TIME DEPENDENT VEHICLE ROUTING IN A LARGE ROAD NETWORK

By Zhu Zhang, Bachelor of Science

A Thesis Submitted in Partial
Fulfillment of the Requirements
For the Degree of
Master of Science
In the field of Industrial Engineering

Advisory Committee:

Xin Chen, Chair

Ryan Fries

Hoo Sang Ko

Graduate School
Southern Illinois University Edwardsville
December, 2014

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

ABSTRACT

TIME DEPENDENT VEHICLE ROUTING IN A LARGE ROAD NETWORK

By

ZHU ZHANG

Chairperson: Professor Xin Chen

Vehicle Routing Problems (VRP) deal with the delivery of vehicles to multiple destinations (or customers). The objective of VRP is to minimize the total travel cost (e.g., time). This thesis investigates and develops algorithms to solve VRP in time-dependent large road networks. An efficient and effective time-dependent shortest path algorithm is developed. Experiment results show that the arc labeling algorithm is more space efficient compared to the classic node labeling algorithm. This thesis also develops assignment algorithms in order to minimize the maximum travel cost while minimizing the total travel cost. Several methods are implemented to investigate the factors that affect computation efficiency of VRP, including road network size and structure, and computer programming. The applicability of Genetic Algorithms is also studied. Computer programs of algorithms are developed in several software environments including General Algebraic Modeling System (GAMS; GAMS Development Corporation, 2013), MATLAB, Microsoft Excel 2013, and Microsoft Visual Studio 2013. The results show that VRP in a time-dependent large road network can be solved efficiently and effectively using the algorithms and methods developed in this thesis.

ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude to my thesis committee chair Dr. Xin Chen for his guidance, patience and motivation. His illuminating and critical advice encouraged my research and allowed me to finish my thesis. Without his supervision and constant help this thesis would not have been possible.

I would like to thank my thesis committee members Dr. Ryan Fries and Dr. Hoo Sang Ko for their precious comments and suggestions. My special thanks also goes to Dr. Emmanuel S. Eneyo who gave me a lot of inspirations and aspirations during my master study. Thanks to Dr. S. Cem Karacal and Dr. H. Felix Lee for encouragement and support.

Lastly, I would like to thank my family for all they have done for me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Problem Statement

Generalized Vehicle Routing Problems (VRP) deal with the delivery of certain number of vehicles to several destinations (or customers). The objective of VRP is to minimize the total travel cost or time. Often, it is convertible from time to cost, and vice versa; so cost or time will have duplicate meanings hereafter in this thesis. In most cases VRP problems have been modified in order to meet the demand of real life. Hereby, there are many variations arising in the research paper and most of them have specific implications.

In this thesis, the VRP problem would be extended in a time-dependent road network where multiple vehicles will travel from several depots to several destinations with some preparation time. The problem is also called Time Dependent Vehicle Routing Problem (TDVRP), though it differs from typical TDVRP featuring traveling salesman problems in several factors as below:

1. One vehicle can go to only one destination.
2. There is no requirement that vehicles should return to depot after servicing destination.

Hence, the core of TDVRPs in this thesis is to find out time-dependent shortest path between depots and destinations and then minimize the total travel cost for the vehicles available.

Purpose of the Study

This thesis is an extension of the author's research group's previous work on evacuation planning (Kuru et al., 2013). In their research, VRP in a time-dependent transportation network, which has around 300 nodes, could be solved in around half an hour. While the transportation network size increases to more than 1000 nodes, their methodologies could not generate proper results within certain computation time due to memory limitation. Since TDVRP in real-life applications, like evacuation planning, requires very short computation time in order to respond to an emergency situation. The thesis aims at efficiently and effectively solving TDVRP in large road networks which has more than 3,000 nodes. The thesis will investigate and develop time-dependent shortest path algorithms to solve time-dependent shortest path problems, as well as apply assignment algorithms to solve some real-life assignment applications. Moreover, several factors affecting computation time for TDVRP will be discussed. Relevant literatures will be reviewed and future research directions will be suggested.

Significance of the Problem

It has been more than half a century since researchers started to investigate VRPs. The problem still receives highlight and attention due to its widespread applications and the fact that, practically it could not be solved optimally and efficiently. The VRP has a wide range of applications such as bank or postal deliveries and school bus or emergency vehicle routing. Research on the VRP stemmed from the travelling salesman problem (Dantzig, 1954) and was further developed in some early applications (Balas and Toth, 1985; Clarke and Wright, 1964; Haimovich et al., 1988; Laporte et al., 1987). Recent research focused on heuristic and

meta-heuristic approaches (Bräysy and Gendreau, 2005). Most early studies assumed vehicle travel time between two nodes was static and did not change along with traffic conditions.

During the last decade, TDVRPs have gained increasing attention (Almoustafa and Mladenović, 2013; Ando, 2006; Chen et al., 2005; Figliozzi, 2012; Haghani and Jung, 2005; Ichoua et al., 2003; Kritzinger, 2012; Lecluyse et al., 2009; Maden and Black, 2010; Spliet and Gabor, 2012; Vidal et al., 2013; Woensel et al., 2008). In the TDVRP, travel time between two directly connected nodes is dynamic and depends on many factors such as traffic and traffic signal timing. In many cases, travel time may be described as a function of the departure node, time at which a vehicle begins to travel, and the node at which the vehicle plans to arrive.

To date, there is yet no effective algorithms that could solve TDVRP in a large road network which has more than 3000 nodes. The purpose of this thesis is to research and develop TDVRP algorithms as well as investigate factors that affect computation time in order to efficiently solve TDVRPs.

CHAPTER II

REVIEW OF LITERATURE

Vehicle Routing Problem

The Vehicles Routing Problem has been extensively studied in recent years. There are innumerable variants of VRP emerging in the literature. In terms of input parameters, VRP includes the number and origins of vehicles, the size and structure of a road network, the number of destinations, and travel times (dynamic or static), preparation time, time window and etc.; In terms of output parameters, the methods introduced to solve the VRP could be single objective, bi-objective, or even triple-objective. These methods stipulated various conditions for one or more parameters. One of the most common conditions was the upper limit for the size of road networks. Other examples of conditions include static travel time and upper limit for the number of destinations. These methods become ineffective (solutions far from optimal) or inefficient (could not identify a good or optimal solution within an acceptable amount of time) when stipulated conditions do not hold. To develop effective and efficient algorithms to solve VRPs with a wide range of parameter values remains a considerable challenge (Vidal et al., 2013).

Since exact methods that identify optimal vehicle assignments and routes are either ineffective or inefficient for generalized VRPs, heuristic methods including Genetic Algorithms (Haghani and Jung, 2005), tabu search (Ichoua et al., 2003, Archetti et al., 2006), branch and price algorithm (Almoustafa and Mladenović, 2013), and column generation algorithm (Spliet and Gabor, 2012) were studied. Haghani and Jung (2005) presented a

Genetic Algorithm to solve a pick-up or delivery VRP with soft time windows. The study considered multiple vehicles with different capacities, real-time service requests, and dynamic travel times between destinations. Ichoua et al. (2003) conducted experiments to solve the VRP with time-dependent travel speeds, which satisfy the first-in-first-out (FIFO) property, using a parallel tabu search heuristic. Almoustafa and Mladenović (2013) improved a branch-and-bound method to solve the asymmetric distance–constrained VRP suggested by Laporte et al. (1987). Chen et al. (2005) formulated a real-time TDVRP with time windows as a series of mixed integer programming models and developed a heuristic algorithm, which included route construction and improvement. Spliet and Gabor (2012) proposed a formulation of a time window asymmetric VRP and developed two variants of a column generation algorithm to solve the linear programming relaxation of this formulation. Kritzinger (2012) applied variable neighborhood search algorithm to solve the TDVRP with time windows. Maden and Black (2010) proposed a heuristic algorithm for the VRP to minimize the total travel time.

