

# Analysis of the Traveling Salesman Problem with a Subset of Intermediate Cities and Dynamic Edge Weights used with Intelligent Transportation Systems

Jeffrey Miller

Department of Computer Systems Engineering  
University of Alaska, Anchorage  
jmill@uaa.alaska.edu

**Abstract – In this paper a specialized routing problem for vehicles in a transportation network that need to visit multiple destinations before returning to the starting location in the minimum time is presented. Although this problem is similar to the Traveling Salesman Problem (TSP), it differs because the edge weights can change constantly and the vehicle only needs to visit a subset of the nodes in the graph. The Dynamic Fastest Paths with Multiple Unique Destinations (DynFast-MUD) algorithm [23] provides a solution to this problem which is tested in a live environment in this study. There are currently 50 vehicles in Anchorage, Alaska that contain devices that report the speed, location, and direction to a central server through a vehicle-to-infrastructure (V2I) architecture. Using this data, the shortest route to a predefined set of destinations was compared to the path identified by the DynFast-MUD algorithm once a day for a two week period. The results show that with this relatively limited number of vehicles contributing to the dynamic changing edge weights, the DynFast-MUD algorithm always provides a route that is at least as fast as the shortest route. It is hypothesized that with more vehicles reporting speed and location data, the DynFast-MUD algorithm will produce even better results.**

## I. INTRODUCTION

One question facing many drivers frequently concerns finding the fastest path from their current location to their desired destination. Often there is not just a single destination, such as would be the case with delivery drivers or individuals making more than one stop before returning to their origin. Although many people may choose to take the shortest route through all of their intermediate destinations, this may not prove to be the fastest route based on the current, and changing, traffic conditions. Further, even if a traveler determines the fastest route through the intermediate destinations before departing, the fastest route is likely to change while the person is in the middle of his route.

Through Intelligent Transportation System (ITS) technologies, vehicles are able to transmit their speed, location, and direction data at all times to a central server through a vehicle-to-infrastructure (V2I) architecture. In a V2I network, vehicles are required to have a GPS receiver and some device, such as a cellular transmitter, capable of

sending a signal to some roadway infrastructure. The data transmitted by the vehicle includes the speed, location, and direction of the vehicle, though additional data can also be sent if desired. Hardware installed along the roadway is responsible for receiving the signal from the vehicle's transmitting device and sending it to a central server. The server will then receive the vehicle data and store it in a database for later retrieval.

The data stored by the server will be quite large, especially as more vehicles transmit their data. To understand the magnitude of the data, assume that the speed can be represented by one byte, the location by two bytes, the direction by one byte, and a timestamp by one byte. In large metropolitan areas, such as Los Angeles, there could be over one million vehicles in the freeway system surrounding the city at any given time [25]. If only 10% of the vehicles were transmitting this data every second for 24 hours a day, the amount of data the central server would be storing each year is:

