > 0$.

Step 3-3. Find connected arcs with the parameters $A^\alpha = 0$ forward from the node v_{a_j}

to the start node v_s to get a $\mu^{\min}_{s\to a_j}$. The μ^i is composed by the $\mu^{\min}_{s\to a_j}$, the arc $v_{a_j}v_{b_j}$ and the $\mu^m_{b_j\to t}$, and its length is

$$L\left(\mu^i\right)=\alpha_t+\theta^j_{a_jb_j} \tag{32}$$

Step 4. Check.

If $L\left(\mu^i\right)\ge\lambda$, stop; and if $L\left(\mu^i\right)<\lambda$, turn to Step 5.

Step 5. Form a short feature set Ω_{i+1}.

Step 5-1. Find all former adjacent arcs $v_{a_{j+1}}v_{b_{j+1}}$ with the parameters $A^\alpha_{a_{j+1}b_{j+1}}>0$ of each node on the $\mu^{\min}_{s\to a_j}$, and then compute $\theta^{j+1}_{a_{j+1}b_{j+1}}$ by using the Equation (31),

$$\theta^{j+1}_{a_{j+1}b_{j+1}}=\theta^j_{a_jb_j}+A^\alpha_{a_{j+1}b_{j+1}}$$

Step 5-2. Form a Ω_{i+1} by removing $\theta^j_{a_jb_j}$ from the Ω_i and putting $\theta^{j+1}_{a_{j+1}b_{j+1}}$ into the Ω_i, then turn to Step 3.

Note that, according to the algorithm in Section 2 of computing the parameter α_i of each node v_i, α_i represents the length of the shortest path section from the start node v_s to the node v_i and without cycle, thence the 1st shortest path $\mu^1=\mu^{\min}$ must be the shortest path without cycle in the graph.

Similarly, we also can search the *k*th shortest path in a connected graph by using the parameter β_i of each node v_i and the parameter A'^β_{ij} of each arc v_iv_j, and the detail steps is similar to above algorithm.

5.2.2. Correctness of the Algorithm

According to the Theorem 4 and the Equation (27),

$$L(\mu)=\alpha_t+\sum\nolimits_\mu A^\alpha_{ij}$$

thence searching a μ is equivalent to computing $\sum_\mu A^\alpha_{ij}$, for example, searching the μ^1 is equivalent to computing the minimum value of $\sum_\mu A^\alpha_{ij}$, searching the μ^2 is equivalent to computing the 2nd minimum value of $\sum_\mu A^\alpha_{ij}$, …… , and searching the μ^k is equivalent to computing the *k*th minimum value of $\sum_\mu A^\alpha_{ij}$.

In addition, according to the concept of the short feature parameter of the terminal node v_t and the Equation (31),

$$\begin{aligned}\theta^j_{a_jb_j}&=\theta^{j-1}_{a_{j-1}b_{j-1}}+A^\alpha_{a_jb_j}\\&=\theta^{j-2}_{a_{j-2}b_{j-2}}+A^\alpha_{a_{j-1}b_{j-1}}+A^\alpha_{a_jb_j}\\&\cdots\cdots\\&=\theta^1+A^\alpha_{a_2b_2}+A^\alpha_{a_3b_3}+\cdots+A^\alpha_{a_jb_j}\\&=\sum\nolimits_{i=2}^j A^\alpha_{a_ib_i}\end{aligned}$$

Thence there is a path $\mu_{a_j\to t}$, and the parameters A^α of only arcs $v_{a_j}v_{b_j}$, $v_{a_{j-1}}v_{b_{j-1}}$, … , $v_{a_2}v_{b_2}$ on the path are nonzero, that is

$$\theta^j_{a_jb_j}=\sum\nolimits_{i=2}^{j}A^\alpha_{a_ib_i}=\sum\nolimits_{\mu_{a_j\to t}}A^\alpha_{xy}$$

And according to the Theorem 1, there is a path $\mu_{s\to a_j}=\mu^{\min}_{s\to a_j}$ from the start node v_s to the node v_{a_j}, and $\sum\nolimits_{\mu_{s\to a_j}}A^\alpha_{a_xy}=0$. The $\mu_{s\to a_j}$ and $\mu_{a_j\to t}$ compose a $\mu_{s\to t}=\mu$, thence

$$\begin{aligned}\theta^j_{a_jb_j}&=\sum\nolimits_{\mu_{a_j\to t}}A^\alpha_{xb_y}\\&=\sum\nolimits_{\mu_{s\to a_j}}A^\alpha_{xy}+\sum\nolimits_{\mu_{a_j\to t}}A^\alpha_{xy}\\&=\sum\nolimits_{\mu}A^\alpha_{xy}\end{aligned}\tag{33}$$

Therefore, the sum of the parameters A^α of all arcs on a path is equal to a short feature parameter $\theta^i_{a_ib_i}$ of the terminal node v_t, and according to the Equation (32),

$$L(\mu)=\alpha_t+\theta^j_{a_jb_j}\tag{34}$$

Searching the μ^k is equivalent to computing the *k*th minimum value of $\theta^i_{a_ib_i}$, thereinto the minimum value is $\theta^1=0$.

(1) The proof that all μ^1 can be found by using Step 1 of the algorithm.

According to the Equation (29), if $\theta^1=\sum\nolimits_\mu A^\alpha_{ij}=0$, the corresponding μ is the shortest path $\mu^{\min}$ that $\mu^1=\mu^{\min}$.

According to the algorithm for computing the node parameter α and the arc parameter A^α, except the start node v_s, each arc has former arc with the parameter $A^\alpha=0$, thence the former adjacent arc v_qv_t with $A^\alpha_{qt}=0$ of the terminal node v_t can be found; similarly, the former adjacent arc v_pv_q with $A^\alpha_{pq}=0$ of the node v_q also can be found; …… ; the process cannot be stopped until to the start node v_s, and then a μ^1 with $\sum\nolimits_{\mu^1}A^\alpha_{ij}=0$ is found.

Since each arc may have more former adjacent arcs with the parameters $A^\alpha=0$, there may be more μ^1, and above process of searching the μ^1 cannot be stopped repeating until no other new μ^1.

Thence all μ^1 can be found by using Step 1 of the algorithm.

(2) For the 2nd ~ *k*th shortest paths, the algorithm uses recursive method to search them, thence here we use mathematical induction to prove its correctness.