Road networks with different sizes were studied. Laporte et al. (1988) examined a class of asymmetrical multi-depot VRPs and location-routing problems for a network of 80 nodes. Haghani and Jung (2005) solved the TDVRP for networks with 30 demand nodes over 30 time intervals. In the paper of Archetti et al. (2006), while problems with up to 10 nodes could be solved in a few seconds, larger problems with more than 10 nodes required between one hour and four days of computation. Kok et al. (2012) developed 15, 50, and 100 customer problem instances. The 100 customer problem instances were approximately the largest instances which they could solve within practical computation times. Almoustafa and

Mladenović (2013) solved an asymmetric distance–constrained VRP for a network of 1,000 demand nodes.

Time Dependent Shortest Path Problem

The time-dependent shortest path (TDSP) problem was initially proposed by Cooke and Halsey (1966). In their paper, network with discrete time was considered. Orda and Rom (1990) introduced a shortest path problem in a network where link delays are time dependent. In their model, modified Dijkstra Algorithm was used to solve the problem in the case that waiting at nodes is allowed. Ziliaskopoulos and Mahmassani (1993) suggested an algorithm based on Bellman's Principle of optimality, finding it is not advisable in the worst case scenario.

Time Dependent Dijkstra Algorithm with node labeling was first mentioned by Dreyfus (1969) and validated by Kaufman and Smith (1993) to solve TDSP problem when non-passing and non-waiting rules are satisfied. When time is discretized as integer intervals, Time Dependent Dijkstra Algorithm can provide optimal solution with the same complexity as the static shortest path problem.

While many other time-dependent shortest path algorithms were proposed in literature, most of them are practically not suitable to apply in transportation networks, either because of lacking efficiency or violating transportation network properties such as non-passing rule, non-waiting rule.

Assignment Problem

When the input parameters of VRP contain multiple vehicles and multiple destination, assignment decision is generally required to determine which vehicle goes to which destination in order to minimize the total cost. Assignment Problem (AP) is a sub-problem of VRP. There are several variants of APs (Pentico, 2007), such as generalized AP, bottleneck AP, quadratic AP, and semi-AP. The Hungarian Algorithm (Kuhn, 1955) is the most used method to solve AP.

In summary, previous research predominately focused on developing heuristic methods for subsets of VRPs or TDVRPs. Effective and efficient algorithms which can be applied to generalized TDVRPs to obtain optimal vehicle assignments and routes were not available. Most algorithms and methods developed in previous research were not tested using real-world road networks and could not be validated for effectiveness or efficiency. The objective of this research is to investigate and develop effective and efficient algorithms for TDVRP.

This rest of this thesis is organized as follows: chapter 3 presents the problem and methodology. Chapter 4 validates the algorithms using real-world road networks and investigates the factors that affect computation time. Chapter 5 concludes the article with future research directions.

CHAPTER III

METHODOLOGY

Problem Definition

Let A, B, Γ represent sets of vehicles, depots, and demand points, respectively, in a time-dependent road network. There are total $|A|$ vehicles stationed at $|B|$ depots at the beginning of a planning period. Some or all of the $|A|$ vehicles need to be dispatched to $|\Gamma|$ demand points, each of which requires d_γ vehicles, where γ represents a demand point, $\gamma \epsilon \Gamma$. Let $c^{\alpha}_{i,j,t}$ be the cost (time) it requires for a vehicle α, $\alpha \epsilon A$, to travel from node i at time t and to node j. $i, j \in V$, where V is the node set in the road network. $B, \Gamma \subset V$. Let (i,j) represent an arc that originates from node i and points at node j, $(i,j) \in E$, where E is the arc set in the road network. $c^{\alpha}_{i,j,t} = \infty$ if $(i,j) \notin E$. When $i = \beta$, β is a depot and $\beta \epsilon B$, $c^{\alpha}_{\beta,j,t} = \infty$, $\forall j$, if α is not stationed at β or if α is not ready to travel from β at time t.

In the TDVRP, the first objective is to identify the earliest time for α to arrive at a demand point γ when α travels from its depot β. α may travel from β when or after α is ready for travel. Suppose α may travel from β at time t_1 and arrive at γ at time t_2. Alternatively, α may travel from β at time t_3 and arrive at γ at time t_4. According to the First In First Out (FIFO) property, $t_2 < t_4$ if $t_1 < t_3$. Therefore, α should travel from β as soon as α is ready for travel. Let t_β be the time that α becomes ready for travel at β. $c^{\alpha}_{\beta,j,t} = \infty$ when $t < t_\beta$, $\forall j$. $c^{\alpha}_{\beta,j,t} \ll \infty$ when $t \geq t_\beta$ and $(\beta, j) \in E$. The first objective of the TDVRP is to identify the TDSP for α to travel from β at t_β to γ. Eq. (1) is the model whose optimal solution is the TDSP between β and γ when α travels from β at t_β.

$$Minimize \quad \sum_{i}\sum_{j} c^{\alpha}_{i,j,t} X_{i,j}$$

Subject to:

$$\sum_{j} X_{i,j} - \sum_{j} X_{j,i} = 1 \text{ when } i = \beta$$

$$\sum_{j} X_{i,j} - \sum_{j} X_{j,i} = -1 \text{ when } i = \gamma$$

$$\sum_{j} X_{i,j} - \sum_{j} X_{j,i} = 0 \text{ when } i \neq \beta \text{ and } i \neq \gamma$$

$$X_{i,j} = \begin{pmatrix} 1 & (i,j) \text{ is on the path for } \alpha \text{ to travel from } \beta \text{ to } \gamma \\ 0 & \text{otherwise} \end{pmatrix}$$

$$\forall i, j \in V \qquad (1)$$

The second objective of the TDVRP is to minimize the total travel time. Let $s^{\alpha}_{\beta,\gamma}$ represent the optimal value of Eq. (1), i.e., the travel time for α to travel from β and arrive at γ at the earliest time. Eq. (2) models an assignment problem (AP) that determines which vehicles are dispatched to a demand point to meet its demand. Note that the objective of Eq. (2) is not to minimize the summation of arrival times or the latest arrival time. Since Eq. (1) identifies the earliest arrival times, Eq. (2) intends to minimize the total travel time. In many VRPs, lower average travel time implies less uncertainty and more reliable vehicle routing and assignment. Both Eqs. (1) and (2) are pure integer programming problems. If $\sum_{\gamma=1}^{|\Gamma|} d_{\gamma} = |A|$, Eq. (2) is a balanced transportation problem and both constrains may be changed to equality constraints. If $\sum_{\gamma=1}^{|\Gamma|} d_{\gamma} < |A|$, Eq. (2) is infeasible.

$$Minimize \quad \sum_{\alpha=1}^{|A|}\sum_{\gamma=1}^{|\Gamma|} s^{\alpha}_{\beta,\gamma} Y_{\alpha,\gamma}$$

Subject to:

$$\sum_{\gamma=1}^{|\Gamma|} Y_{\alpha,\gamma} \leq 1\,, \forall \alpha$$

$$\sum_{\alpha=1}^{|A|} Y_{\alpha,\gamma} \geq d_{\gamma}, \forall \gamma$$

$$Y_{\alpha,\gamma} = \begin{pmatrix} 1 & \alpha \text{ travels to } \gamma \\ 0 & \text{otherwise} \end{pmatrix} \qquad (2)$$

Methodology

Optimization software packages may be used to solve models in Eqs. (1) and (2). For example, GAMS is a high-level modeling system for mathematical programming and optimization. Eqs. (1) and (2) may be described in algebraic statements in GAMS input files and solvers may be used to find optimal solutions and values. For medium to large road networks, however, this approach is ineffective or inefficient due to limitation of computer memory and extremely long computation time. For real-time transportation planning, time- and space-efficient algorithms must be developed to solve the models in Eqs. (1) and (2). There are three steps as follows to identify optimal solutions and values to the TDVRP.

- Step 1: Develop and implement the TDSP algorithm to identify the shortest paths between pairs of depots and demand points
- Step 2: Apply the assignment algorithm to output of Step 1 to minimize the total travel time
- Step 3: Validate and present optimal solutions and values to the TDVRP

All the computation results in this thesis, if not indicated, were obtained using a Windows 8 x64 Laptop, Intel i7-4700 CPU @2.40 GHZ, and 8.0 GB RAM.