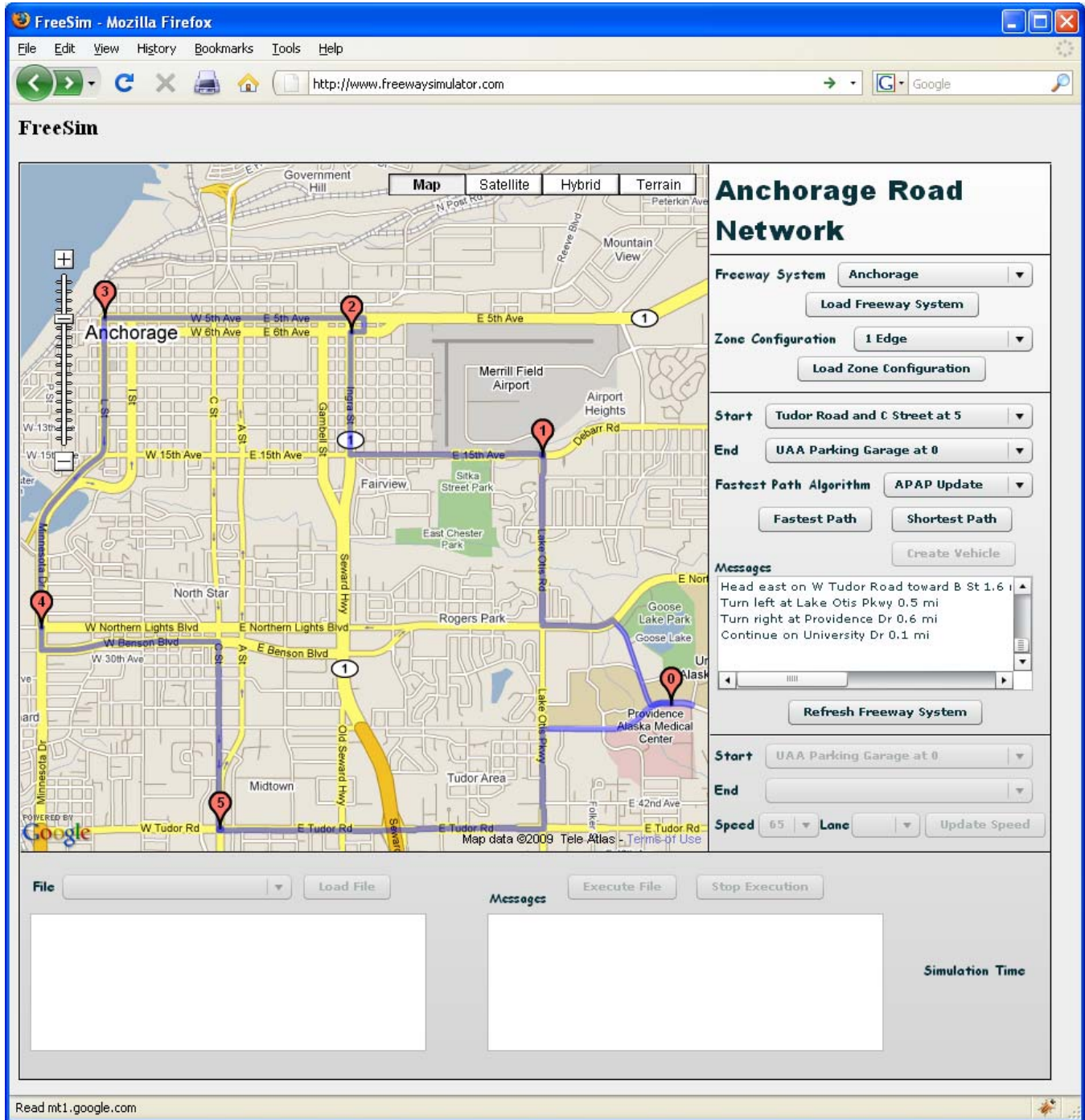
$$5 \text{ bytes/transmission} * 86400 \text{ transmission/day} * 100,000 \text{ vehicles} * 365 \text{ days/year} = 14.3\text{TB}$$

Although existing hardware is able to handle that much data, analyzing this data over time will be quite time intensive. Further, analyzing this data in real-time sufficiently to produce usable results requires the use of some creative algorithms that can operate on a large amount of data in a relatively short amount of time.

The problem addressed in this paper is as follows. Assume that a transportation network can be represented as a graph with  $n$  nodes and  $m$  edges. Each edge has a weight associated with it which is the amount of time to traverse the road segment represented by that edge. A vehicle wants to traverse  $n' \leq n$  nodes. What is the route that produces the minimum time to visit the  $n'$  nodes, keeping in mind that the weights on the edges can change in real-time while the vehicle has already begun traversing a route?

Although the solution to this problem seems similar to the Traveling Salesman Problem (TSP), there are two distinct differences. First, the weights on the edges could change constantly, so the TSP algorithm would have to be re-executed every time an edge weight changed. Second, only a subset of the nodes defined in the graph must be traversed, though the TSP assumes that all of the nodes in the graph need to be traversed. These two differences pose a unique problem known as the Specialized Traveling

FIGURE 1. FREESIM [2] SCREENSHOT WITH SHORTEST ROUTE THROUGH ALL FIVE INTERMEDIATE NODES STARTING AT NODE 0

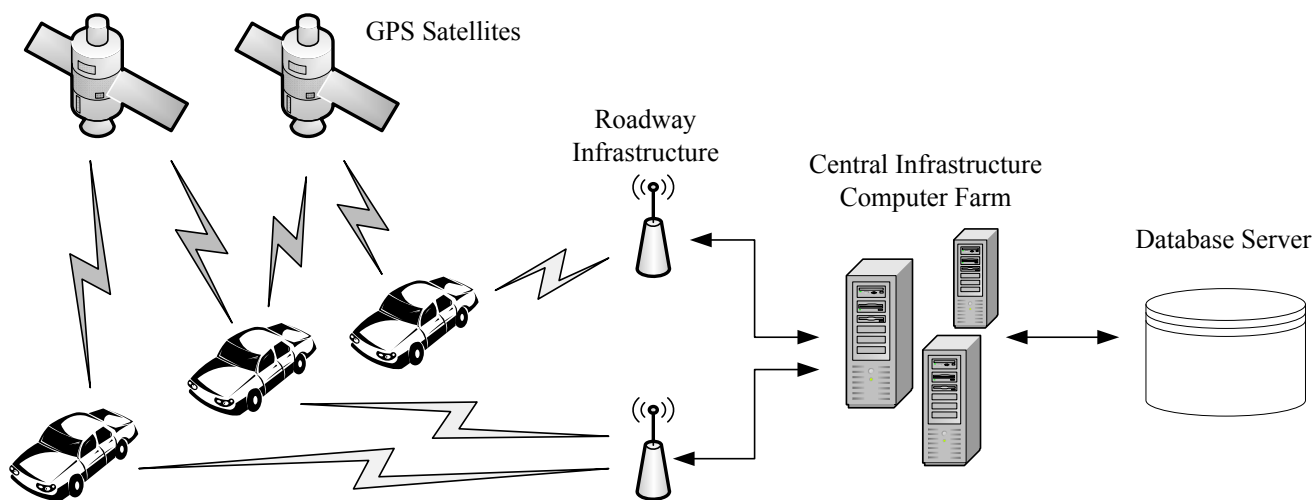


Salesman Problem with Intermediate Cities. The DynFast-MUD algorithm [23] provides an efficient solution to this problem.

In this paper, the DynFast-MUD algorithm is compared to the shortest path through a set of intermediate nodes in a transportation network. In Anchorage, Alaska, 50 vehicles are currently equipped with vehicle-tracking devices that are reporting the speed, location, and direction to a central server every 10 seconds. Of the 50 vehicles, 48 of the

vehicles had no restrictions as to where they could travel. The remaining two vehicles were required to traverse five different intersections before returning to the start node every day for two weeks, departing from the same start node at 4:30p.m. One of the two vehicles always took the shortest path through those five intersections, and the other took the fastest path as determined by the DynFast-MUD algorithm from the data received from the other 48 vehicles. If no data was available for a specific section of the

FIGURE 2. VEHICLE-TO-INFRASTRUCTURE INTELLIGENT TRANSPORTATION SYSTEM ARCHITECTURE



roadway, the speed limit on that section was used. The vehicle following the DynFast-MUD algorithm would receive updates if the fastest route changed while the vehicle was already in route. A map of the start nodes and the five intermediate nodes is provided in Figure 1.

The remainder of this paper is arranged as follows. Section II provides an overview of the related work on shortest path and dynamic path algorithms, as well as ITS routing algorithms and architectures. Section III provides an overview of the V2I implementation this study used and an analysis of the results of the DynFast-MUD algorithm. The conclusion and future work are provided in section IV.

## II. RELATED WORK

Shortest path graph algorithms have been studied for many years. Single source shortest path algorithms are used to determine the shortest path between one node and every other node in a graph. Dijkstra's [7] and Bellman-Ford's [8, 9] algorithms are popular single source shortest path algorithms. Using these algorithms, the all pairs shortest path algorithms compute the shortest path between every pair of nodes in a graph. Johnson's [10] and Floyd-Warshall's [11] algorithms are popular all pairs shortest path algorithms. Although these algorithms could all be applied to a graph with constantly-changing edge weights, they were originally designed assuming the weights of the edges were rather static. If the weight of an edge changes, the algorithm will have to be re-executed to determine if any of the shortest paths change.

Because of this inefficiency, dynamic fastest path (DynFast) algorithms were created. In [13], dynamic fastest path algorithms in general are explained, and in [14] an algorithm for solving the dynamic all pairs shortest path problem is proposed.

Bringing shortest path algorithms to the transportation world allows another optimization to be experienced. In a

transportation graph, edges are rarely added or removed completely, giving us a relatively static graph. However, the weights on the edges are often changed based on the changing traffic conditions. The All Pairs All Paths Pre-Computed class of DynFast algorithms [3] takes advantage of this unique characteristic and provides an efficient algorithm for computing fastest or shortest paths within a transportation network.

As was explained earlier, the TSP [12] is similar to the problem posed in this paper, with the difference that the TSP does not deal with changing edge weights or only visiting a subset of the nodes in the graph. The Dynamic Traveling Salesman Problem (DTSP) [4, 5] is a specialized case of the TSP that allows nodes to be inserted into the graph after the traversal has already begun. The Dynamic TSP with Time Windows is another variation on the TSP where the times at which the traveler can visit each node is defined by a specific time window [6]. Even though there are a number of special cases of the TSP, the TSP with a subset of intermediate cities and dynamic edge weights applied to a transportation network has not yet been studied with respect to ITS architectures.

One additional variation of the TSP which seemed relevant to this study is the Probabilistic TSP [1]. Jaillet presented the Probabilistic TSP, in which a random subset of cities is visited, but his solution used the solution to the traditional TSP. He then removed the nodes from the TSP solution that were not in the random subset to find the solution to the Probabilistic TSP. We discovered that the optimal solution to the subset of cities did not have to coincide with the ordering of the cities in the solution to the traditional TSP, which ruled out Jaillet's work in our specific problem.

The DynFast-MUD algorithm [23] provides a solution to the problem posed in this paper. Using the DynFast algorithms applied to ITS architectures, the DynFast-MUD algorithm accounts for constantly-changing edge weights

TABLE 1. AMOUNT OF TIME TO TRAVERSE THE SHORTEST ROUTE AND THE FASTEST ROUTE AS DETERMINED BY THE DynFast-MUD ALGORITHM ON EACH DAY

Date	Shortest Route (mm:ss)	Fastest Route as determined by DynFast-MUD algorithm (mm:ss)	DynFast-MUD Percentage Improvement over Shortest Route
March 16, 2009 – Monday	33:21	33:21	0%
March 17, 2009 – Tuesday	35:58	34:43	3.5%
March 18, 2009 – Wednesday	41:10	36:22	11.7%
March 19, 2009 – Thursday	36:30	36:30	0%
March 20, 2009 – Friday	32:47	32:47	0%
March 23, 2009 – Monday	33:52	31:55	5.8%
March 24, 2009 – Tuesday	44:23	40:12	9.5%
March 25, 2009 – Wednesday	37:49	34:31	8.8%
March 26, 2009 – Thursday	32:31	32:31	0%
March 27, 2009 – Friday	30:27	30:27	0%

and efficiently computes the fastest route (which consists of one more path than the number of intermediate nodes).

Focusing on ITS architectures, vehicle-to-vehicle (V2V) [15] and vehicle-to-infrastructure (V2I) [16] ITS architectures have been widely studied in literature. The two architectures were combined into the vehicle-to-vehicle-to-infrastructure (V2V2I) architecture in [19], and vehicular ad-hoc networks (VANETs) are becoming increasingly more popular [17]. The means by which speed is determined from gathering data discretely at loop detectors is widely used by departments of transportation and was originally proposed in [18].

For the majority of work done using speed or location data, the data was either gathered at discrete locations (from devices such as loop detectors or video cameras) or the data was simulated to be continuous. More recent projects, such as MIT’s CarTel [20] and UC Berkeley’s Mobile Millennium [21] projects, are gathering data in a distributed manner from cellular phones, though very little data has been published from these projects. Further, there are additional challenges of trying to maintain anonymity, as well as determining if the cellular device is actually located within a vehicle when it is transmitting the speed and location. A company called Airsage [22] has attempted a similar project in the Washington DC area. At the University of Alaska, Anchorage, 50 vehicles have dedicated vehicle-tracking devices installed, so there is no issue in determining which devices are communicating from vehicles. The devices maintain the privacy of the users because there is no identifying information relating the device to a vehicle or a driver.

### III. DynFast-MUD ANALYSIS

In a vehicle-to-infrastructure (V2I) architecture, the vehicles transmit their speed, location, and direction over a wireless link to a device that forwards the data to a central server. Figure 2 depicts this architecture. For this project, the vehicle-tracking hardware was the RTV5 device purchased from Live View GPS [24]. The device contains a GPS receiver and a cellular transmitter. From the GPS

coordinates, the speed and direction are interpolated then sent every 10 seconds with the latitude and longitude defining the location over a cellular link to a base station, which forwards the data to a central server hosted at the university. The data is added to a database with a timestamp for use by real-time or off-line applications.

For this study, there are 50 vehicles in Anchorage, Alaska that have the tracking devices installed. All of the vehicles frequent the University of Alaska, Anchorage, which gives rather accurate data on the roads surrounding the university. Nearly all of the vehicles depart from the university between 4:30p.m.-5:00p.m. on weekdays. Of the 50 vehicles, 48 of them were able to drive along whatever roads they wanted. The remaining two vehicles were used to analyze the DynFast-MUD algorithm [23]. From the transportation network in Figure 1, one of the vehicles was required to traverse the five intermediate nodes in numerical order, which is the shortest route. The other vehicle was required to traverse the five intermediate nodes in the order in which the DynFast-MUD algorithm determined. Both vehicles departed from the university (node 0) at 4:30p.m. each weekday for a two-week period.

In the DynFast-MUD algorithm, if there was no speed data as gathered by one of the vehicles being tracked, the speed limit of the road segment was used. Further, if the fastest route changed in the middle of the vehicle’s travel, a text message was sent to the driver’s cellular phone to alert him of the change. Although this is not the long-term solution for this problem (since it does not support the idea of anonymity), it was a short-term solution that was easily implemented. In the future, the new route will be displayed in a monitor in the vehicle, similar to that of a navigation system.

The amount of time to traverse the shortest route and the fastest route as identified by the DynFast-MUD algorithm are shown in Table 1. Since the entire transportation network in Anchorage, Alaska consists of traffic-regulated streets, the amount of time to traverse the route was dependent upon the delay due to traffic signals. If there was no delay from traffic signals or traffic, the 11.5 mile shortest route would take 23 minutes to traverse. As

can be seen in Table 1, the actual amount of time to traverse the route was always above 30 minutes, which can be attributed at least partly to the traffic signals.

The overall improvement of using the DynFast-MUD algorithm instead of solely taking the shortest route was between 0% and 11.7%. This shows that the DynFast-MUD algorithm never produced a route that ended up being slower than the shortest route. Further, the routes taken by the DynFast-MUD algorithm almost always had the intermediate nodes traversed in the same order, but the paths between the nodes were different. There were three days which had the intermediate nodes in a different order. One of the days had the vehicle traverse in the exact opposite order, and the other two days had the order of 0, 1, 3, 2, 5, 4, 0 (based on the node numbering in Figure 1). Although this path was 15.3 miles instead of 11.5 miles, these were the two days that experienced the greatest improvement over the shortest path (11.7% and 9.5%). It is important to note that the days with the greatest improvement also were the days that the shortest route took the longest amounts of time in the study, showing that as traffic worsens, the DynFast-MUD algorithm has more of an impact in decreasing the overall time to traverse the route.

#### IV. CONCLUSION

In this paper, the DynFast-MUD algorithm [23] for traversing a set of intermediate nodes in a transportation network using a live V2I architecture in Anchorage, Alaska was analyzed. Over a 10 day study period with 50 vehicles communicating using a V2I architecture, the DynFast-MUD algorithm proved to always take no more than the amount of time to traverse the shortest route. On days in which the shortest route was not the route used by the DynFast-MUD algorithm, the improvement in time was anywhere between 3.5% and 11.7%. The days of the greatest overall improvement were also the days in which the shortest route took the longest. Although it cannot be concluded that this will always be the case, it did provide an interesting analysis for the 2-week study conducted.

In the future, the benefits of the DynFast-MUD algorithm will continue to be studied, hopefully including vehicle fleets such as taxis and delivery drivers. Since it would not be feasible to have two vehicles always traversing the same set of intermediate nodes, the driving behaviors of drivers before using the DynFast-MUD algorithm and after using it will be compared to determine if there is an overall improvement, and if so, how much of an improvement. LCD screens will also be installed in the vehicles using the algorithm to provide a more user-friendly interface for communicating with the drivers.

#### V. REFERENCES

[1] Jaillet, Patrick. "A Priori Solution of a Traveling Salesman Problem in which a Random Subset of the

Customers are Visited." Operations Research, Volume 36, Number 6, November 1988.

[2] Miller, Jeffrey, Ellis Horowitz. "FreeSim – A Free Real-Time Traffic Simulator." IEEE 10<sup>th</sup> International Intelligent Transportation Systems Conference, September 2007.

[3] Miller, Jeffrey, Ellis Horowitz. "Algorithms for Real-Time Gathering and Analysis of Continuous-Flow Traffic Data." IEEE 9<sup>th</sup> International Intelligent Transportation Systems Conference, September 2006.

[4] Ghiani, Gianpaolo, Antonella Quaranta, Chefi Triki. "New Policies for the Dynamic Traveling Salesman Problem." Optimization Methods and Software, Volume 22, Issue 6, December 2007.

[5] Lu, Xiangwen, Amelia C. Regan, Sandra Irani. "The Dynamic Traveling Salesman Problem: An Examination of Alternative Heuristics." Transportation Science, 2001.

[6] Larsen, Allan, Oli B.G. Madsen, Marius M. Solomon. "The A-Priori Dynamic Traveling Salesman Problem with Time Windows." Transportation Science, Volume 38, Issue 4, November 2004.

[7] Dijkstra, E.W. "A note on two problems in connexion with graphs." Numerische Mathematik, 1959.

[8] Bellman, Richard. "On a Routing Problem", Quarterly of Applied Mathematics, Volume 16, Issue 1, 1958.

[9] Ford Jr., Lester R., D.R. Fulkerson. Flows in Networks. Princeton University Press, 1962.

[10] Johnson, Donald. "Efficient Algorithms for Shortest Paths in Sparse Networks." Journal of the ACM, 1977.

[11] Floyd, Robert. "Algorithm 97 (SHORTEST PATH)." Communications of the ACM, 1977.

[12] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms – 2<sup>nd</sup> Edition. The MIT Press, 2001.

[13] Demetrescu, Camil and Giuseppe Italiano. "A New Approach to Dynamic All Pairs Shortest Paths." ACM Symposium on Theory of Computing, June 2003.

[14] Misra, S, B.J. Oommen. "New Algorithms for Maintaining All-Pairs Shortest Paths." IEEE 10<sup>th</sup> Symposium on Computers and Communications, June 2005.

[15] Blum, Jeremy, Azim Eskandarian. "A Reliable Link-Layer Protocol for Robust and Scalable Intervehicle Communications." IEEE Transactions on Intelligent Transportation Systems, Volume 8, Number 1, March 2007.

[16] Tarng, Jenn-Hwan, Bing-Wen Chuang. "Investigation of Vehicle-to-Infrastructure Communications based on IPv6-based Automotive Telematics." IEEE 7<sup>th</sup> Conference on Intelligent Transportation Systems Telecommunications, June 2007.

[17] Sklavos, Nicolas, Maire McLoone, Xinmiao Zhang. "MONET Special Issue on Next Generation Hardware Architectures for Secure Mobile Computing." Mobile Networks & Applications, Volume 12, Number 4, August 2007.

[18] Zhanfeng, Jia, Chao Chen, Ben Coifman, Pravin Varaiya. "The PeMS algorithms for accurate, real-time estimates of g-factors and speeds from single-loop

detectors.” *IEEE 4<sup>th</sup> Intelligent Transportation Systems Conference*, February 12, 2001.

[19] Miller, Jeffrey. “Vehicle-to-Vehicle-to-Infrastructure (V2V2I) Intelligent Transportation System Architecture.” *IEEE 4<sup>th</sup> Intelligent Vehicles Symposium*, June 2008.

[20] MIT’s CarTel Project. <http://cartel.csail.mit.edu>.

[21] UC Berkeley’s Mobile Millennium Project. <http://traffic.berkeley.edu>.

[22] Airsage Web Site. <http://www.airsage.com>.

[23] Miller, Jeffrey, Muhammad Ali. “Dynamic Fastest Paths with Multiple Unique Destinations (DynFast-MUD) – A Specialized Traveling Salesman Problem with Intermediate Cities.” *IEEE 12<sup>th</sup> International Intelligent Transportation Systems Conference*, October 2009.

[24] Live View GPS – <http://www.liveviewgps.com>.

[25] California Department of Transportation 2008 Average Annual Daily Traffic Report – <http://www.dot.ca.gov/hq/traffops/saferest/trafdata>.