1) Firstly, we prove that all μ^2 can be found by using Steps 2~3.

i) According to the Equation (30), if $\theta^i_{a_ib_i}$ is the 2nd value that the minimum nonzero value, it corresponds to the μ^2. $\sum\nolimits_{\mu^1}A^\alpha_{ij}=0$, and according to Step 1, as long as the path $\mu_{r\to t}$ from a node v_r to the terminal node v_t meets $\sum\nolimits_{\mu_{r\to t}}A^\alpha_{ij}=0$, then $\mu_{r\to t}\subset\mu^1$, thence according to the concept, the 2nd rank short feature parameter $\theta^2_{a_2b_2}$ of the terminal node v_t is the nonzero parameter $A^\alpha_{a_2b_2}$ of a former adjacent arc $v_{a_2}v_{b_2}$ of each node on μ^1. Therefore, by using Step 2, all the 2nd rank short feature parameters of the terminal node v_t can be calculated and then combined to form the Ω_2.

ii) Assume $\Omega_2 = \left\{\theta^2_{a_2b_2}, \theta^2_{c_2d_2}, \cdots, \theta^2_{x_2y_2}\right\}$, then according to the Equation (31), all nonzero short feature parameters of the terminal node v_t are the short feature parameters in the Ω_2 and the higher ranks short feature parameters generating based on these parameters, as in a Ω_2^* that

$$\Omega_2^* = \begin{Bmatrix} \theta^2_{a_2b_2}, \theta^3_{a_3b_3}, \cdots, \theta^l_{a_lb_l} \\ \theta^2_{c_2d_2}, \theta^3_{c_3d_3}, \cdots, \theta^m_{c_md_m} \\ \cdots\cdots \\ \theta^2_{x_2y_2}, \theta^3_{x_3y_3}, \cdots, \theta^n_{x_ny_n} \end{Bmatrix}, \quad l, m, \cdots, n \ge 2$$

And according to the Equation (31), $\theta^{i+1}_{a_{i+1}b_{i+1}} \ge \theta^i_{a_ib_i}$, so if $\theta^2_{a_2b_2}$ is the minimum value in the Ω_2, then it also is the minimum value in the Ω_2^* and the 2nd minimum value of $\theta^i_{a_ib_i}$. Therefore $\theta^2_{a_2b_2} = A^\alpha_{a_2b_2}$ corresponds to the μ^2 which is composed by the $\mu^{\min}_{s\to a_2}$, the arc $v_{a_2}v_{b_2}$ and the $\mu^1_{a_2\to t}$, and $\sum_{\mu^*_{s\to a_2}} A^\alpha_{xy} = 0$. According to the Equation (34), the length of the μ^2 is

$$L\left(\mu^2\right) = \alpha_t + \theta^2_{a_2b_2}$$

Base on above analysis, we view Step 3.

The all minimum $\theta^2_{a_2b_2}$ are found by using Step 3-1, which are all minimum values in the Ω_2, and it ensures that all μ^2 can be found by using Steps 3-2~3-3.

In Step 3-2, according to the Equation (30),

$$\theta^2_{a_2b_2} = \theta^1 + A^\alpha_{a_2b_2} = A^\alpha_{a_2b_2}$$

θ^1 and the arc $v_{a_2}v_{b_2}$ can be found based on above Equation. According to Step 1, θ^1 corresponds to the μ^1, thence the $\mu^1_{a_2\to t}$ can be found.

Searching the $\mu^{\min}_{s\to a_2}$ by using Step 3-3 is similar to searching the μ^1 by using Step 1, and the only difference is that Step 3 starts from the node v_{a_2}, thence the $\mu^{\min}_{s\to a_2}$ can be found by using Step 3-3.

Hence all μ^2 can be found by using Steps 2~3 of the algorithm.

2) Assume all μ^k are found by using Steps 3~5, now we prove that all μ^{k+1} can be found by using Steps 3~5.

Step 4 is used to judge whether searching the μ^{k+1} or not after finding the μ^k based on actual requirements. This step is independent to the process of searching the μ^{k+1}, thence we can omit it when proving the correctness of the algorithm for searching the μ^{k+1}.

i) For all the μ^k can be found, we analyze the step of searching the μ^k. Assume $\Omega_k = \left\{\theta^j_{a_jb_j}, \theta^p_{c_pd_p}, \cdots, \theta^q_{x_qy_q}\right\}$, then all the kth minimum short feature parameters, the $k+1$th minimum one, … of the terminal node v_t are showed in a Ω_k^* that

$$\Omega_k^* = \begin{Bmatrix} \theta_{a_j b_j}^j, \theta_{a_{j+1} b_{j+1}}^{j+1}, \cdots, \theta_{a_l b_l}^l \\ \theta_{c_p d_p}^p, \theta_{c_{p+1} d_{p+1}}^{p+1}, \cdots, \theta_{c_m d_m}^m \\ \cdots\cdots \\ \theta_{x_q y_q}^q, \theta_{x_{q+1} y_{q+1}}^{q+1}, \cdots, \theta_{x_n y_n}^n \end{Bmatrix}$$

Assume all μ^k are found based on $\theta_{a_j b_j}^j$, according to Step 3-1, $\theta_{a_j b_j}^j$ is the minimum value in the Ω_k, which also is the minimum value in the Ω_k^* and the kth minimum value of $\theta_{a_i b_i}^i$, thence all short feature parameters bigger than $\theta_{a_j b_j}^j$ of the terminal node v_t are showed in a Ω_{k+1}^* that

$$\Omega_{k+1}^* = \begin{Bmatrix} \theta_{a_{j+1} b_{j+1}}^{j+1}, \theta_{a_{j+2} b_{j+2}}^{j+2}, \cdots, \theta_{a_l b_l}^l \\ \theta_{c_p d_p}^p, \theta_{c_{p+1} d_{p+1}}^{p+1}, \cdots, \theta_{c_m d_m}^m \\ \cdots\cdots \\ \theta_{x_q y_q}^q, \theta_{x_{q+1} y_{q+1}}^{q+1}, \cdots, \theta_{x_n y_n}^n \end{Bmatrix}$$

The minimum value in the Ω_{k+1}^* is the $k+1$ th minimum value of $\theta_{a_i b_i}^i$, and it corresponds to the μ^{k+1}. Because $\theta_{a_{i+1} b_{i+1}}^{i+1} \geq \theta_{a_i b_i}^i$, finding the minimum value in the Ω_{k+1}^* is equivalent to finding the minimum value in $\Omega_{k+1} = \left\{ \theta_{a_j b_j}^{j+1}, \theta_{c_p d_p}^p, \cdots, \theta_{x_q y_q}^q \right\}$.