TDSP Algorithms

Real-time TDVRPs remain a great challenge due to time and space complexities. In the TDVRP, characteristics of the road network change with time; a shortest path computed from a snapshot of the road network may not be the shortest path at a different time. The TDSP algorithm is developed to find the shortest paths between depots and demand points in real-time. Three assumptions related to the TDSP algorithms are:

(a) The road network satisfies the FIFO principle, which specifies that if two vehicles take the same route from the same depot to the same demand point, the vehicle leaving the depot first always arrives at the demand point first. According to the FIFO principle, a vehicle should leave its depot or other nodes whenever it is ready. Waiting at any node is never beneficial because a vehicle that leaves later always arrives later (Dean, 2004);

(b) The planning period is "discretized" into sufficiently small time intervals, δ's, $\delta = 1, 2, 3, \ldots$ and $\delta \epsilon \Delta$, where Δ is the set of time intervals over the planning period; and

(c) Travel time between two nodes connected by an arc depends on the time at which a vehicle leaves the beginning node of the arc.

The TDSP algorithm with node labeling (or Time Dependent Dijkstra Algorithm) was first developed by Dreyfus (1969) and further validated by Kaufman and Smith (1993). The TDSP algorithm with node labeling listed below finds the earliest arrival time of a vehicle at a demand point given that the vehicle is stationed at a depot when travel begins. The algorithm

is executed for each pair of demand point and depot for each vehicle. Please note that vehicles stationed at the same depot may be ready to travel from the depot at different times because different vehicles may require different preparation times. For example, time for a driver to arrive at the depot and get ready may vary. The earliest arrival times of vehicles, which are stationed at the same depot, at the same demand point may be different and need to be calculated separately using the TDSP algorithm.

TDSP algorithm with node labeling

1: Assign to every node i, $i \in \mathrm{V}$, in a transportation network a value representing the arrival time of a vehicle α, $\alpha \epsilon \mathrm{A}$, at the node. For a depot node β where α is stationed, set the value to a finite positive integer number representing the time at which α is ready to travel from β. Set the value to infinity for all other nodes i's, $i \in \mathrm{V}$ and $i \neq \beta$;

2: Mark β visited. Mark all other i's unvisited. Set β as the current node;

3: Calculate the arrival time of α at each unvisited neighbor j, $j \in \mathrm{V}$, of the current node i, $i \in \mathrm{V}$. A node j is a neighbor of i if there is an arc that begins at i and points at j, i.e., $(i,j) \in \mathrm{E}$. The arrival time at j is the summation of the value set for i and the travel time $c_{i,j,\delta}^{\alpha}$ between i and j. The travel time $c_{i,j,\delta}^{\alpha}$ is obtained from a three-dimensional matrix, $|\mathrm{V}| \times |\mathrm{V}| \times |\Delta|$, which stores time-dependent travel times. For example, if $i = 1$, $j = 2$, and the value set for node 1 is 15, the travel time between nodes 1 and 2, $c_{1,2,15}^{\alpha}$, is a component in the matrix identified by node 1, node 2, and $\delta = 15$, $\delta = 15$ indicates α begins travelling from node 1 at time 15;

4: For each unvisited neighbor j of the current node i, compare the arrival time at j calculated in Step 3 and the value set for j. Set the value for j as the smaller one between the two;

5: Identify the unvisited node with the smallest value. Mark the node visited. Set the node as the current node. If the node is the desired demand point γ, $\gamma \epsilon \Gamma$, stop. The value set for γ is the earliest arrival time of α travelling from β at γ. Otherwise go to Step 3.

The TDSP algorithm with node labeling was implemented in Visual Basic Application (VBA) for Microsoft Excel and GAMS. The algorithm can only solve TDVRPs for small road networks with less than 1,000 nodes when there are more than 600 time intervals using a Windows 7 x64 PC, Intel i7-3770 CPU @3.40 GHZ, and 16.0 GB RAM. The main reason for size limitation on the road network is the large random-access memory (RAM) space required by node labeling. The algorithm needs to manipulate a three-dimensional matrix, $|V| \times |V| \times |\Delta|$, for node labeling. Each component in the matrix is travel time from one node to the other during a time interval. These travel times are often obtained through field observations (Rakha *et al.*, 2006). $|V|$ is the size of the road network, i.e., the number of nodes. $|\Delta|$ is the number of time intervals. For example, if the road network size increases by tenfold, the storage space for the three-dimensional matrix increases by 100 times.

The degree of a node in a network is the number of arcs connected to the node. Most real-world road networks have a mean degree between two and four (Barabasi, 2002; Jeong, 2003). On average, each node is connected to two to four arcs. If arc labeling is used, the TDSP algorithm manipulates a two-dimensional matrix, $|E| \times |\Delta|$, where $|E|$ is the number of arcs in the road network. Each component in the two-dimensional matrix is travel time

along an arc during a time interval. Travel times in the two-dimensional matrix are the same as those in the three-dimensional matrix, but are organized in a different format that reduces space requirement. Since $|\mathrm{E}| \leq 4|\mathrm{V}|$ according to the mean degree of a road network, storage space requirement for arc labeling is at most $4|\mathrm{V}| \times |\Delta|$, which is much less than $|\mathrm{V}| \times |\mathrm{V}| \times |\Delta|$ required for node labeling for large road networks. The TDSP algorithm with arc labeling described below is used to identify the earliest arrival times of vehicles at demand points. This proposed change is a major improvement over TDSP algorithm with node labeling and will greatly enhance memory performance in solving TDVRP for a larger road network.

TDSP algorithm with arc labeling

1: Assign to every arc (i,j), $(i,j) \in \mathrm{E}$, in a transportation network a value representing the arrival time of a vehicle α, $\alpha \epsilon \mathbf{A}$, at j, $i,j \in \mathrm{V}$. For (i,j) in which $i = \beta$, a depot node where α is stationed, set the value to a finite positive integer number representing the time at which α arrives at j. The arrival time at j is the summation of time at which α is ready to travel from β and travel time from β to j. The travel time from β to j, $c^{\alpha}_{\beta,j,\delta}$, is obtained from a two-dimensional matrix, $|\mathrm{E}| \times |\Delta|$, which stores time-dependent travel times. For example, if $\beta = 1$, $j = 2$, and the time at which α is ready to travel from β is $\delta = 15$, the travel time between nodes 1 and 2, $c^{\alpha}_{1,2,15}$, is a component in the matrix identified by arc $(1,2)$ and $\delta = 15$. Set the value to infinity for all other arcs (i,j), $(i,j) \in \mathrm{E}$ and $i \neq \beta$;

2: Mark all (i,j) unvisited;

3: Identify the unvisited (i,j) with the smallest value. Mark (i,j) visited. Set the destination node, i.e., the second node j, in (i,j) as the current node. If j is the desired demand point γ, $\gamma \epsilon \Gamma$, stop. The value set for (i,γ) is the earliest arrival time of α travelling from β at γ;

4: For each unvisited (i,j) whose i is the current node, compare the arrival time at j and the value set for (i,j). The arrival time at j is the summation of time at which α arrives at i and travel time from i to j. The arrival time at i is the value set for the arc marked as visited in Step 3. The travel time from i to j, $c^{\alpha}_{i,j,\delta}$, is obtained from the matrix $|\mathrm{E}| \times |\Delta|$. δ is the value set for the arc marked as visited in Step 3;

5: For each unvisited (i,j) whose i is the current node, set its value as the smaller one between the arrival time at j calculated in Step 4 and the value set for (i,j). Go to Step 3.

The TDSP algorithm with arc labeling is implemented in VBA and Visual Basic (VB). The algorithm can not only solve TDVRPs for small road networks with less than 500 nodes, but also TDVRPS for large road networks with more than 3000 nodes using a Windows 7 x64 PC, Intel i7-3770 CPU @3.40 GHZ, and 16.0 GB RAM.

