"PATHWORLD" - A NEW SOFTWARE APPROACH TO VEHICLE ROUTING ON LARGE NETWORKS

Qifeng Wang¹, Peter S. Lindquist², Jeff Eloff³

(¹) Department of Geography and Planning, University of Toledo, 2801 W. Bancroft Street, Toledo, OH 43606
   Phone: (419) 450-2851
   Fax: (419) 530-7919
   E-mail: Qi-Feng.Wang@rockets.utoledo.edu
   (Corresponding Author)

(²) Department of Geography and Planning, University of Toledo, 2801 W. Bancroft Street, Toledo, OH 43606
   Phone: (419) 530-4287
   Fax: (419) 530-7919
   E-mail: Peter.Lindquist@utoledo.edu

(³) Department of Geography and Planning, University of Toledo, 2801 W. Bancroft Street, Toledo, OH 43606
   Phone: (419) 410-5834
   E-mail: Jeffery.Eloff@dot.state.oh.us

Submission Date: 11/08/2012

Word Count:
- Body Text = 4,100
- Abstract = 250
- Tables 1 x 250 = 250
- Figures 9 x 250 = 2,250
- Total = 6,850
ABSTRACT

The Shortest Path Problem (SPP) is a set of classic logistics problems whose main objective is to minimize total cost between origin and destination on a network or fulfill the demand of each customer on a tour while satisfying additional requirements. To this end, a wide variety of software and algorithms have been developed and applied to solve this problem. Examples range from everyday usage in Google Maps and vehicle GPS to more detailed professionally-oriented which provide users more robust functions for scheduling delivery tours and related tasks. This paper introduces a set of new variations on the SPP in a software package tentatively entitled PathWorld which is developed in Matlab environment and the software mainly deals with the shortest path problem on large data sets by applying a binary tree spanning algorithm (BTSM) to find basic route recommendations. This process was linked with a Graphical User Interface to illustrate the route recommendation on the scale of the entire North American Continent. The paper begins with an overview on the algorithms devoted to the shortest path problem and some commercialized software that incorporate these algorithms. The paper continues with a review of the main algorithm (BTSM) used in the PathWorld software for solving basic shortest path problems. Finally, the paper provides an overview of the functionalities of PathWorld, its architecture, and the main techniques for the GUI as a means to introduce a new perspective on creation of specialized logistics software. The paper concludes with a discussion on future development.

KEY WORDS: Vehicle Routing, shortest path algorithm, Matlab, GUI
INTRODUCTION

This paper introduces a new variation on vehicle routing problems in a software package tentatively entitled *PathWorld*. The *PathWorld* program provides vehicle routing solutions between O/D pairs within large networks, as well as routing between an origin and multiple destinations both in a tree and loop structure on a network.

Vehicle Routing Problems (VRP), Least Cost Routing (LCR) and Shortest Paths (SP) are a large set of interrelated problems which have garnered interest across many fields of study. This class of problems have especially useful applications in operations research and freight transportation activities, in addition to others. The VRP can be viewed as a generalized class of the “Traveling-Salesman Problem” [1]. LCR and SP deal with similar problems. They can be defined generally as the problem of identifying the path along a graph that minimizes some form of cost (such as distance, impedance, travel time, etc). Common LCR applications include telecommunications routing decisions, while SP problems are commonly applied to route planning for land based travel on a network, such as those incorporated within GPS navigation units. Traditional solutions for minimizing the distance of a route are well established, yet the computing time remains a large barrier to efficient utilization of “on-the-fly routing” on large networks. In other words, classical methods minimize distance but with a tradeoff lying in burdensome computation time.

Related vehicle routing software is divided into two types. The first type is aimed at individual users (e.g., *Google Maps* and software launched in personal GPS units). The second type takes the form of logistics or GIS software. The two major packages are *TransCAD* and *Arc Logistics*.

*TransCAD* [2] is in part, a logistics software package developed by Caliper Corporation. Overall, *TransCAD* is a comprehensive GIS/Transportation planning and analysis tool. Basic network analysis functions in *TransCAD* like "planning a trip to a number of customer sites" and "finding the shortest path between O/D pairs" is similar to those routines found in the *PathWorld* package documented in the current paper. Although *TransCAD* provides a wider variety of functions, the processing steps are more complicated in solving the problem this paper deals with. Users first need to create two input files, one is the network and the other containing depot and stops.

A second routing package is found in *ArcLogistics* [3]. *ArcLogistics* is distributed by ESRI and can fulfill similar functions to *TransCAD*; users can maintain specific data sets to carry out transportation planning. *ArcLogistics* also provides a navigation function; it is like a personal GPS integrated with the latest NAVTEQ or *Tele Atlas/TomTom* street map data for the United States. This system can run on a Windows platform installed on a vehicle.

The *PathWorld* Program introduced in this paper incorporates path finding routines within larger networks. In this case, *Pathworld* will be applied to the Oak Ridge National Laboratory (ORNL) North American highway network, which includes 80,789 highway points and 115,987 highway links. While the underlying algorithms, data structures, and GUI architecture will be described in this paper, users will also be able to access the software online as a means to solicit feedback and users’ recommendations.

COMPUTATIONAL MODIFICATIONS ON RELATED SHORTEST PATH ALGORITHMS

This review begins with the work of Chen et al. [4], and Demetrescu & Thorup [5], who developed methods of finding the shortest path distance between locations in a planar...
environment. They proposed a novel cell decomposition approach which transforms the Euclidean distance through the application of a circular path-planning wave. The search routine based on this new data structure is known as the framed-quadtree. This framed-quadtree data type is superior as it combines the accuracy of "high resolution grid-based path planning techniques" [5] with the more efficient quadtree search techniques. In addition, Demetrescu & Thorup propose a solution that deals with failing links along the shortest path route so as to re-route to the next best route [given the failure of any given link]. The Demetrescu-Thorup (DT) solution is unique as it avoids the obvious choice of recomputing over the entire possibility set (without the failed link) utilizing dynamic all pairs shortest path (APSP). The main goal of DT’s algorithm is to return the next best choice in near real time. Bauer [6] seems to propose the most efficient speedup for shortest path algorithms which applies a hierarchical and Goal-directed speedup technique, this method is 3 million times faster than Dijkstra's based on the simulation result in his paper. Schultes [7] adopted a two-phase pre-processing method to speed up shortest path algorithm. Nodes are first classified into level and then shortcuts are recursively computed bottom up. For more detailed review and comparison on the latest shortest path algorithms please refer to Daniel [8]. The paper gave an comprehensive overview on the most cutting edge research on more challenging variants of the shortest path problem. It also list the speedup techniques in chronological order and compare the performance statics based on different data sets.

In another approach, Murty [9], Yen [10], Eppstein [11], Roddity [12] and Fack [13] discuss various specifics of K-shortest path computations to solve shortest path problems. Murty provides an algorithm to rank the set of solutions for the more general assignment class of optimization problems. Yen’s 1971 algorithm provides a manner of finding K-shortest paths from origin [node] to destination [node] on a given graph. The contribution of Yen’s algorithm is that “its computational upper bound increases linearly with the value of K” [10]. It is worth noting that this proposed algorithm does not increase with the size of K due to the unique data structure employed. Fack (2011) presents a similar method of determining K-shortest loopless paths, via a deviation path algorithm. Although it is unable to determine the exact set of K-shortest paths [13], it is substantially faster than other proposed algorithms. Despite being unable to determine the exact K-shortest path, Fack’s algorithm performs quite well. Roddity [12] details an algorithm with the goal of obtaining the first approximation to find the “K-simple shortest paths connecting a pair of vertices in a weighted directed graph” [12]. His work builds on Eppstein [11] and Demetrescu & Thorup [5]. Eppstein [11] presents a number of computations to determine unrestricted (non-simple) K-shortest paths. Roddity concludes that determining simple shortest paths may be an easier problem to solve than the all-pair-shortest-paths (APAS).

**Review on MATLAB Processing**

*PathWorld* is developed within *MATLAB*, an engineering development tool that has an extensive use in digital signal processing, econometrics, and image processing. *MATLAB* was chosen for its strong capabilities in matrix processing, given that the highway network is reduced to a matrix of connections.

The *MATLAB* package has a history of spatially-referenced data processing utilities developed for specialized applications. LeSage [14] in his paper presents a *MATLAB* -based “ARC_MAT tool box”, which is a spatial data analysis software package whose source has been placed in the public domain [14] [15]. Ozgu [16] (2011) introduced a Potensoft program created
in MATLAB, which includes the basic steps of gravity and magnetic data processing, mapping and modeling. The Potensoft application is based on spatial and frequency domain filtering of gravity and magnetic data. Its accompanying GUI enables users change all the required parameters. One of the major advantages of the program is to display the input and processed maps in a preview window in order to arrive at geologically meaningful anomaly values.

Roberts [17] developed the Marine Geospatial Ecology Tools (MGET), an extensive collection of powerful, easy to use, open-source geo-processing tools that ecologists can invoke from ArcGIS without resorting to external computer programming. Internally, MGET integrates Python, R, MATLAB, and C++, bringing the power of these specialized platforms to tool developers without requiring developers to orchestrate the interoperability between them. The paper demonstrates how the software is applied to a habitat model for Atlantic spotted dolphin (Stenella frontalis); it predicts dolphin presence using a statistical model fitted with oceanographic predictor variables.

THE BTSM ALGORITHM AND DATA STRUCTURE

The BTSM (Binary Tree Sparse Matrix) algorithm a new developed algorithm for solving shortest path problem which is based on a basic data structure- binary tree. It is a fast and parallel method for pathfinding which means that multiple solutions between O/D pairs can be returned simultaneously. The advantage for the BTSM algorithm is that the computation time does not depend on the dataset volume but it does depend on the O/D pairs. Based on the complexity and the volume of the network, a BTSM algorithm can have several derivatives. The user can then select one solution according to the trade-off between cost and time. When the data set is relatively small, the basic BTSM routine can return several path solutions for the user. As networks increase in size, such as at the level of several states within the North American highway network, a bounding box method is employed to use only that portion of the network within those states in order to improve computation efficiency. As the network increases to the level of the entire continent, the bounding box method is accompanied by a series of forbidden data set constraints in order for the basic BTSM to save computing time. In addition, the PathWorld program will rely on precalculated paths for long hauls between more distance states.

Binary search trees (BST) are a useful data structure for storing data in which sorting or search operations are to be performed. This structure lends itself nicely to networks with a pre-defined topology. As the goal of the algorithm is efficient traversal of the network’s topological relationships, BST is an obvious choice for the network data storage. To illustrate the operations of the algorithm, we use a small (artificial) dataset with \( N = 10 \) nodes displayed in Figure 1. The process of the algorithm is presented in pseudo code (below). However, we must first define some terms to make the process more understandable. Denote the matrix of the branch of the binary tree \( BT_i \), origin as \( O \), destination as \( D \), the number of paths desired as \( K \), the current number of paths, \( k \), \( N \) as the number of nodes in the network and \( i \) as the number of search iterations. The binary tree takes the form of Figure 2. The number of iterations needed for the algorithm depend upon the height of the binary tree. Thus, the binary tree is expanding or shrinking depending upon the current iteration of the algorithm. At each iteration there is a matrix attached to the BST, denoted \( BT_i \), which increases in column space by one (over its previous). This arises because each iteration moves to the next level of the BST and adds each
point on this level as the next node in each respective route sequence being created. The number of rows in BT cannot be determined analytically; row space comes from the BST for the given data-set and is determined via the number of branches of the tree at this stage.

FIGURE 1 Illustrative Dataset

FIGURE 2 Binary Tree (Origin as 1 and destination as 9)

Take the data set of Figure 1 as example, we are trying to find the shortest path between point 1 and point 9. The root of the tree is the origin, denoted $Stage_0$. $Stage_1$ is a matrix containing two columns, the first of which contains the origin node; the second contains all points connected to the origin node (all points in the next level of the tree). $Stage_2$ is a matrix that expands, relative to the matrix in $Stage_1$, via the addition of one column containing all nodes
connected to the nodes in column 2. When the destination node is found in [the last column of] the current stage, this row contains a complete route. This sequence of nodes is then entered into the set of candidate routes, denoted \( \{CR\} \) and removed from the matrix. Similarly, if the last column of the current stage contains any null points (i.e. a degenerate branch was encountered) this row is also removed from the matrix. In this fashion the algorithm proceeds through the BST, updating \( \{CR\} \) with all possible paths. The shortest path must be contained in the set \( \{CR\} \), as it is the set containing all possible paths between O-D. Take the following as an example for \( \text{Stage}_i \):

\[
BT_i = \begin{pmatrix}
O & \ldots & P_i \\
O & \ldots & Q_i \\
O & \ldots & R_i \\
O & \ldots & S_i \\
\end{pmatrix}
\]

The next stage, \( \text{Stage}_{i+1} \) is given by:

\[
BT_{i+1} = \begin{pmatrix}
O & \ldots & P_i & S_{i+1} \\
O & \ldots & P_i & U_{i+1} \\
O & \ldots & Q_i & D \\
O & \ldots & R_i & X \\
O & \ldots & S_i & V_{i+1} \\
\end{pmatrix}
\]

The last column is added according to the connectivity matrix and the rows of \( BT_{i+1} \) are expanding. \( S_{i+1} \) and \( U_{i+1} \) are \( \neq D \) or \( \emptyset \) and connected with \( P_i \); and are therefore maintained in \( \text{Stage}_{i+1} \). \( BT_{i+1} \) contains D in its last entry and is therefore removed from \( BT_{i+1} \) and entered into \( \{CR\} \). Say X in \( BT_{i+1} = \emptyset \), then this row is also removed before proceeding to the next level of the binary tree, \( BT_{i+2} \) which is \( \text{Stage}_{i+2} \). Thus, it follows that \( \text{Stage}_{i+2} \) is:

\[
BT_{i+2} = \begin{pmatrix}
O & \ldots & P_i & S_{i+1} & Q_{i+2} \\
O & \ldots & P_i & U_{i+1} & T_{i+2} \\
O & \ldots & P_i & U_{i+1} & M_{i+2} \\
O & \ldots & S_i & V_{i+1} & N_{i+2} \\
\end{pmatrix}
\]

As previously mentioned, at each stage the number of columns grows by one (w.r.t the previous stage) and expands row-wise based upon the connectivity matrix/BST. This is done iteratively until the entire BST has been traversed. Given the sample binary tree presented in Fig. 2 with O=1 and D=9, this process will finish after 7 iterations to return all the possible routes and follow the procedure displayed in Fig. 3. Pseudo Code is presented below:
FIGURE 3 Expansion and Reduction of Binary Tree (i is no. of iterations)

Step 1:
Generate distance based connectivity matrix, choose origin node O, destination node D.

Step 2:
If $\sum BT \neq 0$ and $k \leq K$
- Generate a new page of BT and copy previous tree to new page.
- Find jump point to expand BT based upon distance based connectivity matrix.
- Add new point, np, to tree and prevent loop
  - if $np = D$
    * Delete row from BT and add route to Candidate Route \{CR\}
  - elseif $np \neq \emptyset$
    * delete this row
  - elseif $np \neq D$ AND $np \leftrightarrow \emptyset$
    * Add $np$ into \{R\} and expand BT
end
end
Repeat (Step 2).
FIGURE 4. Simulated network (Path between node 1 and 1999).

As a way of illustrating the efficiency of the new algorithms (as well as its accuracy) we present a comparison to Dijkstra's [18] and Floyd's [19] LCR algorithm. A simulated dataset including 2,500 nodes is created as shown in Fig 4. Corner nodes are connected to two other nodes, edges are connected to three nearest neighbor nodes and all interior nodes are connected in a N-S-E-W fashion (1st order connectivity). The total links in this simulated network includes nearly 10,000 links. The reason for creating such a simulated network is that users can visually detect if the result of the algorithm returned is the shortest path. Table 1 presents the comparison result between the proposed BTSM algorithm with Dijkstra's and Floyd's algorithm, including running time and distance for 10 different O/D pairs. All the three testing algorithms returned the exact shortest path, and BTSM method is the most efficient one based on running time. The unit for running time is in seconds and distance is measured as meters. As the efficiency and the accuracy of the new developed algorithm in large dataset is demonstrated, BTSM will be applied to PathWorld as a main routing algorithm.

<table>
<thead>
<tr>
<th>From Node</th>
<th>To Node</th>
<th>Td</th>
<th>Tf</th>
<th>Tb</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>761</td>
<td>2019</td>
<td>59</td>
<td>566</td>
<td>1.18</td>
<td>33</td>
</tr>
<tr>
<td>123</td>
<td>2018</td>
<td>54</td>
<td>535</td>
<td>1.44</td>
<td>43</td>
</tr>
<tr>
<td>17</td>
<td>1971</td>
<td>57</td>
<td>518</td>
<td>1.27</td>
<td>43</td>
</tr>
<tr>
<td>17</td>
<td>2047</td>
<td>54</td>
<td>510</td>
<td>2.48</td>
<td>70</td>
</tr>
<tr>
<td>19</td>
<td>1977</td>
<td>59</td>
<td>508</td>
<td>2.39</td>
<td>57</td>
</tr>
<tr>
<td>891</td>
<td>2491</td>
<td>57</td>
<td>509</td>
<td>1.13</td>
<td>32</td>
</tr>
<tr>
<td>245</td>
<td>1651</td>
<td>58</td>
<td>495</td>
<td>2.31</td>
<td>73</td>
</tr>
<tr>
<td>571</td>
<td>2019</td>
<td>56</td>
<td>520</td>
<td>1.22</td>
<td>31</td>
</tr>
<tr>
<td>99</td>
<td>1999</td>
<td>55</td>
<td>508</td>
<td>0.64</td>
<td>38</td>
</tr>
<tr>
<td>299</td>
<td>2081</td>
<td>57</td>
<td>544</td>
<td>1.54</td>
<td>54</td>
</tr>
</tbody>
</table>

* Td, Tf, Tb is running time in Dijkstra's, Floyd's and BTSM algorithm (time in seconds)  
* D is the path distance returned by all the three algorithms (distance measured as meter)
PathWorld doesn't use any advanced databases or specialized data formats; it reads the network data from Microsoft Excel files and ESRI shapefiles directly. Each file represents a layer in ArcGIS. The files includes highway network, points, ZIP layers and cross border facility layer.

ARCHITECTURE AND FUNCTIONALITIES OF "PATHWORLD"

PathWorld functions primarily as a pathfinding resource, where users can input origin and destination ZIP code centroids pairs or tour stops (ZIP Centroids) in order to obtain the least-cost path between origins and destinations. Users need only to input O/D ZIP pairs but not the network, as the network is already built into the GUI. This program also features a GUI which provides road recommendations for the users. Four major functions are provided by PathWorld, which include: 1) Single O/D pair vehicle routing in the U.S.A, 2) Single O/D pair vehicle routing in Canada, 3) TSP routing between a single origin and multiple destinations, and 4) cross border vehicle routing between the U.S.A. and Canada. Cross Border vehicle routing between the U.S.A. and Canada functions much like a Spatial Decision Support System, where the program will recommend a set of border crossings to the user for the user to choose from. The entire ORNL North American highway network (Figure 4) is integrated into the GUI consisting of 80,789 highway points and 115,987 highway links.

FIGURE 5 North American Highway Network Topology

LINKING NODE IDS WITH ZIP CODES

All nodes in the network have a unique ID number which are independent of the origin and destination points for routing. In the case of PathWorld, all routing is between ZIP code pairs. In PathWorld, 41,738 U.S.A. ZIP codes and 1,603 Canadian Postal Codes are integrated into the software. Pre-calculations have already been done to build an artificial link between ZIP code centroids and highway nodes; the relationship between highway nodes and ZIP code centroids are already paired up and stored in the ZIP code data layer. The coordinates of the two layers (ZIP and Highway Nodes ID) are used to find the nearest highway nodes that can be artificially attached to each selected ZIP centroids using great circle distance as seen in Figure 5. In turn, each ZIP code will attached with a node ID. Once an O/D ZIP code pair is inputted, the ZIP to
ZIP vehicle routing transfer to will take place on the transportation network based on highway node connectivity.

FIGURE 6 Linking ZIP code and Node ID Layer

The following section details the process of how the GUI transfers the ZIP to ZIP vehicle routing to nodes in the highway network during vehicle routing. Once the ZIP pair is inputted from the user, the GUI first finds the highway node pairs in the highway layer which are already paired with ZIP codes within the database; the least cost path will then be identified between those initial node pairs. While there is no actual network connections between ZIP pairs, the precalculated relationships between highway nodes and ZIP code centroids assures that ZIP to ZIP routing on the highway network will take place.

In order to make the software efficient and fast, The GUI has 3 working scenarios. These include:

1. Routing where O-D ZIP pairs lie within the same state (Intrastate routing)
2. Routing between states within a relatively small portion of the North American Network (limited inter-state routing)
3. Long-haul routing between multiple states over large portions of the network (Long-haul)

In Scenario 1 (Intrastate routing), an individual state’s network data will be read and processed by the BTSM algorithm in its entirety, resulting in a direct routing guide as well as visualization of the path found.

In Scenario 2 (limited inter-state routing), the algorithm will input the related states’ data that will be needed to find the path. For example, Figure 6 shows nine different hypothetical states containing a set of ZIP Code points. It is assumed that the highway network lies in a layer below the state boundaries. Now suppose that a selected O/D ZIP pair is contained both in state 7 and state 1, it would not be necessary to input all of the nine states’ network data. Only states 1, 4 and 7 would be needed to find the optimal path. Therefore, states 1, 4, 7 will be inputted to created a sub-network and find the path. The state ID sequence attached to each state ZIP pair will be stored in a structure data set--this structure does not store the state ID sequence according to each ZIP pair, but rather it is based on the state pair (e.g., all ZIP code centroids in state 1 and all ZIP code centroids in state 2 will have the same state ID sequence).
In Scenario 3 (long haul transportation), where the path between O/D ZIP pairs will go through several states, the algorithm will reduce the computation time by using pre-calculated paths. For example, Figure 6 shows a path from point 1 (State 7) to point 28 (State 3). Under this situation, a set of access and egress points will be set up for an established precalculated path (e.g., the middle path represented by the dashed line). The algorithm will first detect this O/D pair as a long haul path between state 3 and state 7, and the middle path for this state pair is between access point 26 and egress point 25. The algorithm will find path between 1 to 26 (access path) and 25 to 28 (egress path), then connect the middle path with access and egress path. The algorithm actually combines the pre-calculated path with access and egress path. This process is best summarized in the following pseudo code (architecture) for PathWorld in the following. The purpose of three scenarios is to read less data and use the pre-calculation to save computing time, this is why the PathWorld is very efficient and fast on such a huge transportation network.

**Pseudo Code for Architecture of PathWorld**

```
begin
    Input the Origin and Destination ZIP code find out the state of O/D ZIP pair
    If it is an intra state routing
        Input individual state highway network and nodes
        Run BTSM, find path and visualize it (BTSM is the Binary Tree Shortest Path Algorithm)
    If it is a inter-state routing
```
Read a serial of state data and create a sub network

Run BTSM, find path and visualize it

If it is a long-haul routing

Read pre-calculated middle path, find access and egress highway points

Run BTSM twice to find access path between Origin to access point and egress path between egress point and destination

Connect access path with middle path and egress path, and visualize it

end

CROSS BORDER VEHICLE ROUTING

*PathWorld* also has a cross border vehicle routing option which is a new feature compared with other vehicle routing software packages; 91 cross border facilities (CBFs) between the U.S. and Canada border are integrated into the GUI. The program uses Canadian 3-digit string postal codes (e.g., V3M), which are obviously different from the U.S. 5-digit ZIP codes. Users simply input the O/D ZIP/Postal Code pairs, and the system will pre-process the data and display O/D pair on the GUI. The user can then select a CBF from a list of 91 facilities. The user thus not only waits for the result but also is involved in the decision making process of finding the appropriate border crossing. The system will then return a whole cross border route between O/D pair.

The CBF vehicle routing actually takes place among different layers. First, the U.S. and Canadian highway networks are separated into individual units. The CBF layer is then inserted between them. *PathWorld* will then build the linkage between the two highway networks by selecting the appropriate border crossing; each CBF facility will be attached with 2 other points, one is the USA ZIP code and another is the Canadian postal code. A series of precalculated artificial links have been built into the network database that connect each CBF with their closest U.S.A. ZIP Code Centroid and Canadian postal code centroids (see Figure 5). As a result, *PathWorld* will calculate the shortest path through that Canadian postal code centroid and ZIP code centroid that are nearest the CBF between an ZIP Code O/D pair. As a result, the CBF can be connected to the entire highway network based on its nearest ZIP centroid.

An example of this process appears in Figure 7. First, the user needs to input the O/D ZIP pair as Point A and Point D in the figure. After the user specifies a CBF, Point B and point C will be found. A path between A and B, another path between C and D will be found. The program will then connect the path segments as A-B-CBF-C-D, which comprises the entire shortest path between the Canadian postal code origin to the U.S.A. ZIP destination.
Most existing logistics software products are all unimodal in terms of vehicle routing. In contrast, this method of vehicle routing in different layers as applied to cross border routing can be extended to intermodal vehicle routing such as between highway and maritime or highway and rail in a more complex transportation network.

**FIGURE 8 Cross Border Vehicle Routing**

The BTSM algorithm algorithm were carried out on the ORNL North American Highway Network (Fig. 4) which includes 80,789 highway points and 115,987 highway links. The path was consistently returned within 3-4 seconds. The times include reading the data as well as drawing the image and the path. These results were obtained on a PC Laptop with 8GB ram, Intel core-I5 2.3 GHZ CPU, Intel solid state drive running Windows 7 Professional (64-bit). Figure 8 shows a demo for the path found between California and Florida. The GUI will give the path recommendation as well as the traveling time and total distance.
CONCLUSION AND FURTHER DISCUSSION

The paper documented the development of the PathWorld Program, which combines the newly developed shortest path algorithm (BTSM method) and a GUI to display the North American Highway network. The program is a specialized logistics software created within MATLAB. The GUI can be accessed online to give path recommendations on the highway network to users. The software uses the entire North American highway network to make logistics decisions that can help the user to minimize transportation costs. The methods of linking different layers during transportation is not only applied to unimodal vehicle routing but can also be extended to intermodal vehicle routing. The run time for the GUI to return a path depends on the separation of the O/D pair but not the volume of the network. Normally the result will be returned in 3-4 seconds including read the data and a visualized result. If the problem is extended to a cross border routing between extreme northeast part of Canada and southwest part of USA, it will take only less than 10 seconds to compute the path. The configuration of the PC used for the software has a configuration of 8 GB RAM, 160 SSD, Windows 7 64 bits operation system.

Ten seconds run time may not be very feasible for the on-the-fly applications between ZIP Code centroid pairs, so pre-calculations may be necessary, perhaps among aggregates of counties within states. The shortest path between county aggregate centers to all other county aggregate centers could be pre-calculated and stored in the system. The program could then calculate the access and egress paths and connect the three segments to construct the whole path. This could further save computing time.

ACKNOWLEDGEMENTS

This project was completed with the support and partnership of the National Center for Freight and Infrastructure Research and Education (CFIRE), a Tier 1 University Transportation Center.
(UTC) funded by the US Department of Transportation (USDOT) Research and Innovative Technology Administration (RITA).
REFERENCES