Therefore, after calculating $\theta_{a_j b_j}^j$ and μ^k by using Step 3, we form a Ω_{k+1} by using Step 5 that removing $\theta_{a_j b_j}^j$ from the Ω_k and putting $\theta_{a_{j+1} b_{j+1}}^{j+1}$ into the Ω_k.

ii) Assume the minimum value in the Ω_{k+1} is $\theta_{c_p d_p}^p$, then it corresponds to the μ^{k+1} based on 2)-i). According to the Equation (34), the length of the μ^{k+1} is

$$L\left(\mu^{k+1}\right) = \alpha_t + \theta_{c_p d_p}^p$$

According to the Equation (31),

$$\theta_{c_p d_p}^p = \theta_{c_{p-1} d_{p-1}}^{p-1} + A_{c_p d_p}^\alpha$$

Assume $\theta_{a_{p-1} b_{p-1}}^{p-1}$ corresponds to the μ^m, since the node v_{d_p} locates on a $\mu_{s \to c_{p-1}}^m$, and $\mu_{s \to c_{p-1}}^m = \mu_{s \to c_{p-1}}^{\min}$ based on the composition of the μ^m, we know that

$$\sum\nolimits_{\mu_{d_p \to c_{p-1}}^m} A_{xy}^\alpha = \sum\nolimits_{\mu_{s \to c_{p-1}}^m} A_{xy}^\alpha = \sum\nolimits_{\mu_{s \to c_{p-1}}^{\min}} A_{xy}^\alpha = 0$$

According to the Equation (33),

$$\begin{aligned} \theta_{c_p d_p}^p &= \theta_{c_{p-1} d_{p-1}}^{p-1} + A_{c_p d_p}^\alpha \\ &= \sum\nolimits_{\mu_{c_{p-1} \to t}^m} A_{xy}^\alpha + A_{c_p d_p}^\alpha \\ &= \sum\nolimits_{\mu_{d_p \to c_{p-1}}^m} A_{xy}^\alpha + \sum\nolimits_{\mu_{c_{p-1} \to t}^m} A_{xy}^\alpha + A_{c_p d_p}^\alpha \\ &= \sum\nolimits_{\mu_{d_p \to t}^m} A_{xy}^\alpha + A_{c_p d_p}^\alpha \\ &= \sum\nolimits_{\mu_{s \to c_p}^{\min}} A_{xy}^\alpha + A_{c_p d_p}^\alpha + \sum\nolimits_{\mu_{d_p \to t}^m} A_{xy}^\alpha \end{aligned}$$

thereinto $\sum\nolimits_{\mu_{s \to c_p}^{\min}} A_{xy}^\alpha = 0$. Thence the μ^{k+1} is composed by the $\mu_{s \to c_p}^{\min}$, the arc $v_{c_p} v_{d_p}$

and the $\mu^{m}_{d_p \to t}$.

Base on above analysis, we view Step 3.

All minimum $\theta^{p}_{c_p d_p}$ are found from the Ω_{k+1} by using Step 3-1, and it ensures that all μ^{k+1} can be found by using Steps 3-2~3-3.

Because $\theta^{p-1}_{a_{p-1} b_{p-1}}$ and corresponding μ^{m} are found based on the Equation (31) by using Step 3-2, the $\mu^{m}_{d_p \to t}$ can be found.

Searching the $\mu^{\min}_{s \to c_p}$ by using Step 3-3 is similar to searching the μ^{1} by using Step 1, and the only difference is that Step 3 start from the node v_{c_p}, thence the $\mu^{\min}_{s \to a_2}$ can be found by using Step 3-3.

Therefore, after forming the Ω_{k+1} by using Step 5, all μ^{k+1} can be found by using Step 3.

In summary, all the 2nd shortest paths can be found by using Steps 2~3, and all the 3rd~*k*th shortest paths can be found by using Steps 3~5 repeatedly.

According to (1) and (2), the algorithm is correct.

5.2.3. Complexity of the Algorithm

Assume there are *n* nodes and *m* arcs in a connected graph, and according to the steps of the algorithm except the step of computing the parameter α in Section 5.2.1, searching the $k+1$th shortest path on the basis of the *k*th shortest path by using the algorithm is equivalent to considering each arc and its parameter A^{α} at most once, thence its complexity is $O(m)$ and the complexity of searching the *k*th shortest path is $O(km)$.

By combining with the complexity of computing the parameter α, we know that, for the connected graph without cycle, the complexity of the algorithm in Section 5.2.1 for searching the *k*th shortest path is

$$O(m)+O(km)=O(km)$$

For the connected graph with cycles but without negative length arc, the complexity of the algorithm in Section 5.2.1 for searching the *k*th shortest path is

$$O(m+nlogn)+O(km)=\begin{cases} O(m+nlogn), & k=1 \\ O(km), & k>1 \end{cases}$$

And for the connected graph with cycles and negative length arcs, the complexity of the algorithm in Section 5.2.1 for searching the *k*th shortest path is

$$O(mn)+O(km)=\begin{cases} O(mn), & n \ge k \\ O(km), & n<k \end{cases}$$

6. An Algorithm for Researching the *k*th Longest Path

According to the algorithm and its principle in Section 5 for searching the *k*th shortest path in connected graph, we can design an algorithm for searching the *k*th longest path by using the longest path model that the parameters α' and β' of each node and the parameters A'^{α} and A'^{β} of each arc. The longest path problem in the connect graph with cycles is *NP*-hard, therefore we mainly consider the *k*th shortest path in connected graph without cycle in the section. Here we mainly design an algorithm by using the parameters β' and A'^{β}.

Similarly, in order to discourse the algorithm conveniently, we first propose concepts and calculations of some new parameters, and then discourse the algorithm in detail.

6.1. A Long Feature Parameter of the Start Node of Connected Graph

A long feature parameter of the start node v_s of graph is defined as a value of the node to an arc, and it is also named that the start node v_s has a long feature parameter to an arc.

(1) We mark the 1st rank long feature parameter of the start node v_s as θ'^1, and define its value as zero, that is

$$\theta'^1 = 0 \tag{35}$$

(2) The 2nd rank long feature parameter of the start node v_s.

If there is a path composed by arcs with the parameters $A'^\beta = 0$ and from the start node v_s to a node v_r, and the parameter A'^β_{rp} of an arc $v_r v_p$ is not zero in hinder adjacent arcs of the node v_r, then the start node v_s has the 2nd rank long feature parameter to the arc $v_r v_p$. We mark the long feature parameter as θ'^2_{rp}, and define its value as

$$\theta'^2_{rp} = \theta'^1 + A'^\beta_{rp} = A'^\beta_{rp} \tag{36}$$

(3) The kth rank long feature parameter of the start node v_s.

Assume the start node v_s has the $k-1$th rank long feature parameter θ'^{k-1}_{ef} to arc $v_e v_f$, and $k \geq 2$, if there is a path composed by arcs with the parameters $A'^\beta = 0$ and from a node v_f to v_g, and the parameter A'^β_{gh} of an arc $v_g v_h$ is not zero in hinder adjacent arcs of the node v_g, then the start node v_s has the kth rank long feature parameter to the arc $v_g v_h$. We mark the long feature parameter as θ'^k_{gh}, and define its value as

$$\theta'^k_{gh} = \theta'^{k-1}_{gh} + A'^\beta_{gh} \tag{37}$$

6.2. Description of the Algorithm

For finding the path with length λ in a connected graph, we design an algorithm for searching the *k*th longest path.

Step 1. Search the 1st longest path μ^1.

Step 1-1. Compute the parameter β_i' of each node v_i and the parameter $A_{ij}'^{\beta}$ of each arc $v_i v_j$.

Step 1-2. Find connected arcs with the parameters $A'^{\beta}=0$ backward from the start node v_s to the terminal node v_t to get $\mu'^1=\mu^{\max}$ with the length $L\left(\mu'^1\right)=\beta_t'$. This step cannot be stopped repeating until no other new μ'^1.

Step 2. Form long feature sets Ω_1' and Ω_2'.

Step 2-1. Form a $\Omega_1'=\varnothing$, then compute the parameters $A_{a_2b_2}'^{\beta}$ of all hinder adjacent arcs $v_{a_2}v_{b_2}$ of each node on the μ'^1, and get $\theta_{a_2b_2}'^2=A_{a_2b_2}'^{\beta}>0$.

Step 2-2. Form a Ω_2' by putting $\theta_{a_2b_2}'^2$ into the Ω_1'.

Step 3. Search the *i*th ($i\geq 2$) longest path μ'^i.

Step 3-1. Find all minimum values $\theta_{a_jb_j}'^j$ from the Ω_i'.

Step 3-2. According to the Equation (37) (the Equation (36) if $j=2$), find $\theta_{a_{j-1}b_{j-1}}'^{j-1}$ (assume it corresponds to a μ'^m) and an arc $v_{a_j}v_{b_j}$ with the parameter $A_{a_jb_j}'^{\beta}>0$.

Step 3-3. Find connected arcs with the parameters $A'^{\beta}=0$ backward from the node v_{b_j} to the terminal node v_t to get a $\mu_{b_j\to t}^{\max}$. μ'^i is composed by the $\mu_{b_j\to t}^{\max}$, the arc $v_{a_j}v_{b_j}$ and the $\mu_{s\to a_j}'^m$, and its length is

$$L\left(\mu'^i\right)=\beta_t'-\theta_{a_jb_j}'^j \tag{38}$$

Step 4. Check.
If $L\left(\mu'^i\right)\leq\lambda$, stop; and if $L\left(\mu'^i\right)>\lambda$, turn to Step 5.

Step 5. Form a long feature set Ω_{i+1}'.

Step 5-1. Find all hinder adjacent arcs $v_{a_{j+1}}v_{b_{j+1}}$ with the parameters $A_{a_{j+1}b_{j+1}}'^{\beta}>0$ of each node on the $\mu_{b_j\to t}'^{\max}$, and then compute $\theta_{a_{j+1}b_{j+1}}'^{j+1}$ by using the Equation (37),

$$\theta_{a_{j+1}b_{j+1}}'^{j+1}=\theta_{a_jb_j}'^j-A_{a_{j+1}b_{j+1}}'^{\beta}$$

Step 5-2. Form a Ω_{i+1}' by removing $\theta_{a_jb_j}'^j$ from the Ω_i' and putting $\theta_{a_{j+1}b_{j+1}}'^{j+1}$ into the Ω_i', then turn to Step 3.

Similarly note that the 1st longest path $\mu'^1=\mu^{\max}$ must be the longest path without cycle in the graph.

The analysis on correctness of the algorithm is similar to the analysis on correctness of the algorithm of "searching the *k*th shortest path" in Section 5.2.2. And for the complexity of the algorithm, similarly, except computing the parameter β' of each node, the complexity of searching the *k*th shortest path is $O(km)$. But by combining with the complexity of computing the parameter β', because there are differences between computing the parameters β' and α according to Section 2, the total complexity of the algorithm is also a little different to the complexity of the algorithm in Section 5.2.1. For the connected graph without cycle, the complexity of the algorithm for searching the k^{th} longest path is

$$O(m)+O(km)=O(km)$$

And for the connected graph with cycles, the complexity of the algorithm for searching the k^{th} longest path is

$$O(mn)+O(km)=\begin{cases} O(mn), & n \geq k \\ O(km), & n < k \end{cases}$$

Similarly, we also can search the k^{th} longest path in a connected graph by using the parameter α_i' of each node v_i and the parameter $A_{ij}'^{\alpha}$ of each arc $v_i v_j$, and the detail steps is similar to above algorithm.

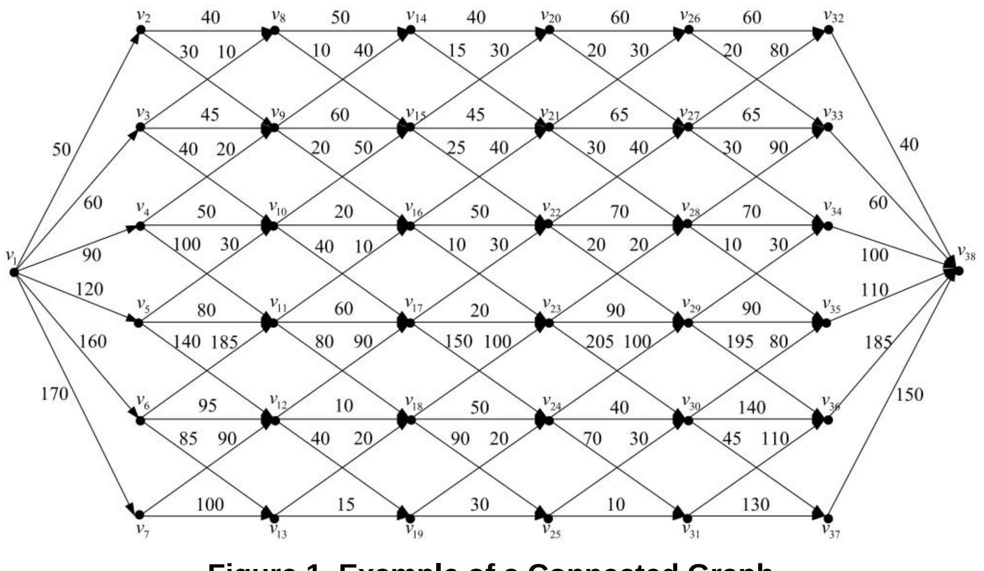
7. Illustration

Find all paths with lengths 500 from the graph in Figure 1.