Assignment Algorithm

After calculating all the shortest paths between depots and demand points, an assignment algorithm needs to be implemented to determine which vehicles from a depot will travel to a demand point to meet the demand. There are several variants of APs (Pentico, 2007), e.g., generalized AP, bottleneck AP, quadratic AP, and semi-AP. The TDVRP is a semi-AP that a

vehicle is not required to be assigned to a demand point. The Hungarian Algorithm (Kuhn, 1955) is used as the assignment method to solve the TDVRP.

Revised Hungarian Algorithm

To best of knowledge, Hungarian Algorithm can only be used to minimize or maximize the total cost. In some circumstances, though, maximum cost of assignment is strictly enforced in order to meet specific cost limit. This kind of problem is being referred in literature as bottleneck assignment problem (BAP). The objective of BAP is to minimize the maximum cost of assignment. To give an example, an ambulance is required to arrive at an incident site within a stipulated time. If one area is equipped with several ambulances while at the same time they are dispatched to different locations, minimizing the maximum traveling time for ambulances is more real-life practical than minimizing the total time spent on routing for all the ambulances. Similar examples were also given in other literatures (such as Gross, 1959; Ford and Fulkerson, 1966; Ravindran and Ramaswami, 1977). In their work, even though multiple models and approaches of solving such problems were developed, solutions are still too complex to be implemented. Hungarian Algorithm will be revised for the purpose of minimizing the maximum cost of assignment problem while minimizing the total cost.

Problem description:

Assume cost matrix can be expressed as (M, N), where M denotes the number of vehicles, N denotes the number of customers. In order to simplify the explanation, assume every customer needs one vehicle and every vehicle can be dispatched to only one customer. In other words, there are more vehicles than customers (i.e.: M is greater than N)

Procedures:

1. Subtract the largest of the smallest entries in each column from all the entries of cost matrix.
2. Draw lines through appropriate rows and columns so that all the non-positive entries of the cost matrix are covered and the minimum number of such lines is used. Check whether the number of covering lines is greater than or equal to N. If yes, then go to step 4, otherwise go to step 3.
3. Subtract the smallest positive entry of the new matrix from all the rows, then go to step 2.
4. Label all the non-positive entries while replacing all unlabeled entry with infinity.
5. Solve the problem from step 4 using Hungarian Algorithm.

Reductio ad absurdum:

After step 1, 2, 3 and 4, the maximum cost is minimized based on following two facts:

a) Number of marked entries in each column is equal or greater than one and number of rows that contain marked entries is equal or greater than one. i.e.: there is at least one vehicle is selected for each customer.
b) For each non-positive entry that is marked, a connection is established between customers and vehicles. Any increase in updated cost matrix will result in no-connection for at least one customer. Any decrease in updated cost matrix will result in more than one connection for at least one customer.

Step 5 will minimize the total cost after minimizing the maximum cost.

Example 1: An original cost matrix was given in table 1.

Table 1: Original cost matrix for example 1

3	2	0
5	4	3
4	4	4
4	4	2

Step 1: Subtract the largest of the smallest entries (respectively as 3, 2, and 0) in each column from all the entries of cost matrix. (i.e. subtract 3 from each column)

Table 2: Cost matrix after step 1 for example 1

0	-1	-3
2	1	0
1	1	1
1	1	-1

Step 2: The number of covering lines is 2 (less than column number), so go to step 3.

Step 3: Subtract 1 from all the entries.

Table 3: Cost matrix after step 3 for example 1

-1	-2	-4
1	0	-1
0	0	0
0	0	-2

Step 2: The number of covering lines is 3, so go to step 4.

Step 4: Label all the non-positive entries while replacing all unlabeled entry with infinity.

Table 4: Cost matrix after step 4 for example 1

-1	-2	-4
∞	0	-1
0	0	0
0	0	-2

Step 5: Solve the matrix in table 4 using Hungarian Algorithm, the resulting matrix is shown in table 5.

Table 5: Cost matrix after step 5 for example 1

0	0	0
0	4	0
4	0	0
0	0	0

After implementing the revised Hungarian Algorithm, the maximum cost is 4 and the total cost is 8.

Example 2: Another cost matrix was given in table 6.

Table 6: Original cost matrix for example 2

158	7	144	164	28	143	70	140	101
64	152	171	128	66	82	19	79	163
40	43	50	150	8	169	108	72	48
137	166	3	52	68	53	85	1	131
29	113	41	59	65	62	23	153	149
69	110	37	134	57	103	114	18	26
61	139	16	160	93	34	4	49	71
117	54	5	39	33	83	138	15	31
141	77	87	99	17	90	104	60	146
136	20	44	32	63	30	147	123	130
23	67	118	112	109	24	84	86	111
16	96	162	14	47	91	167	106	17
11	125	157	38	148	151	161	111	16
142	36	98	168	121	92	76	35	126
102	165	58	124	127	78	105	80	107
22	2	73	9	6	88	156	100	25
154	89	25	74	116	45	159	133	135
129	170	94	119	21	12	132	120	81
18	56	122	42	46	97	27	51	75

Calculated by Hungarian Algorithm, the maximum cost is 17 and total cost is 74. The resulting matrix is in table 7.

Table 7: Resulting matrix after applying Hungarian Algorithm for example 2

0	7	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	8	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	4	0	0
0	0	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	17
11	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	9	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	12	0	0	0
0	0	0	0	0	0	0	0	0

However, after applying the revised Hungarian Algorithm, the maximum cost is 16, decreasing by 1 while the total cost increases to 78. The resulting matrix is shown in table 8.

Table 8: Resulting matrix after applying the revised Hungarian Algorithm for example 2

0	7	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	8	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	4	0	0
0	0	5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	16
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	9	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	12	0	0	0
0	0	0	0	0	0	0	0	0

The above examples showed each step of the revised Hungarian Algorithm and made comparisons between the resulting matrixes calculated by Hungarian Algorithm and the revised Hungarian Algorithm.

Since computation time for assignment problem is trivial comparing to that of TDSP in this thesis, the remaining section will not implement these two algorithms at the same time. Instead, all the experiments in the next chapter, other than indicated, will only be using Hungarian Algorithm to solve assignment problems.

CHAPTER IV

EXPERIMENTS AND RESULTS

Case Study of a Small Road Network

A road network in the Midwest of the United States of America (Figure 2) was analyzed to validate the TDSP and assignment algorithms. The network has 346 nodes (diamonds in Figure 2), out of which 6 are depots and 15 are demand points. There are 654 roads (arcs in Figure 2) connecting all the nodes (mean degree is $\frac{654\times2}{346} = 3.78$). Total 92 vehicles are available at 6 depots. Each time interval is one minute. The TDSP and assignment algorithms were implemented in VBA and GAMS. The computation results were obtained using a Windows 7 x64 PC with Intel i7-3770 CPU @3.40 GHZ and 16.0 GB RAM.

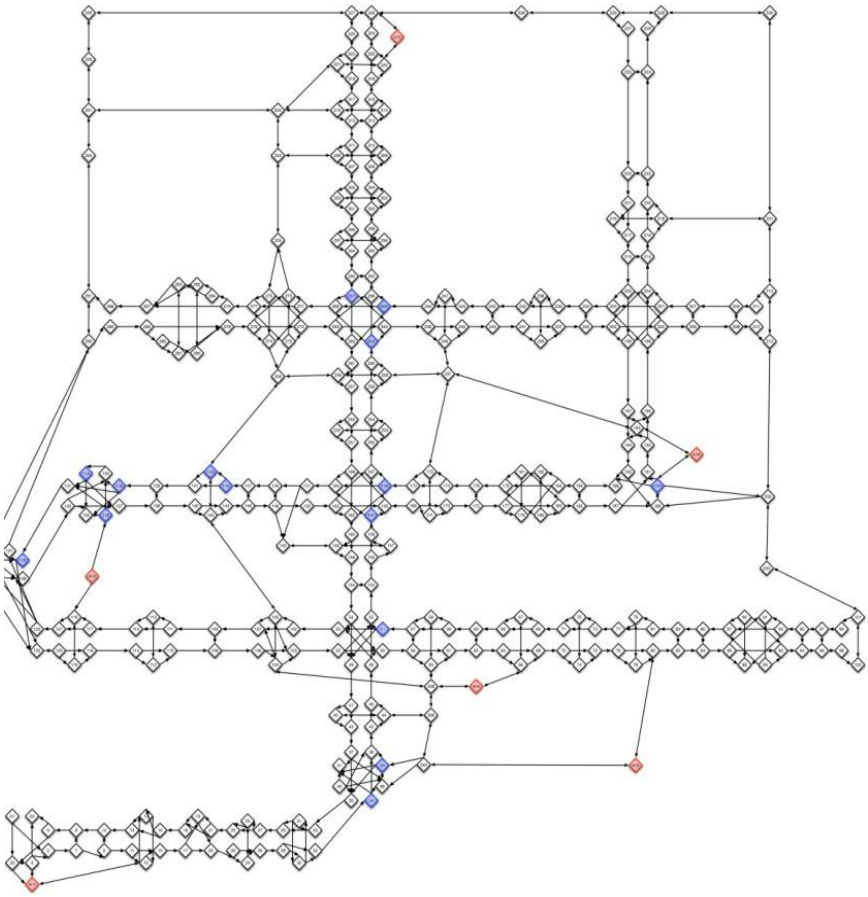

Figure 1: A transportation network in the metropolitan area of St. Louis, USA

Table 9 summarizes run times of the algorithms in VBA and GAMS. It shows that there is a substantial advantage in using VBA to identify the optimal solution and value for real-time TDVRPs. GAMS employs a suite of solvers, e.g., CPLEX, and algorithms, e.g., simplex and branch-and-cut algorithms, to solve linear programming and mixed integer programming problems, but lacks efficiency in computer memory management. The TDSP and assignment algorithms implemented in VBA directly manipulate data and are customized to solve the TDVRPs; they are more efficient in terms of run time and computer memory space. Table 9 does not show a substantial difference in run times between node labeling and arc labeling for the TDSP algorithm. The difference between these two methods needs to be further investigated using larger road networks.

Table 9: Algorithms run time (seconds) for the road network in figure 1

Planning Period	GAMS	VBA Node Labeling	VBA Arc Labeling
120 minutes	191	9	13
240 minutes	449	11	13
360 minutes	681	13	13
480 minutes	872	15	15
1,440 minutes	1,954	33	24

Table 10 shows the snapshot from Microsoft Excel spreadsheet, where we can easily see the information of depot, destination, vehicle number, preparation time and travel time as well as route. All of the results were obtained by implementing TDSP algorithm with arc labeling and Hungarian Algorithm in VBA for Microsoft Excel 2013.

Table 10: Results for TDVRP in figure 1

Depot	Dest.	Vehicle No.	Prep. Time	Travel Time	Route
345	36	v67	11	17	345>340>38>39>36
344	37	v53	12	27	344>338>104>103>52>49>47>43>37
345	41	v66	11	18	345>340>38>39>41
345	41	v64	12	18	345>340>38>39>41
344	51	v49	12	31	344>338>104>103>52>50>56>57>51
344	51	v47	11	31	344>338>104>103>52>50>56>57>51
345	53	v62	12	29	345>340>38>39>36>42>46>48>53
344	54	v50	10	33	344>338>104>103>52>55>153>154>54
344	55	v51	12	23	344>338>104>103>52>55
342	123	v28	14	22	342>119>121>123
342	123	v27	14	22	342>119>121>123
345	125	v59	11	43	345>340>339>338>104>105>142>143>141>139>136>131>129>125
342	125	v24	14	28	342>119>121>123>125
342	125	v23	10	21	342>133>131>129>125
342	125	v26	12	21	342>119>121>123>125
342	125	v20	12	21	342>119>121>123>125
342	125	v18	10	21	342>133>131>129>125
342	125	v15	12	21	342>119>121>123>125
342	125	v13	15	31	342>119>121>123>125
342	125	v12	12	21	342>119>121>123>125
344	129	v54	16	39	344>338>104>105>142>143>141>139>136>131>129
342	129	v22	15	27	342>133>131>129
342	129	v17	15	27	342>133>131>129
342	129	v14	15	27	342>133>131>129
342	133	v19	16	17	342>133
342	133	v16	16	17	342>133
342	134	v25	13	23	342>133>135>134
342	134	v21	13	23	342>133>135>134
345	136	v58	11	33	345>340>339>338>104>105>142>143>141>139>136
345	136	v57	12	33	345>340>339>338>104>105>142>143>141>139>136
345	143	v56	12	26	345>340>339>338>104>105>142>143
344	145	v55	16	31	344>338>104>105>142>144>146>147>145
345	145	v61	12	31	345>340>339>338>104>105>142>144>146>147>145
344	161	v52	12	34	344>338>104>103>52>55>153>155>159>161
346	166	v74	12	27	346>195>337>172>170>166
346	166	v75	12	27	346>195>337>172>170>166
346	168	v85	10	35	346>195>337>258>259>257>255>251>168

Study of a Larger Road Network

Figure 2 shows a road network in the City of San Francisco (Brinkhoff, 2002), which included 174,956 nodes and 223,001 arcs with a mean degree of 2.55 ($= \frac{223,001 \times 2}{174,956}$).

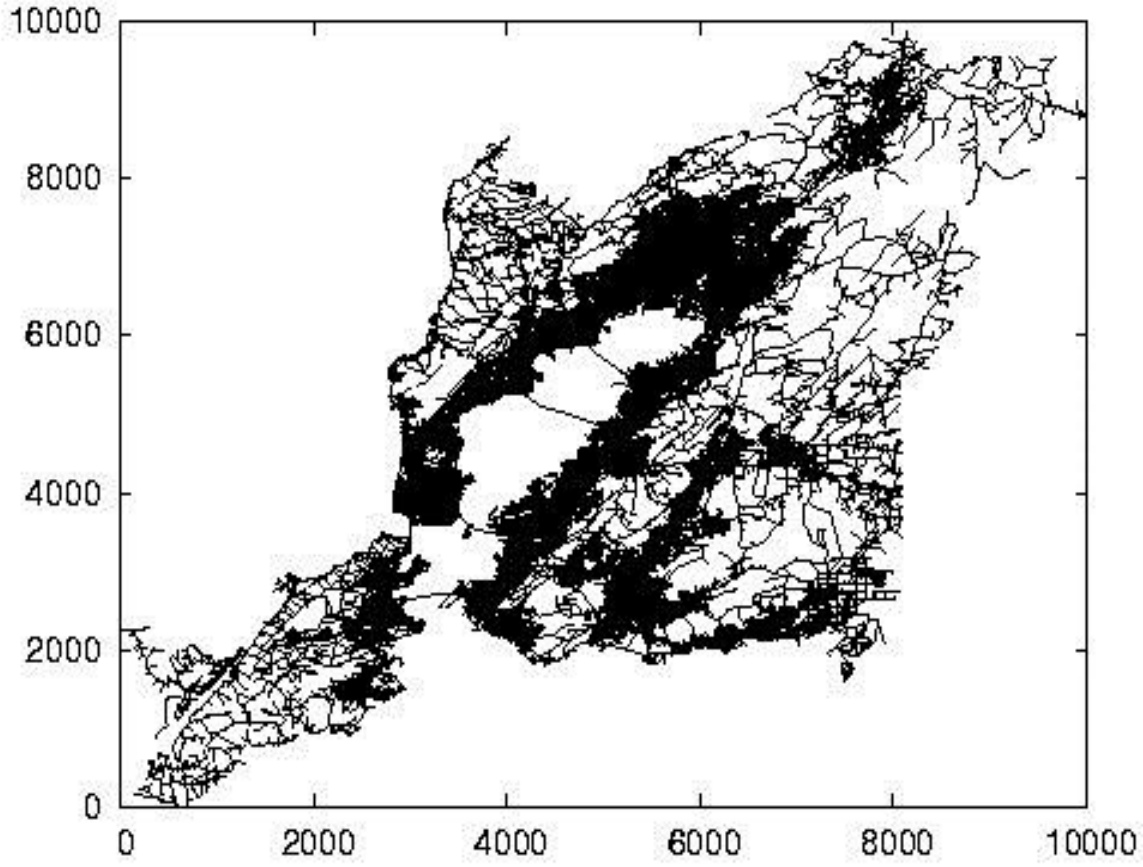

Figure 2: A road network in the city of San Francisco (Brinkhoff, 2002)

Because the entire San Francisco road network in figure 2 is too large to apply existing TDSP algorithms to calculate TDVRP, 9 different portions of the network were selected for experiment. As shown in Figure 3, each portion has the same area size (i.e.: 500*500); dense area means there are more roads, therefore the arc/node ratio is high; Likewise, sparse area means the arc/node ratio is relatively low.

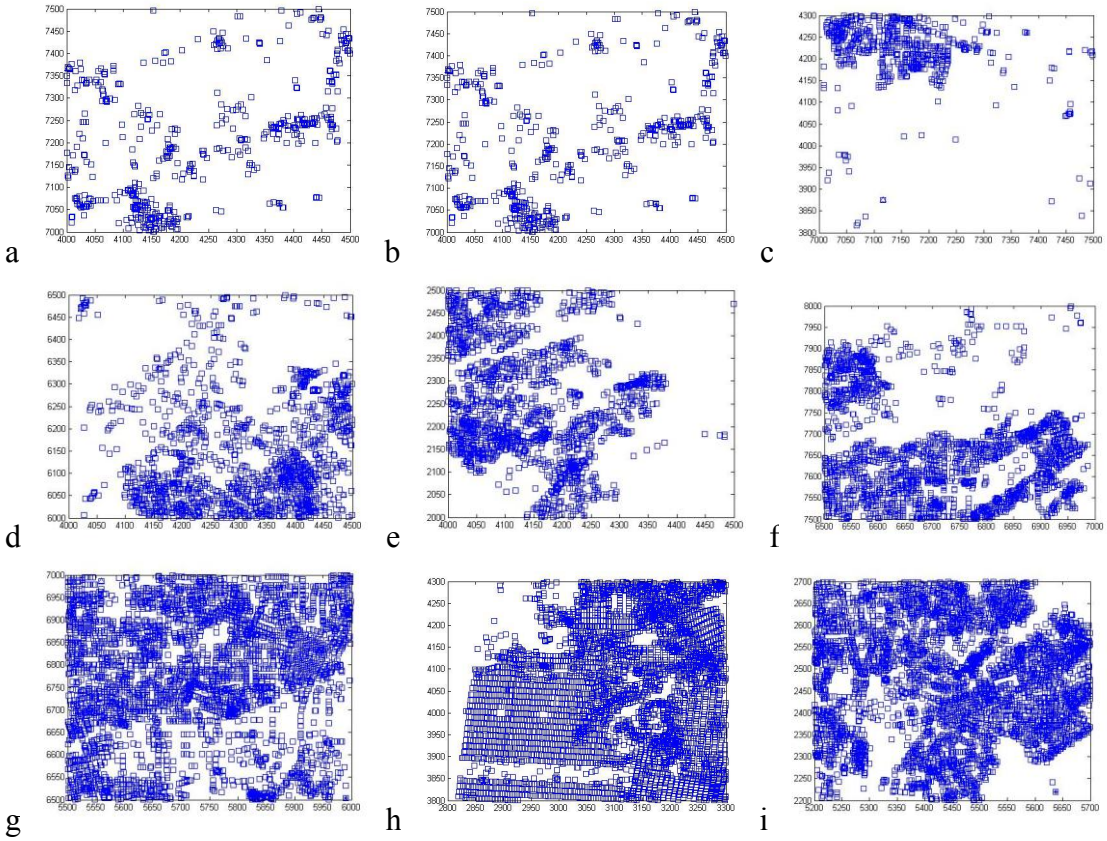

Figure 3: 9 sub road networks taken from figure 2 with the same area size

Table 11 summarizes the mean degree of the 9 sub road networks which are shown in figure 3. For the same area size road network, the mean degree ranges from 2.23 to 3.21; the node number ranges from some 500 to 4,000; the arc number ranges from some 500 to 6,000. In order to be consistent throughout the study hereafter, it was assumed that depot points and demand points are randomly uniformly selected among the network nodes. There are 100 vehicles which are randomly uniformly stationed at random number of depots (i.e.: depot number ranges from 1 to 100) and there are 20 random destinations where 40 vehicles will be needed. The planning horizon is 600 minutes with one minute time interval. For each vehicle, preparation time is randomly generated.

Table 11: Mean degree of the 9 sub road networks shown in figure 3

Network	Node Number	Arc Number	Mean Degree
a	508	572	2.25
b	512	571	2.23
c	669	780	2.33
d	1,459	1,760	2.41
e	1,629	1,872	2.30
f	2,172	2,589	2.38
g	3,674	4,861	2.65
h	3,926	6,299	3.21
i	4,078	4,998	2.45

As TDSP algorithm with node labeling is not feasible to solve TDVRP in a road network with more than 500 nodes because of limitation of computer memory and GAMS requires more than 10 hours to compute the optimal solution to the TDVRP for 500-node network and sometimes stops unexpectedly due to insufficient memory. The study, hereafter, will only use TDSP algorithm with arc labeling if not indicated, otherwise. Computation time for VBA program is summarized in table 12.

Table 12: Computation time for TDVRP in each road network of figure 3 (seconds)

Node Number / Number of Depots	508	512	669	1,459	1,629	2,172	3,674	3,926	4,078
1	95	64	68	272	323	526	1,720	2,907	1,813
2	58	61	69	263	297	546	1,708	2,872	1,895
3	44	52	66	292	289	509	1,703	2,841	1,777
4	59	52	70	263	313	550	1,746	2,780	1,778
5	45	44	71	290	269	532	1,696	2,878	1,799
6	54	95	73	301	275	549	1,722	2,839	1,865
7	39	60	56	259	279	504	1,641	2,767	1,788
8	44	46	65	426	293	519	1,709	2,804	1,852
9	49	59	73	309	303	539	1,680	2,759	1,876
10	58	55	75	253	303	571	1,672	2,769	1,819
20	55	65	74	250	276	556	1,738	2,778	1,870
30	62	57	79	274	284	527	1,731	2,860	1,939
40	45	52	78	264	302	512	1,666	2,830	1,791
50	83	51	65	267	305	525	1,692	2,722	1,807
60	96	61	75	288	289	523	1,731	2,780	1,963
70	49	43	58	295	299	535	1,741	2,837	1,776
80	46	55	78	262	289	509	1,714	2,811	1,853
90	60	51	72	287	280	497	1,694	3,142	1,775
100	55	46	64	308	327	523	1,647	2,835	1,855
AVERAGE	58	56	70	286	295	529	1,703	2,832	1,836
STANDARD DEVIATION (STD)	16	11	6	39	16	19	31	89	56
STD/AVERAGE	0.29	0.20	0.09	0.14	0.05	0.04	0.02	0.03	0.03

Even though TDSP algorithm with node labeling and GAMS could not provide optimal solution to TDVRP in a large road network due to insufficient memory, TDSP algorithm with arc labeling still worked effectively with adequate computation time. This finding further indicated that TDSP algorithm with arc labeling is more suitable than node labeling in calculating TDVRP in a large road network.

As shown in table 12, we can see that average run time increased along with node number and mean degree. Noticeably, though 4,078 nodes was the largest sub road network, it required less computation time comparing to the road network which has 3,926 nodes. The

only reason could be the difference of mean degree since the other factors are the same. The coming section will discuss the factors that affect computation time.

Factors that Affect Computation Time

Effects of road network size on computation time

For whatever kind of VRPs, size of road network is a major concern for computation time. Harwood et at. (2013) observed that under certain circumstances when number of nodes increases, run time grows rather more slowly than one may expect, in relation to the size of the instance.

As we can see from table 12, computation time varied significantly along with node number and mean degree; other minor factors such as number of depots, had a limited effect on computation time. To decrease complexity of problem, node number and mean degree were treated as the two main factors that affect computation time in this study. Two more sub road networks obtained from the road network in the City of San Francisco, together with two existing sub road networks from figure 3, are shown in figure 4.

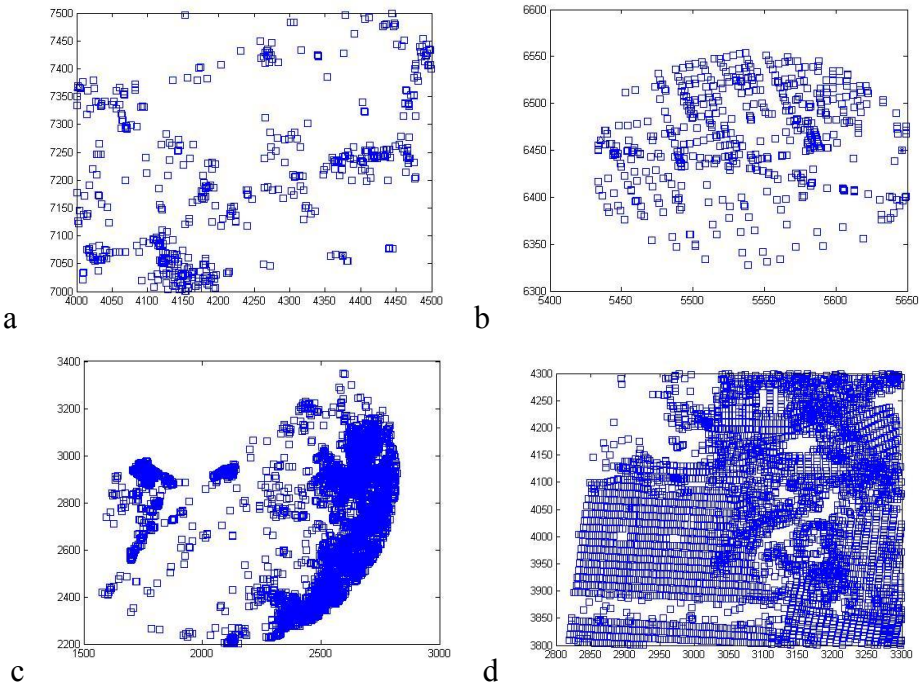

Figure 4: 4 sub road networks (2 levels of node number and 2 levels of mean degree)

Table 13 shows the information of the sub road networks in figure 4. For simplicity of investigation of individual effects of each factor, the four sub road networks were divided into two levels for each factor. Some 500 nodes was considered as small road networks (-1) while some 3,900 nodes was considered as large road networks (+1); Road network with mean degree of 2.23 was considered as sparse (-1) while mean degree of 3.21 was regarded as dense (+1).

Table 13: Mean degree of four road networks in figure 4

Network	Node Number	Arc Number	Mean Degree
a	512	571	2.23
b	516	828	3.21
c	3,916	4,375	2.23
d	3,926	6,299	3.21

Parameters of TDVRP in these four road networks were set the same with previous section, i.e., there are 100 vehicles which will be randomly assigned to every depot (number of depot may vary from 1 to 100) and there are 20 destinations (or customers) where 40 vehicles are required. The planning horizon is 600 minutes with one minute time interval. To ensure unbiased result and balance out inequality, location of every depot point and destination point was randomly generated, computation time for each road network was considered as a block, computation for each problem ran in a random sequence. The proposed regression model can be written as:

$$t=\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_{12} x_1 x_2+\pi+\varepsilon \qquad (3)$$

t: computation time

x_1: node number

x_2: mean degree

β_0: average run time

β_1: coefficient of node number

β_{12}: coefficient of interaction between node number and run time

π: block effect

ε: error effect

Data in table 14 was analyzed in Minitab 16 using above regression model. Figure 5 below shows the normal probability plot of residuals. Since the plot is approximately linear, it means residuals conform to normality assumption and regression model can adequately describe relationship between computation time and node number and mean degree. (Note: residual is arithmetic difference between actual computation time and computation time predicted by regression model.)

Table 14: Computation time for TDVRP in four road networks shown in figure 4 (seconds)

Node Number / Number of depot	512		516		3,916		3,926	
	Replicate		Replicate		Replicate		Replicate	
	1	2	1	2	1	2	1	2
1	64	59	105	80	1,481	1,555	2,972	3,058
2	58	57	97	78	1,512	1,552	2,986	3,108
3	66	56	106	93	1,521	1,552	3,000	3,057
4	63	63	87	100	1,515	1,530	2,963	3,104
5	66	60	100	82	1,510	1,505	2,958	3,049
6	52	68	87	69	1,492	1,529	2,968	2,966
7	56	54	98	108	1,531	1,548	3,004	3,060
8	54	55	95	103	1,493	1,579	3,023	3,158
9	47	51	105	101	1,564	1,495	3,008	3,103
10	45	54	98	102	1,486	1,587	2,996	2,959
20	58	56	92	100	1,543	1,580	2,931	3,037
30	55	48	169	90	1,477	1,561	3,162	3,111
40	56	57	114	82	1,454	1,561	2,934	3,106
50	55	61	101	106	1,514	1,638	2,942	3,154
60	58	58	104	78	1,484	1,551	2,997	3,138
70	70	56	100	99	1,463	1,571	3,017	3,045
80	66	61	94	102	1,522	1,530	3,071	3,168
90	50	69	91	90	1,512	1,525	2,956	2,916
100	58	55	95	87	1,524	1,504	2,990	3,117

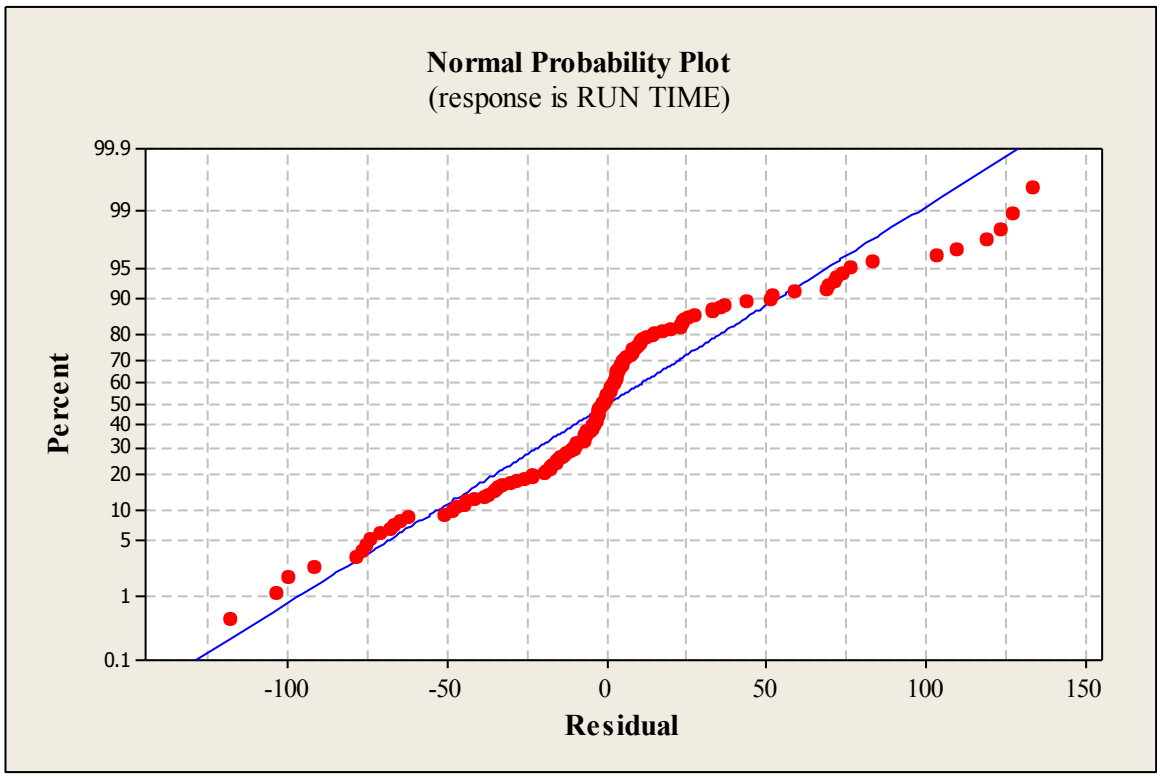

Figure 5: Normal probability plot of computation time residuals

Figure 6 shows the main effects of node number and mean degree on run time. As we can see, neither of the two lines is horizontal, it means both factors have main effects on run time. When node number is higher, run time mean is higher. When mean degree of road network is higher, run time mean is also higher. The line for node number is steeper than that of the mean degree, it means the magnitude of node number is greater in this setup.

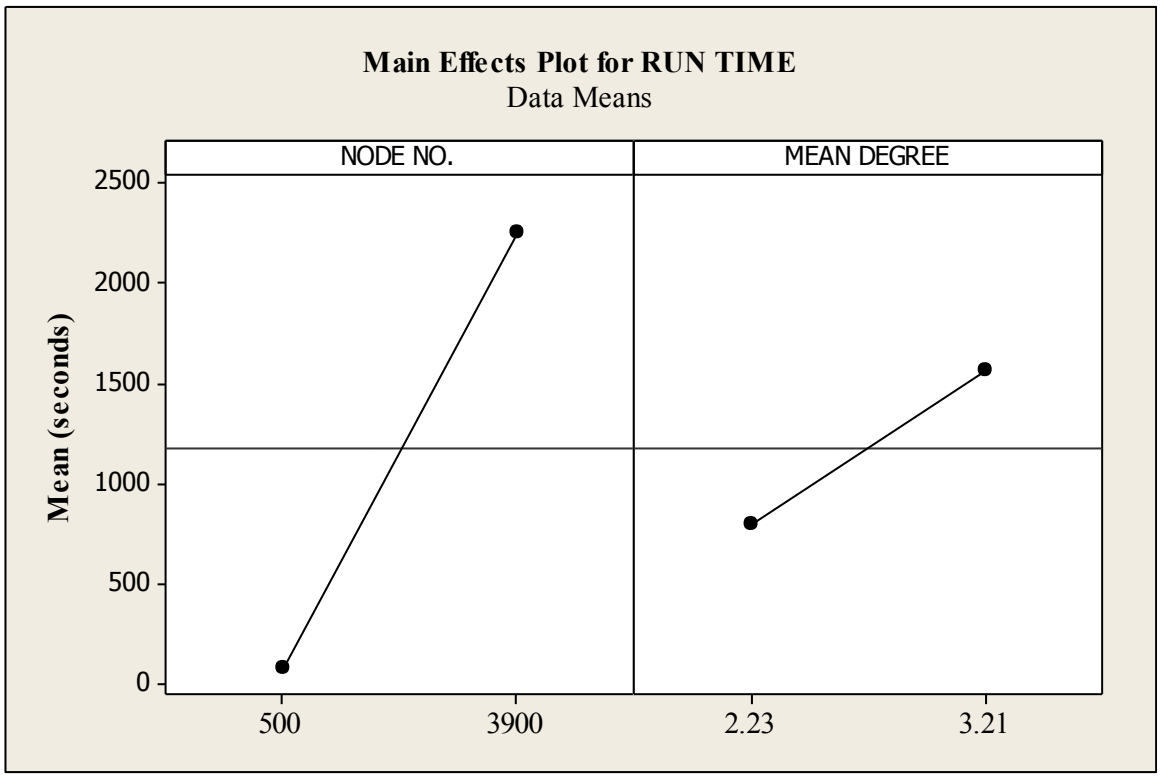

Figure 6: Main effects plot for run time

The interaction effect plot in figure 7 shows there is an interaction between node number and mean degree since the two lines are not parallel. When the node number is high, mean degree has a substantial positive effect; but when node number is low, mean degree has a minor positive effect.

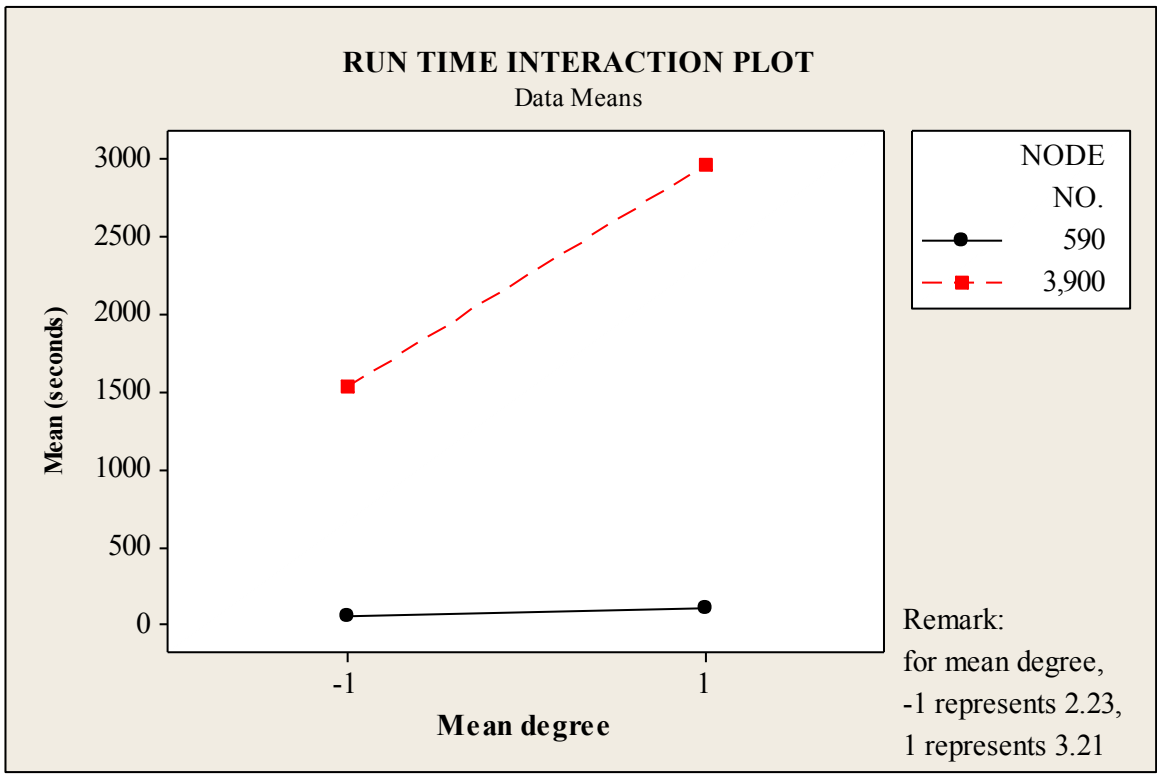

Figure 7: Interaction effect plot for run time

A simple mathematic model for the effects of node number and mean degree on run time is proposed as follows:

$$t=1{,}179+1{,}102x1+386x2-1{,}465-1{,}098x1*x2 \qquad (4)$$

Where: t is run time, x1, x2 represents node number and mean degree respectively.

Table 15 summarizes estimated effects and coefficients for run time.

Table 15: Estimated effects and coefficients for run time (coded units)

Term	Effect	Coef	SE Coef	T	P
Constant		1,179	3.446	342.12	0
Block		-1,465	21.45	-68.29	0
C2	2,203	1,102	3.446	319.69	0
C3	773	386	3.446	112.11	0
C2*C3	-2,196	-1098	21.452	-51.18	0
S = 42.3459, R-Sq(adj) = 99.88%					

Effects of computer programming and parallel computing

Ghiani et al. (2003) pointed out one of the three major developments that have contributed to the acceleration and quality of algorithms relevant in a real-time context is parallel computing. In their paper, several parallel algorithms such as tabu-search were introduced. This section works on parallel computation as well as computer programming.

Let's still take a large time-dependent road network from figure 4 which has 3,926 nodes and 6,299 arcs as an example, the parameter for TDVRP were kept the same with previous setup. As far as we understand, computation of TDVRP in the computer program could be divided into three steps as follows:

- