Free from a Shortest Path

Takeshi Shirabe

Department of Geoinformation and Cartography
Vienna University of Technology
1040 Vienna, Austria
Email: shirabe@geoinfo.tuwien.ac.at

1. Introduction

The classical one-to-one shortest path problem seeks a shortest sequence of arcs that connects two specified nodes in a network. The problem has a wide range of applications to geographic information technology including vehicle and pedestrian navigation systems, which aim to provide the user with a series of unambiguous route directions to any chosen destination. This abstract considers a situation where the user appreciates some freedom to take detours during the trip—for example, in response to unexpected local events—as long as the resulting path does not overly deviate from a shortest one.

Let us begin with a trivial navigation problem in the grid-like network laid out on the left-hand side of Figure 1. We are at node 1 and would like to go to node 6. Any shortest path algorithm will find path 1-2-4-6, 1-2-5-6, or 1-3-5-6 as a shortest path. Then a navigation system will arbitrarily select one from these, and generate directions for the chosen path. For instance, path 1-2-5-6 could be sequentially described as "Go north and turn right at the first intersection. Turn left at the next intersection. Stay on the current street and you will reach the destination."

Once a route is set and directions are given, the rest seems straightforward. Some are willing to obey every step of the given route instruction almost blindly, for example, because they are total strangers in the area. Others, however, may find doing so too restrictive—for various reasons: they may know the locality better than the system and cannot resist making different (presumably better) turns than instructed; they may encounter unexpected events that prevent them from staying on any fixed route; or simply they are such a capricious character and want to wander as long as we are getting to the destination.

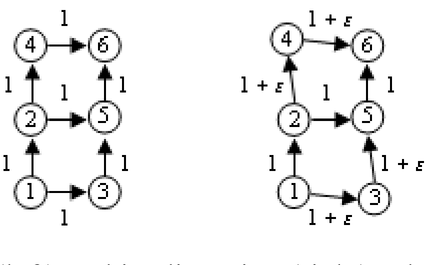

Figure 1. A grid network (left) and its distortion (right). There are three paths from node 1 to node 6 in both networks. Note that the number associated with each arc represents the length of that arc.

In the grid network illustrated above (on the left in Figure 1), we can be free of a route direction. There are only three paths from node 1 to node 6 and they all have the same length. Thus wherever we are, we just pick any direction (even randomly), and will reach the destination and realize that we have taken a shortest path.

Of course, we should not be that fortunate in a more realistic network. Even a small modification to the grid network may deprive us of the freedom to take any arc at any node. The network on the right-hand side of Figure 1 serves as a good example. It is similar to the grid network except that nodes 3 and 4 are slightly displaced in diagonal directions, which elongates arc (1,3), (2,4), (3,5), and (4,6) by a small fraction, ε. This makes path 1-2-5-6 the only shortest path from node 1 to node 6, and we will have no choice but to make every turn exactly as instructed if we want to keep the trip as short as possible. However, if the kind of freedom we had were valuable, would it be sensible to completely abandon it just to avoid a little extra travel distance (as small as 2ε)?

This abstract proposes a similar alternative to the "take any arc at any node" strategy, assuming that we can make some compromise between the desire to have more route choices during the trip and the desire to make the trip shorter. It is designed such that we do not have to select any particular path *prior to the trip*; instead, wherever we are *during the trip*, we will be offered one or more direction choices that are guaranteed to lead to a path of specified length or shorter.

2. Approach

As outlined below, the approach currently under development is to determine, at any given node, which arcs are "admissible" (i.e. allowed to take without making all resulting paths prohibitively long) as a function of the path taken prior to reaching that node.

First, recall that a solution to a shortest path problem is not just a single path but a set of shortest paths to the specified destination (or origin). There may be more than one such set depending on the underlying network, but there is always one that forms a tree (see Figure 2). From this tree, one can construct a shortest path from any node to the destination simply by tracing down the tree until the destination is reached.

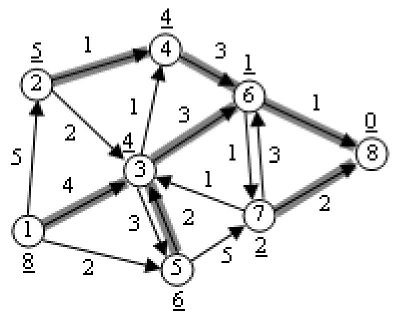

Figure 2. A tree of shortest paths (shaded) rooted at the destination (node 8). The number associated with each arc represents its length and the number associated with each node (also underlined) represents the length of a shortest path from that node to the destination.

Now let $\bar{c}(i,j)$ be $c(i,j)+d(j)-d(i)$ where c_{ij} represents the length of arc(i,j), and $d(i)$ the length of a shortest path from node *i* to the destination. The value derived this way for each arc(i,j) is generally referred to as its "reduced cost" (Ahuja et al. 1993) and can be intuitively understood as representing how much the inclusion of arc(i,j) would increase the shortest path length from node *i* to the destination.

As illustrated in Figure 3, reduced costs help us find more than one shortest path from any chosen node to the sink because any sequence of arcs with zero reduced cost is a shortest path. Paths 1-3-6-8 and 1-5-3-6-8 are two such sequences.

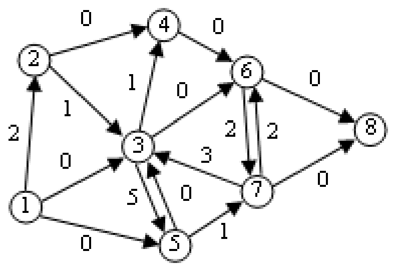

Figure 3. Reduced costs associated with the network presented in Figure 2.

It is straightforward to utilize reduced costs to determine which arcs are admissible. Suppose that a traveler reaches node *i* after traversing path W_i and finds a set of arcs $S(i)$ admissible at node *i*. As so assumed in the beginning, $S(i)$ depends on W_i and is generally expressed as:

$$S(i) = \{(i, j) \in A \mid f(i, j, W_i) = \textbf{true}\} \qquad (1)$$

where $f(i, j, W_i)$ is some function that determines whether arc (i, j) is admissible.

While I have no intention to claim any particular form of $f(i, j, W_i)$ to be the best arc admissibility function, I can provide some ideas for a good design of $f(i, j, W_i)$. To do so, it is useful to note that the traveler has the following two opposing incentives:

- He/she wants to take any arc at any node.
- He/she needs to stay on a path of acceptable length.

The dilemma can be at least partially resolved by letting the traveler take detours until it is inevitable that any resulting path will be longer than allowed, which, in turn, suggests a possible characterization of $f(i, j, W_i)$: the longer the length of path W_i is, the less likely arc (i, j) becomes admissible. An example of such $f(i, j, W_i)$ is presented as follows.

According to the definition of reduced cost, the difference between the length of path W_i and that of a shortest path is expressed as $\sum_{(k,l)\in W_i} \bar{c}(k,l)$. If $\sum_{(k,l)\in W_i} \bar{c}(k,l)$ amounts to *u* at node *i*, the traveler must take a sequence of arcs with zero reduced cost from there on until reaching the sink. If $\sum_{(k,l)\in W_i} \bar{c}(k,l)$ is still smaller than *u* at node *i*, the traveler may take another arc next unless the addition of its reduced cost to $\sum_{(k,l)\in W_i} \bar{c}(k,l)$ results in exceeding *u*. This reasoning leads to the following simple form of $f(i, j, W_i)$ that guarantees to make the resulting path equal to *u* or shorter:

$$f(i, j, W_i) = \textbf{true} \text{ if } \bar{c}(i, j) \le u - \sum_{(k,l)\in W_i} \bar{c}(k,l); \textbf{ false} \text{ otherwise} \qquad (2)$$

Figure 4 illustrates how Equation 2 determines the admissibility of arcs. Suppose $u = 5$ and that a traveler has traversed path 1-2-3 to reach node 3 (on the left). The sum of the reduced costs of the arcs comprising this path is 3 (= 2 + 1), which is greater than *u* by 2 (= 5-3). Thus the traveler can afford to take any arc with reduced cost of 2 or less. This makes arcs (3,4) and (3,6) admissible but arc (3,5) inadmissible. However, if the traveler takes path 1-3 (on the right), all of them are admissible.

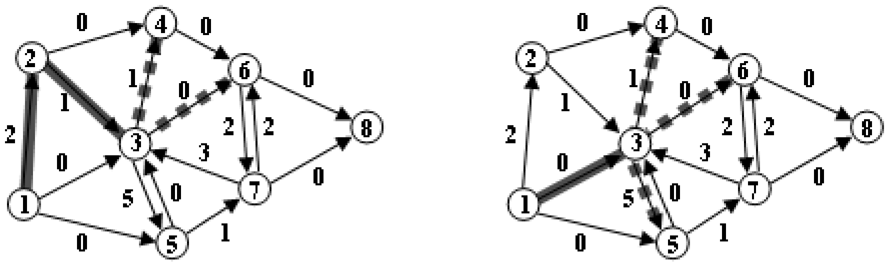

Figure 4. Evaluation of the admissibility of arcs. Assuming that *u* = 5, node 3 may have different admissible arcs (marked by dashed lines) depending on the path traversed (marked by solid lines) before that node.

Preliminary computer simulations were performed to see how the above arc-admissibility rule will navigate a random walker through a sample network. Starting at the same origin (a node at the bottom), each path drawn in Figure 5 was progressively constructed by selecting with equal probability one of the admissible arcs at each node (determined by Equation 2 with *u* equal to 20% of the shortest path length) until the destination was reached.

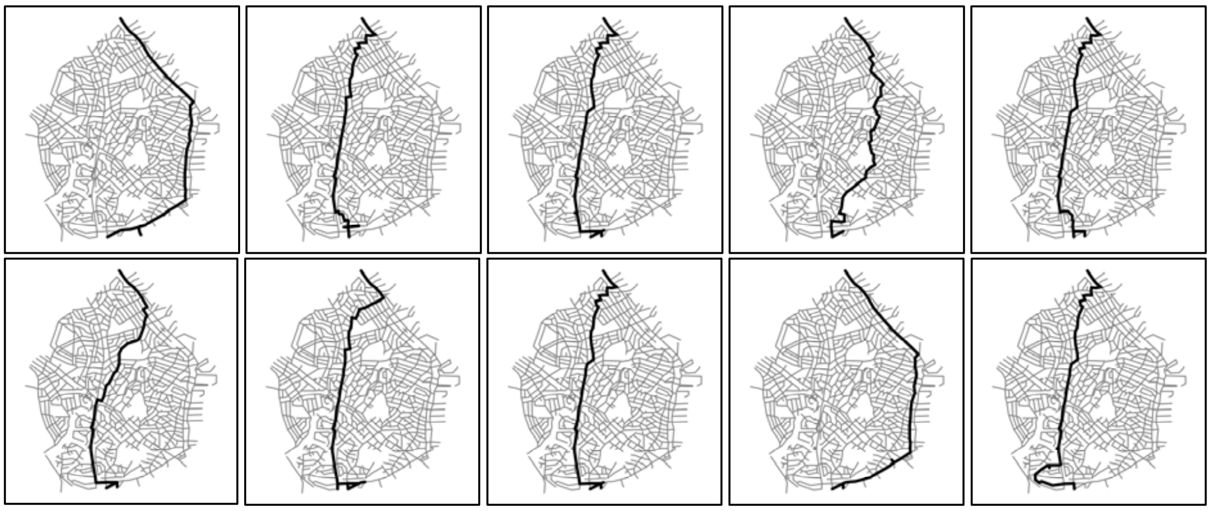

Figure 5. Paths realized according to the arc-admissibility rule expressed by Equation 2. Note that some nodes and arcs may be visited more than once in each path.

Although it was not originally intended for this purpose, another use of the proposed procedure is to generate alternative routes, from which the "best route" (with respect to some intricate and intangible criteria) is to be selected prior to the trip. An existing approach to this task is to solve "the *k* shortest paths problem," which, as its name suggests, seeks *k* shortest paths where *k* is a certain natural number (see, e.g., Hoffman and Pavley 1959, Yen 1971, Lawler 1972, Eppstein 1998).

References

Ahuja RK, Magnanti TL, and Orlin JB, 1993, *Network flows: theory, algorithms, and applications*, Englewood Cliffs, NJ: Prentice Hall.

Eppstein D, 1998, Finding the *k* shortest paths. *SIAM Journal on Computing* 28(2): 652–673.

Hoffman W. and Pavley R, 1959, A method for the solution of the Nth best path problem. *Journal of ACM* 6: 506–514.

Lawler EL, 1972, A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science* 18, 401–405.

Yen JY, 1971, Finding the K shortest loopless paths in a network. *Management Science* 17: 712–716.