**3 Computational Test and Results**

* Ten large real road networks were obtained for testing the algorithms.
* These 10 networks are state-level road networks compiled from U. S. Geological Surveys’ Digital Line Graphs.
* These 10 states are Alabama (AL), Florida (FL), Georgia (GA), Iowa (IA), Louisiana (LA), Minnesota (MN), Missouri (MO), Mississippi (MS), Nebraska (NE), and South Carolina (SC).
* Each of the 10 networks consists of four levels of roads, including interstate highways, principal arterials, major arterials and one additional level of more detailed roads.
* An example of the 10 networks, Mississippi, is shown in Figure 1.
* The road networks were stored and maintained in the Arc/Info1 GIS. The nodes, links and link-lengths of the networks were downloaded from Arc/Info into ASCII files.
* It was made sure that the networks were fully connected before they were downloaded to files.
* Some characteristics of the 10 networks are summarized in Table 1.
* The 10 networks cover a wide range of different sizes ranging from 35,793 nodes to 92,792 nodes.
* The arc to node ratios of the networks vary from 2.66 to 3.28.

----

The two algorithms, namely DIKBA and TWO-Q, were coded in the C programming language.

The C programs were the set of one-to-all shortest path C source codes provided byCherkassky et al. (1996) with only minor modifications.

The input networks are represented using the forward-star data structure (Ahuja et al. 1993, pp.35-37).

The programs were compiled with the GNU gcc compiler version 2.5.6 using the O4 optimization option.

The experiments were conducted on a stand-alone SUN Sparc-20 workstation, model HS21 with a 125MHZ Hypersparc processor and 64 Megabytes of RAM running under the Solaris 2.4 environment.

Because the C programs require input arc lengths to be integral, the arc lengths were multiplied by a scaling factor of 1000, and the resulting arc lengths were then truncated to integers as the input arc lengths to the C programs.

**Three sets of experiment were conducted.**

Each aims at answering one of the three questions introduced in Section 1.

**3.1 Average Performance for Computing One-to-One Shortest Paths**

In order to answer the first question, CPU times for both algorithms to compute one-to-one shortest paths must be obtained.

Because computing the one-to-one shortest path between each possible pair of source and destination nodes on each network involves enormous computation time, only five sample sets of nodes were selected from each network.

Each sample set consists of 100 randomly selected nodes. For each sample set, the following experiment was conducted.

For algorithm DIKBA, each of the 100 randomly selected nodes in each sample set on each network was treated as a source node in turn, and the remaining 99 randomly selected nodes were treated as destination nodes.

Thus, there are a total of 9900 pairs of source and destination nodes for each network in each sample set.

The CPU time for computing the one-to-one shortest path between each of the 9900 pairs of source and destination nodes was obtained on each network.

In order to obtain the CPU time, a shortest path tree rooted at each source node was constructed.

The CPU time for computing a one-to-one shortest path was obtained by recording the CPU time used to reach and permanently label a destination node from the source node.

----

After the minimum, maximum, mean CPU time, the standard deviation and the ratio between the maximum and mean CPU time for computing one-to-one shortest paths on each network were obtained for each sample set of nodes, the average of these five parameters from the five sets of sampled nodes were then computed.

These average results are summarized in Table 2.

As an example, let us take a look at the results associated with the Louisiana network.

Among the five sets of results,

* the average minimum is 0.0167 seconds
* the average maximum is 0.2933 seconds
* the average mean is 0.1177 seconds
* the average standard deviation is 0.0591 seconds
* the average maximum to mean ratio is 2.49

Similar results were obtained for algorithm TWO-Q.

The only difference is that the CPU time used for computing one-to-all shortest paths is treated as equivalent to the CPU time used for computing a one-to-one shortest path because a complete shortest path tree rooted at the source node must be constructed in both cases when using TWO-Q.

Again, for each of the five sample sets, each of the 100 randomly selected nodes was treated as a source node in turn, and the CPU time for the algorithm to construct each of the 100 shortest path trees on each network was obtained.

For the part associated with algorithm TWO-Q, each row in Table 2 represents the results associated with each of the 10 networks.

For each algorithm, the overall average of the minimum, maximum, mean CPU time, standard deviation and the maximum to mean ratio across all networks were computed.

The ratios between the values of the five parameters of the two algorithms for computing a one-to-one shortest path were also obtained (bottom of Table 2).

It can be easily seen that, on average, the label-setting algorithm DIKBA is approximately 14% slower than the label-correcting algorithm TWO-Q for computing a one-to-one shortest path for a set of pairs of randomly selected source and destination nodes.

**3.2 Lower End, Middle and Upper End Relative Threshold Distances**

In order to answer the second question, five sets of randomly sampled nodes were generated from each network again.

Each set contained 100 randomly selected nodes.

For each set of nodes, the following experiment was carried out.

First, each of the 100 nodes was treated as a source node in turn, and the CPU time for computing a shortest path tree rooted at each source node on each network was obtained for algorithm TWO-Q.

That is, 100 shortest path trees were constructed on each network using TWO-Q, and the CPU time for TWO-Q to construct each of the 100 shortest path trees was obtained for each network.

Second, algorithm DIKBA was tested against TWO-Q.

Similar to the first step, each of the 100 randomly selected nodes in each set of nodes on every network was treated as a source node in turn.

For each source node, a shortest path tree was constructed using algorithm DIKBA, and the CPU time used to permanently label each of the 99 destination nodes from the source node in question was recorded.

Also each of the 99 one-to-one shortest path distances was obtained.

**In addition, the longest shortest path distance on each shortest path tree rooted at the current source node was obtained.**

**This distance is called the longest distance.**

----

The three types of data associated with each source node are:

* the longest distance
* the list of 99 CPU times
* the list of 99 one-to-one shortest path distances

These two lists were sorted together using the CPU time as a sorting key.

Because DIKBA permanently labels the node with the minimum distance label at each iteration, the values of the elements in the list of 99 one-to-one shortest path distances should increase as the values in the list of 99 CPU times increase.

Hence, there is always a node whose associated CPU time is immediately larger than the CPU time used by algorithm TWO-Q to construct a shortest path tree rooted at the same source node.

**The shortest-path distance associated with this node is called the threshold distance and is reported here in proportion to the longest distance;** for example a threshold distance half the longest distance is reported as a threshold ratio of 0.5.

For each set of 100 randomly selected nodes, 100 such threshold ratios were obtained on each network.

----

For each set of sampled nodes on each network, the minimum, mean, median, maximum, and standard deviations of the threshold ratios were computed along with a maximum to minimum ratio.

This process was repeated for all five sets of nodes on each network, and the average of the values of the above six parameters during the five runs are tabulated in Table 3.

The overall averages of the six parameters across all networks were then computed (bottom of Table 3).

The results are relatively consistent across all networks.

The overall average of the minimum (low end) threshold ratio is 0.24.

For the sake of simplicity, the low end threshold ratio is assumed to be 0.20.

This choice is an adequate one as will be seen in the results obtained in the next subsection.

Similarly, the mean middle end and maximum upper end threshold ratios are 0.40 and 0.70.

In summary, it can be said that when the threshold ratio is 0.20 or less, DIKBA is highly likely to be faster than TWO-Q.

The speed of the two algorithms is roughly equivalent when the threshold ratio increases to 0.40.

At threshold ratios above 0.70, TWO-Q is highly likely to run faster than DIKBA.

**3.3 The Likelihood for DIKBA to be Faster Than TWO-Q for Given Threshold Ratios**

In order to answer the third question, the likelihood for DIKBA to be faster than TWO-Q across six threshold ratio values was computed.

These six threshold ratios are 0.20, 0.30, 0.40, 0.50, 0.60 and 0.70, and they were chosen at equal intervals between the lower end and upper end ratios identified in the last subsection.

For each network, five additional sets of nodes were drawn randomly, and each set also consists of 100 nodes.

For each set of nodes on every network, each node was treated as a source node in turn, and 100 shortest path trees were constructed.

The likelihood that DIKBA is faster than TWO-Q within each of the six threshold ratios was computed.

The averages of the five resulting likelihoods corresponding to particular threshold ratios from the five sets of nodes were obtained and are summarized in Table 4.

----

Based on the data shown in Table 4, one can observe that:

a) when the threshold ratio is 0.20 or less, there is at least a 98% chance that DIKBA is faster than TWO-Q

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

c) when the threshold ratio increases to 0.50, there is only a less than 20% chance for DIKBA to be faster than TWO-Q

d) when the threshold ratio approaches 0.60, there is a less than 6% chance for DIKBA to run faster than TWO-Q

e) when the threshold ratio exceeds 0.70, the chance for DIKBA to be faster than TWO-Q is less than 2%