**Abstract:**

It has been a general belief that

**label-setting algorithms are better choices than label-correcting algorithms**

for computing the

shortest path distance between a source node and a destination node (one-to-one)

on a network.

The reason is that

iterations in label-setting algorithms can be terminated

whenever the destination node is scanned and permanently labeled.

In this paper,

a fast-performing implementation of each type of algorithm is selected based on the results of a recent study (Zhan and Noon 1998)

and

compared for computing one-to-one shortest paths using 10 large real road networks.

The selected label-setting algorithm is

**Dijkstra’s algorithm** implemented with approximate buckets (DIKBA)

and

the selected label-correcting algorithm is

**Pallottino’s graph growth algorithm** implemented with two queues (TWO-Q).

It is concluded that, in some situations,

the TWO-Q algorithm **(label-correcting algorithm)** is a better choice for computing one-to-one shortest paths on real road networks.

**The conclusion contradicts the general belief!!**

**Keywords:**

**1. Shortest path algorithms**

**2. Networks**

**3. Transportation**

**4. Vehicle routing**

**Introduction**

The computation of shortest paths between different source and destination nodes on a network is a central and computationally-intensive task in many transportation and network analysis problems.

With the advancement of Geographic Information Systems (GIS) technology and the availability of high quality network data,

network and transportation analyses within a GIS environment have become a common practice in many areas.

These analyses often involve large road networks containing hundreds of thousands or even millions of nodes.

As a consequence, high performance shortest path algorithms are necessary for accomplishing these types of analysis tasks.

Depending on the nature of the problem, it is sometimes necessary to compute shortest paths

* from a source node to every other node (**one-to-all)** or
* from every node to every other node (**all-to-all**) on a network.

Sometimes, it is only necessary to compute shortest paths

* from a source node to a destination node (**one-to-one**) or
* from a source node to some destination nodes (**one-to-some**)

on a network.

One-to-one or one-to-some shortest paths are often needed in several areas of transportation analysis, for instance, in vehicle routing, emergency response and product delivery, just to name a few.

Empirical evaluation of shortest path algorithms has been an extensively researched topic in the literature of operations research and management science.

Among these evaluations, the most recent comprehensive evaluations are the ones conducted

* by Cherkassky et al., and
* by Zhan and Noon.

In both studies, the algorithms were categorized into two groups:

* label-setting
* label-correcting

**(Similarities)**

Both groups of algorithms are iterative and both employ the labeling method in computing one-to-all shortest paths (discussed in Section 2).

**(Differences)**

The two groups of algorithms differ,

however,

* in the ways in which they update the estimate (i.e., upper bound)

of the shortest path distance associated with

each node at each iteration

and

* in the ways in which they converge to the final optimal one-to-all shortest paths.

In **label-setting algorithms**,

the final optimal shortest path distance from the source node to the destination node is determined once the destination node is scanned and permanently labeled.

Hence,

if it is only necessary to compute a one-to-one shortest path,

then

a label-setting algorithm can be terminated as soon as the destination node is scanned,

and there is no need to exhaust all nodes on the entire network.

In contrast, a **label-correcting algorithm**

treats the shortest path distance estimates of all nodes as temporary

and

converges to the final one-to-all optimal shortest path distances at its final step

when the shortest paths from the source node to all other nodes are determined.

This characteristic of label-correcting algorithms implies

there is no difference in the computational time used for computing

a one-to-one shortest path

and

one-to-all shortest paths.

If a label-setting code and a label-correcting code had equivalent performance in computing one-to-all shortest paths, then clearly the label-setting code would be preferred in computing a one-to-one shortest path.

Such equivalence, however, was not observed in the computational study of Zhan and Noon.

**(Study by Zhan and Noon)**

* the best-performing label-setting algorithm was observed to be over 50% slower than the best-performing label-correcting algorithm

for computing

**one-to-all** shortest paths on real road networks.

* It was concluded that

Dijkstra’s algorithm implemented with approximate buckets (DIKBA) is the best-performing algorithm in the group of label-setting algorithms,

and

Pallottino’s graph growth algorithm implemented with two queues (TWO-Q) is the best-performing algorithm from the group of label-correcting algorithms.

**----**

This finding leads to several interesting questions with respect to the performance of algorithms DIKBA and TWO-Q in computing **one-to-one** shortest paths.

**(Question 1)**

For a given set of arbitrarily (e.g., randomly) chosen pairs of source and destination nodes,

when algorithms DIKBA and TWO-Q are used to compute a **one-to-one** shortest path distance,

which algorithm is faster on average?

**(Question 2)**

Because algorithm DIKBA selects the node with the least shortest distance estimate as the next node to be scanned at each iteration,

when a destination node is sufficiently close to a given source node,

algorithm DIKBA has an apparent advantage over TWO-Q.

On the other hand,

if a destination node is sufficiently far away from a given source node,

then TWO-Q should be faster in computing one-to-one shortest paths.

This observation leads to the following question:

**Would there be a threshold distance from the source node**

**within which algorithm DIKBA is highly likely to be faster, and**

**beyond which algorithm TWO-Q is highly likely to be faster**

**in computing one-to-one shortest paths?**

These two threshold distances are called the lower end and upper end threshold distances, respectively.

**In addition, would there be a threshold distance at which the speed of the two algorithms is about equal?**

This threshold distance is called the middle threshold distance.

**(Question 3)**

If there exist such relative threshold distances, what is the likelihood that DIKBA is faster than TWO-Q for some given relative threshold distances?

**(This article)**

This article aims to provide answers to the above three questions through an empirical comparison of the best-performing label-setting and label-correcting algorithms in computing one-to-one shortest paths.

The exclusive focus of this article is on the relative speed of the two algorithms.

Algorithms DIKBA and TWO-Q were tested on 10 large real road networks ranging in size from 35,793 nodes to 92,792 nodes.

**The algorithms are compared using C code implementations provided in the rigorously tested public domain codes described in Cherkassky et al. (1996).**

This study should contribute to the practical understanding of the behavior of these **label-setting** and **label-correcting** algorithms in computing **one-to-one** shortest paths on real road networks.

The results from this article should be useful to practitioners in areas such as GIS, transportation, operations research and management science.

The rest of the article is organized as follows.

* an overview of the labeling method is given along with details on the tested algorithms is discussed in Section 2
* Computational test procedures and results are discussed in Section 3
* finally a summary is given in Section 4

**----**

**4 Summary, Discussion and Recommendation**

The focus of this paper has been a comparison of two high-performance shortest path algorithms for solving shortest distance problems on large, real-world networks.

Many problems faced by practitioners (for example, time-window vehicle routing) are concerned with finding shortest time paths through networks.

These problems are typically modeled by assigning arc travel speeds (usually as a function of road class) and then computing arc travel time to be used as the arc length metric.

Both DIKBA and TWO-Q can be used for solving shortest time path problems, however, the relative speeds of the two algorithms on such problems may be different than observed in this study.

In particular, TWO-Q may be negatively affected by an increase in the number of scans for networks which lacked triangle inequality.

Problems in which arc travel times vary as a function of time (real time dispatching or IVHS applications) present additional complexities that must be taken into account when selecting a solution approach.

Based on the results presented in Section 3, the following answers to the three questions posed in Section 1 can now be summarized.

----

**(Question 1)**

**For a given set of arbitrarily (e.g., randomly) chosen pairs of source and destination nodes,**

**when algorithms DIKBA and TWO-Q are used to compute a one-to-one shortest path distance,**

**which algorithm is faster on average?**

**(Answer 1)**

For a set of randomly chosen pairs of source and destination nodes, algorithm TWO-Q is generally about 14% faster than DIKBA on average in computing a one-to-one shortest path.

----

**(Question 2)**

**Would there be a threshold distance from the source node**

**within which algorithm DIKBA is highly likely to be faster, and**

**beyond which algorithm TWO-Q is highly likely to be faster**

**in computing one-to-one shortest paths?**

These two threshold distances are called the lower end and upper end threshold distances, respectively.

**In addition, would there be a threshold distance at which the speed of the two algorithms is about equal?**

This threshold distance is called the middle threshold distance.

**(Answer 2)**

The lower end, middle and upper end threshold ratios are 0.20, 0.40 and 0.70, respectively.

These ratios suggest that the distance range from a source node to a destination node within which DIKBA is faster than TWO-Q is less than the distance range within which TWO-Q is faster than DIKBA on any given network.

----

**(Question 3)**

**If there exist such relative threshold distances, what is the likelihood that DIKBA is faster than TWO-Q for some given relative threshold distances?**

**(Answer 3)**

For some given threshold ratios, the following conclusions can be drawn:

a) When the ratio is 0.20 or less, it is highly likely (98%) that DIKBA is faster than TWO-Q for computing one-to-one shortest path distances on real road networks;

b) when the ratio reaches 0.40, there is a 50% chance for DIKBA to be faster than TWO-Q;

c) when the ratio approaches 0.60, there is a 20% chance for DIKBA to run faster than TWO-Q; and

d) when the ratio exceeds 0.70, there is a 98% chance for algorithm DIKBA to be slower than TWO-Q.

----

These results can be easily generalized to the situation of computing **one-to-some** shortest paths.

Clearly, when the shortest path distances of all destination nodes have a threshold ratio of 0.40 or less,

DIKBA would likely be faster than TWO-Q, on average.

However, when at least one of the destination nodes falls beyond a threshold ratio of 0.40, TWO-Q would likely be faster.

Overall, there is no clear advantage to use DIKBA instead of TWO-Q for computing **one-to-one** shortest paths.

The reason is that the threshold ratio at which the speed of the two algorithms is about equal is 0.40.

This middle threshold ratio favors algorithm TWO-Q when the distribution nature of network nodes in a two-dimensional space is taken into consideration.

----

The above argument can be illustrated by the following simplified example.

Suppose that there is a network covering a circular area, and the circle has a radius of one unit.

The most favorable condition for DIKBA would be that the source node is always at the center of the circle.

Furthermore, let us assume that all nodes on the network are distributed uniformly in the area, and that the distance between any pair of nodes increases linearly from the centerof the circle toward its edge.

If the possibility for any node (other than the source node) being the destination node is equal, then

* the percentage of destination nodes for which DIKBA is almost absolutely faster is only 4%

(corresponding to the lower end threshold ratio of 0.20)

* the percentage of destination nodes for which DIKBA is about as fast as TWO-Q is only 16%

(corresponding to the middle threshold ratio of 0.40)

* the percentage of destination nodesfor which TWO-Q is almost certainly faster than DIKBA is 51%

(corresponding to the upper end threshold ratio of 0.70)

----

Based on the answers to the three questions and the foregoing discussions, it is concluded that

**the TWO-Q algorithm is generally a better choice if one faces the task of computing one-to-one or one-to-some shortest path distances.**

Therefore, our recommendations are:

a) When there is no prior knowledge about the network-based shortest path distance between a source node and a destination node relative to the longest shortest path distance on a shortest pathtree rooted at the source node, use TWO-Q to compute one-to-one (some) shortest paths on anetwork; and,

b) DIKBA may be chosen if it is known that the shortest path distance from a source node to a destination node is within 40% of the longest shortest path distance on the shortest path tree rooted at the source node.

The choice of TWO-Q is perhaps most appropriate in situations where it is necessary to compute a large number of relatively long routes.

- An example would be in setting up distribution system optimization models.

An obvious choice for DIKBA would be situations in which the paths are short relative to the extent of the network.

- An example would be the case of a centralized dispatching system for emergency response.

An emergency call (destination) would be assigned to a close facility and then a path would be calculated.

Given the variety of applications and the relative closeness in performance, it is perhaps prudent to test both types of algorithms before making a selection.

----