Because the graph is complex, it is very difficult to research paths directly, and we adopt the ideal that first simplifying the graph and then searching the required paths in the simple graph.

(1) Simplify the connected graph.

The graph has no cycle, therefore we could simplify the problem based on the path model in Section 2.1.


v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14} v_{15} v_{16} v_{17} v_{18} v_{19} v_{20} v_{21} v_{22} v_{23} v_{24} v_{25} v_{26} v_{27} v_{28} v_{29} v_{30} v_{31} v_{32} v_{33} v_{34} v_{35} v_{36} v_{37} v_{38}
50 60 90 120 160 170
40 50 40 60 60
30 10 10 40 15 30 20 30 20 80
45 60 45 65 65
40 20 20 50 25 40 30 40 30 90
50 20 50 70 70
100 30 40 10 10 30 20 20 10 30
80 60 20 90 90
140 185 80 90 150 100 205 100 195 80
95 10 50 40 140
85 90 40 20 90 20 70 30 45 110
100 15 30 10 130
40 60 100 110 185 150
Figure 1. Example of a Connected Graph

Step 1. Mark the graph in Figure 1 as *G*. Because there is no cycle in the *G*, we can use the algorithm in Section 2.1.1 to compute the parameters α_i and β_i of each node v_i in the *G*, as in Figure 2.

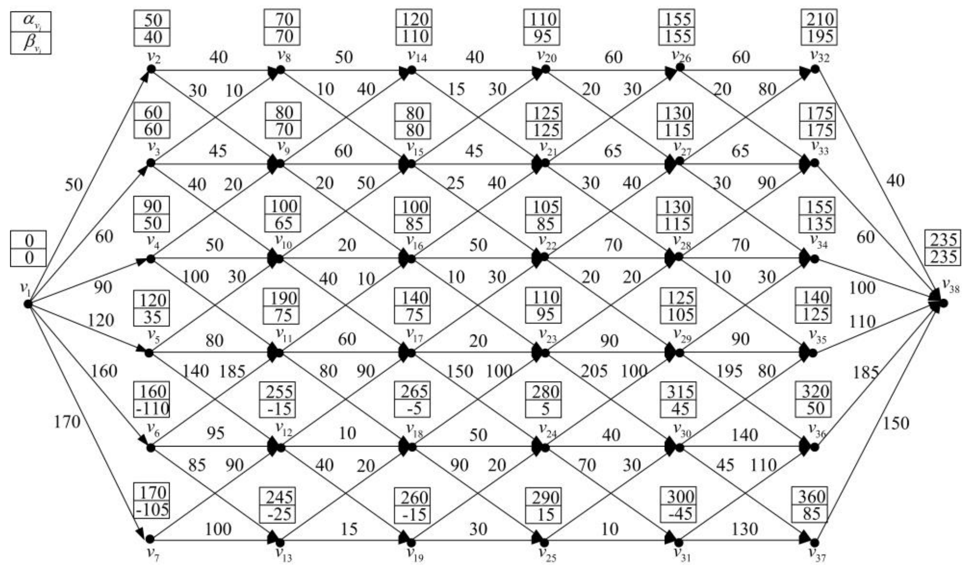


Figure 2. Graph *G*

Compute the parameter N_i of each node v_i in the *G*, and remove nodes v_6, v_7, v_{12}, v_{13}, v_{18}, v_{19}, v_{24}, v_{25}, v_{30}, v_{31}, v_{36} and v_{37} with $N_k > \lambda - \alpha_{38} = 500 - 235 = 265$, and also remove their adjacent arcs. Then we obtain a graph G_1, as in Figure 3.

Step 2. Compute the parameter $A_{ij}^{\alpha\beta}$ of each arc $v_i v_j$ in the G_1. Because there is no arc with $A_{ij}^{\alpha\beta} > \lambda - \alpha_{38} = 265$, we obtain a graph $G_2 = G_1$.

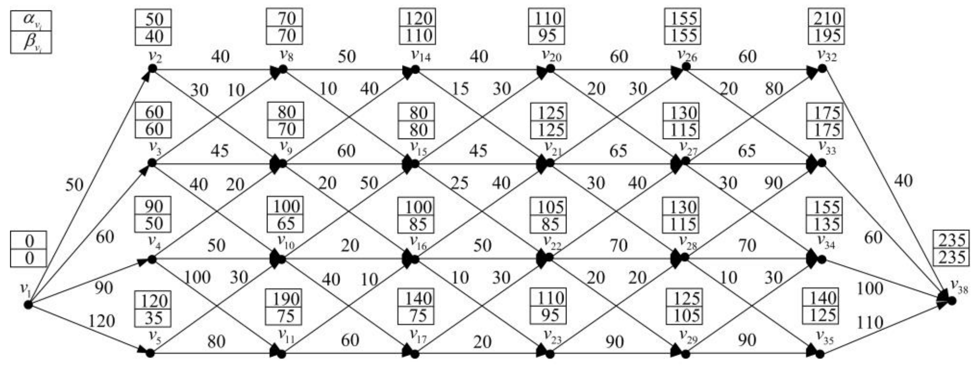


Figure 3. Graph G_1

Step 3. Compute the parameters α_i' and β_i' of each node v_i in the G_2 by using the algorithm in Section 2.1.2, as in Figure 4.

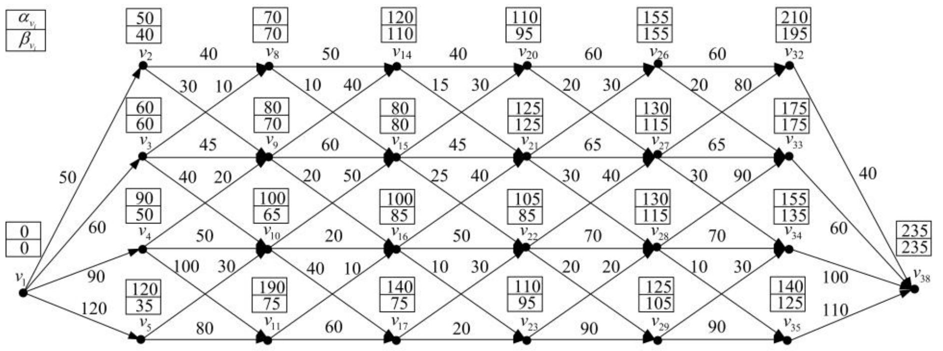


Figure 4. Graph G_2

Compute the parameter N'_i of each node in the G_2, and remove nodes v_2, v_3, v_8, v_9, v_{14}, v_{15}, v_{20}, v_{21}, v_{26}, v_{27} and v_{32} with $N'_k > \alpha'_{38} - \lambda = 570 - 500 = 70$, and remove their adjacent arcs. Then we obtain a graph G_3, as in Figure 5.

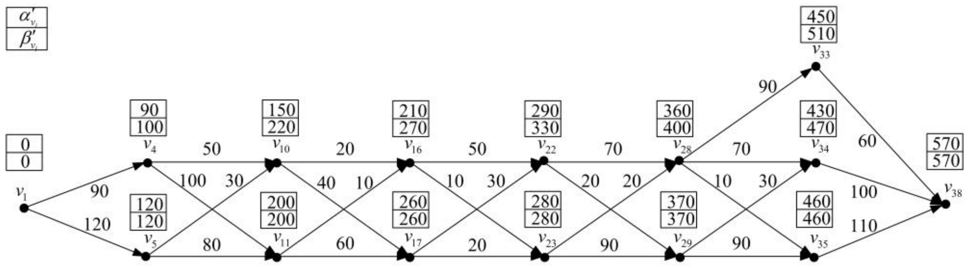


Figure 5. Graph G_3

Step 4. Compute the parameter $A'^{\alpha\beta}_{ij}$ of each arc $v_i v_j$ in the G_3, and remove arcs $v_4 v_{10}$, $v_{10} v_{16}$, $v_{23} v_{28}$ and $v_{28} v_{35}$ with $A'^{\alpha\beta}_{ij} > \alpha'_{38} - \lambda = 70$. Then we obtain a graph G_4, as in Figure 6.

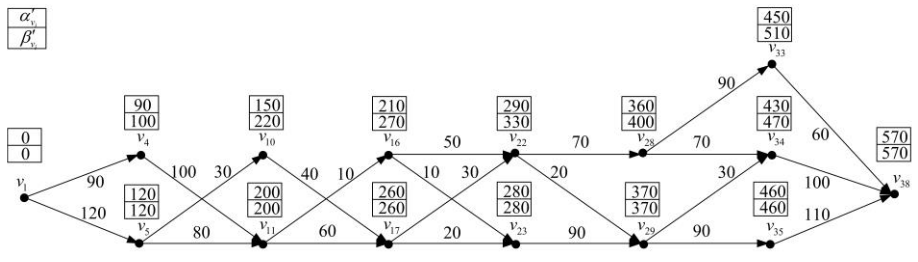


Figure 6. Graph G_4

All paths with the lengths 500 in Figure 1 are also in Figure 6, thence we only need to search these paths in Figure 6. Obviously, Figure 6 is much simpler than Figure 1.

(2) Search paths with the lengths 500.

Compute the parameter A'^{β}_{ij} of each arc $v_i v_j$ by using the parameter β'_i of each node v_i in Figure 6, and then search the kth longest path by using the algorithm in Section 6.2.

1) Search the 1st longest path μ'^1.

Find connected arcs with the parameters $A'^{\beta} = 0$ backward from the start node v_1 to the terminal node v_{38} to get a $\mu^{\max}$. Let $\mu'^{1} = \mu^{\max}$ that

$$\mu'^{1} = \mu^{\max} = v_1 v_5 v_{11} v_{17} v_{23} v_{29} v_{35} v_{38}$$

and its length is

$$L\left(\mu'^{1}\right) = \beta'_{38} = \beta'_{38} + \theta'^{1} = 570$$

Because $L\left(\mu'^{1}\right) = 570 > \lambda = 500$, continue to search the 2nd longest path μ'^{2}.

2) Search the 2nd longest path μ'^{2}.

i) Form a $\Omega'_1 = \varnothing$.

ii) Because $\theta'^{1} = 0$, in Figure 6, the parameters $A'^{\beta} > 0$ of hinder adjacent arcs of each node on the μ'^{1} is the 2nd rank long feature parameters of the start node v_1. Form a Ω'_2 by putting these parameters into the Ω'_1 that

$$\Omega'_2 = \left\{\theta'^{2}_{5,10} = 70,\ \theta'^{2}_{11,16} = 60,\ \theta'^{2}_{17,22} = 40,\ \theta'^{2}_{29,34} = 70\right\}$$

iii) Find the minimum value $\theta'^{2}_{17,22} = 40$ from the Ω'_2.

iv) According to the Equation (36), find θ'^{1} which corresponds to $\theta'^{2}_{17,22}$ and μ^{1}.

v) Find connected arcs with the parameters $A'^{\beta} = 0$ backward from the node v_{22} to the terminal node v_{38} to get a $\mu^{\max}_{22\to 38} = v_{22} v_{28} v_{34} v_{38}$. The μ'^{2} is composed by the $\mu^{\max}_{22\to 38}$, the arc $v_{17} v_{22}$ and the $\mu'^{1}_{1\to 17}$, that is

$$\mu'_2 = v_1 v_5 v_{11} v_{17} v_{22} v_{28} v_{34} v_{38}$$

and its length is

$$L\left(\mu'^{2}\right) = \beta'_{38} - \theta'^{2}_{17,22} = 530$$

Because $L\left(\mu'^{2}\right) = 530 > \lambda = 500$, continue to search the 3rd longest path μ'^{3}.

3) Search the 3rd longest path μ'^{3}.

i) In Figure 6, find a hinder adjacent arc $v_{28} v_{33}$ with the parameter $A'^{\beta} > 0$ of each node on the $\mu^{\max}_{22\to 38} = v_{22} v_{28} v_{34} v_{38}$, and compute by using the Equation (37) that

$$\theta'^{3}_{28,33} = \theta'^{2}_{17,22} + A'^{\beta}_{28,33} = 40 + 20 = 60 \tag{39}$$

ii) Form a Ω'_3 by removing $\theta'^{2}_{17,22}$ from the Ω'_2 and putting $\theta'^{3}_{28,33}$ and $\theta^{3}_{4,8}$ into the Ω'_2.

$$\Omega'_3 = \left\{\theta'^{2}_{5,10} = 70,\ \theta'^{2}_{11,16} = 60,\ \theta'^{2}_{29,34} = 70,\ \theta'^{3}_{28,33} = 60\right\}$$

iii) Find the minimum values $\theta'^{2}_{11,16} = 60$ and $\theta'^{3}_{28,33} = 60$ from the Ω'_3.

iv) According to the Equation (36), find θ'^{1} which corresponds to $\theta'^{2}_{11,16}$ and μ'^{1}; and according to the Equation (39), find $\theta'^{2}_{17,22}$ which corresponds to $\theta'^{3}_{28,33}$ and μ'^{2}.

v) For $\theta'^{2}_{11,16}$, find connected arcs with the parameters $A'^{\beta} = 0$ backward from the node v_{16} to the terminal node v_{38} to get a $\mu^{\max}_{16\to 38} = v_{16} v_{23} v_{29} v_{35} v_{38}$. The μ'^{3} is composed by the $\mu^{\max}_{16\to 38}$, the arc $v_{11} v_{16}$ and the $\mu'^{1}_{1\to 11}$, that is

$$\mu'^{3} = v_1 v_5 v_{11} v_{16} v_{23} v_{29} v_{35} v_{38}$$

and its length is

$$L\left(\mu'^{3}\right) = \beta'_{38} - \theta'^{2}_{11,16} = 510$$

For $\theta'^{3}_{28,33}$, find connected arcs with the parameters $A'^{\beta}=0$ backward from the node v_{33} to the terminal node v_{38} to get a $\mu^{\max}_{33\to 38}=v_{33}v_{38}$. The μ'^{3} is composed by the $\mu^{\max}_{33\to 38}$, the arc $v_{28}v_{33}$ and the $\mu'^{2}_{1\to 28}$, that is

$$\mu'^{3}=v_1v_5v_{11}v_{17}v_{22}v_{28}v_{33}v_{38}$$

and its length is

$$L\left(\mu'^{3}\right)=\beta'_{38}-\theta'^{3}_{28,33}=510$$

Because $L\left(\mu'^{3}\right)=510>\lambda=500$, continue to search the 4th longest path μ'^{4}.

4) Similarly, we can find the 4th longest path μ'^{4}, that are

$$\mu'^{4}=v_1v_5v_{10}v_{17}v_{23}v_{29}v_{35}v_{38}$$

$$\mu'^{4}=v_1v_5v_{11}v_{17}v_{23}v_{29}v_{34}v_{38}$$

$$\mu'^{4}=v_1v_5v_{11}v_{16}v_{22}v_{28}v_{34}v_{38}$$

And their lengths are $L\left(\mu'^{4}\right)=500=\lambda$. Thence, there are 3 paths with lengths 500 in Figure 1 that

$$\mu=v_1v_5v_{10}v_{17}v_{23}v_{29}v_{35}v_{38}$$

$$\mu=v_1v_5v_{11}v_{17}v_{23}v_{29}v_{34}v_{38}$$

$$\mu=v_1v_5v_{11}v_{16}v_{22}v_{28}v_{34}v_{38}$$

Therefore, after simplifying Figure 1 to Figure 6, all paths with required the lengths 500 in Figure 1 can be obtained only by searching the 4th longest path in Figure 6.

8. Conclusion

As a core problem of the graph theory, the path problem is very important in theory and practice. The shortest path problem is the most classic issue. But both of the shortest and longest path problems are only special path problems, and there are many extended problems of them in practice, such as the problem of searching path with required length, and so on. Comparing with the shortest path problem and the longest path problem, these path problems under general sense can be applied more widely, but they are also more difficult. Especially in current and future circumstances, problems in practice will have much larger scale and be much more complex. The path problem is also no exception, when its scale is very large and the structure of a graph is very complex, it is difficult to resolve effectively by using any algorithm, let alone those *NP*-hard problems. Thence the main bottlenecks of solving the path problem effectively in practice are the large-scale, high degree of difficulty and the resulting huge computation.

For overcoming the bottlenecks, we apply the idea of simplification to shrink scale of problems under the precondition of not affecting final results, which can greatly reduce the corresponding computation of using any algorithm to solve the problem. Aiming at the path problem with required length in connected graph, the process of “first simplifying and then solving” is used in this paper to solve the problem effectively, that is, design a simple algorithm to simplify the problem and further design simple an polynomial algorithm to search the *k*th shortest or longest paths in a connected graph.

For designing simple algorithms to simplify and solve the path problem conveniently, the shortest path models and the longest path models are first founded in this paper. On one hand, simplification for the path problem with required length in connected graph is actually to remove unnecessary paths. Although the unnecessary paths is much more than the required paths, removing the former is much easier than searching the latter by using the path models which are founded in this paper, thence the path problem can be simplified by using a very simple algorithm. On the other hand, for searching the path with required length in connected graph, the complex relationships between paths in the graph can be represented by using simple parameters of the path models, thence if substituting paths and their lengths by these parameters, the paths with required lengths can be found by using a simple algorithm to search the k^{th} shortest or longest paths.

Acknowledgments

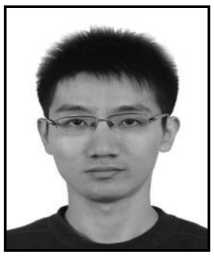
The authors would like to acknowledge the Natural Science Foundation of Jiangxi Provincial Department of Science and Technology in China (No. 20151BAB211015), and the Humanities and Social Science Foundation for Universities affiliated of Jiangxi in China (No. GL1591). The authors are grateful to the anonymous referee for a careful scrutiny of details and for comments that helped improve this paper.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, “Introduction to Algorithms (3rd Edition)”, The MIT Press, Cambridge, **(2009)**.
[2] X. M. Li, J. X. Qi and Z. X. Su, “Free float theorem and algorithm of seeking the k-th order critical path”, Journal of Management Science in China, vol. 12, no. 2, **(2009)**, pp. 98-104.
[3] J. X. Qi, L. H. Zhang and X. M. Li, “Properties and Applications of Time floats in Network Planning Management”, Science Press, Inc., Beijing, **(2009)**.
[4] Z. X. Su, J. X. Qi, and H. Y. Wei, “Solving the kth shortest path problem in directional connected graphs without loops”, Journal of Systems & Management.
[5] K. I. Kawarabayashi and Y. Kobayashi, “A linear time algorithm for the induced disjoint paths problem in planar graphs”, Journal of Computer and System Sciences, vol. 78, no. 2, **(2012)**, pp. 670-680.
[6] M. A. Bolívar, L. Lozano and A. L. Medaglia, “Acceleration strategies for the weight constrained shortest path problem with replenishment”, Optimization Letters, vol. 8, no. 8, **(2014)**, pp. 2155-2172.
[7] S. Mokarami and S. Mehdi Hashemi, “Constrained shortest path with uncertain transit times”, Journal of Global Optimization, vol. 63, no. 1, **(2015)**, pp. 149-163.
[8] J. Fiala, M. Kaminski, B. Lidicky and D. Paulusma, “The k-in-a-path problem for claw-free graphs”, Algorithmica, vol. 62, no. 1-2, **(2012)**, pp. 499-519.
[9] H. L. Bodlaender, B. M. P. Jansen and S. Kratsch, “Kernel bounds for path and cycle problems”, Theoretical Computer Science, vol. 511, **(2013)**, pp. 117-136.
[10] B. L. Fox, “Calculating kth shortest paths”, INFOR—Canad. J. Operational Res. and Information Processing, vol. 11, **(1973)**, pp. 66-70.
[11] B. L. Fox, “More on kth shortest path”, Communications of ACM, vol. 18, no. 5, **(1975)**, pp. 279-279.
[12] B. L. Fox, “K-th shortest paths and applications to probabilistic networks”, Operations Research, vol. 23, **(1975)**, pp. B263.
[13] B. András and K. Péter, “Determining the kth shortest path by the matrix method”, Szigma---Mat.-Közgazdasági Folyóirat, vol. 10, no. 1-2, **(1977)**, pp. 61-67.
[14] I. Lari, F. Ricca and A. Scozzari, “Comparing different metaheuristic approaches for the median path problem with bounded length”, European Journal of Operational Research, vol. 190, no. 3, **(2008)**, pp. 587- 597.
[15] S. K. Kim, “Linear-time algorithm for the length-constrained heaviest path problem in a tree with uniform edge lengths”, IEICE Transactions on Information and Systems, vol. E96-D(3), **(2013)**, pp. 498-501.
[16] B. Y. Wu, “A simpler and more efficient algorithm for the next-to-shortest path problem”, Algorithmica, vol. 65, **(2013)**, pp 467-479.
[17] L. Wang, “Research on an improved quantum particle swarm optimization and its application”, International Journal of Multimedia and Ubiquitous Engineering, vol. 11, no. 2, **(2016)**, pp. 121-131.
[18] D. X. Feng, “Research on track and field training system based on network technology”, International Journal of Multimedia and Ubiquitous Engineering, vol. 11, no. 1, **(2016)**, pp. 407-416.
[19] Y. Lirov, “K-th shortest collision-free path planning”, Applied Mathematics Letters, vol. 1 no. 1, **(1988)**, pp. 61-64.
[20] M. Hiroshi, “A new kth-shortest path algorithm”, IEICE Trans. Inf. & Syst., vol. E76–D(3), **(1993)**, pp. 388-389.
[21] Y. L. Chen and W. Tang, “Finding the Kth shortest path in a time-scheduling network”, Naval Research Logistics, vol. 52, **(2005)**, pp. 93-102.
[22] J. Li, S. F. Liu, Y. Y. Ren, Y. B. Jia and H. Y. Shang, “An improved Bellman algorithm for the kth shortest path problem”, Mathematics in Practice and Theory, vol. 36, no. 1, **(2006)**, pp. 215-219.
[23] S. Gao, F. Liu and Y. Y. Duan, “A Kth shortest path algorithm implemented with bi-directional search”,

Geomatics and Information Science of Wuhan University, vol. 33, no. 4, **(2008)**, pp. 418-421.
[24] M. M. B. Pascoal and A. Sedeno-Noda, "Enumerating k best paths in length order in DAGs", European Journal of Operational Research, vol. 221, no. 2, **(2012)**, 308-316.
[25] M. Jiang, Y. K. Chen, Y. C. Zhang, L. Chen, N. Zhang, T. Huang, Y. D. Cai and X. Y. Kong, "Identification of hepatocellular carcinoma related genes with k-th shortest paths in a protein-protein interaction network", Molecular BioSystems, vol. 9, no. 11, **(2013)**, pp. 2720-2728.
[26] G. S. Liu, Z. Qiu, H. Qu and L. P. Ji, "Computing k shortest paths using modified pulse-coupled neural network", Neurocomputing, vol. 149, no. Part C, **(2015)**, pp. 1162-1176.
[27] S. N. Antonio and A. R. Sergio, "An enhanced K-SP algorithm with pruning strategies to solve the constrained shortest path problem", Applied Mathematics and Computation, vol. 265, **(2015)**, pp. 602-618.
[28] Z. X. Su, J. X. Qi and H. Y. Wei, "Simplifying the path problem with desired bounded lengths in acyclic networks", Journal of Systems Science and Systems Engineering, vol. 24, no. 4, **(2015)**, pp. 500-519.

Authors



Zhi-xiong Su, is a lecturer at Nanchang Institute of Technology, China. Dr. Zhi-xiong Su received his bachelor's degree in industrial engineering from Tianjin University in 2005. H received his master's degree and PH.D. degrees in Technical Economic and Management Science from North China Electric Power University in 2008 and 2014, respectively. Since 2014, he has been an lecturer, in Nanchang Institute of Technology, China. His research interests include network planning technology, project scheduling, optimization, and software algorithm design. Dr. Su is a member of Chinese Society of Optimization, Overall Planning and Economical Mathematics and Operations Research Society of China.

Han-ying Wei, is a lecturer at Nanchang Institute of Technology, and an economist at Jiangxi Nuclear Power Corporation, China. M.A. Han-ying Wei received his bachelor's degree and master's degree in Technical Economic and Management Science from North China Electric Power University in 2007 and 2010, respectively. She joined Jiangxi Nuclear Power Corporation in 2010, and left the corporation and joined Nanchang Institute of Technology in 2014. Her research interests include Management Science, project scheduling, software algorithm design, and Finance.

Xue-min Yu, is a college student in the School of Government, Beijing Normal University, China. Her research interests include Mathematics Modeling and software algorithm design.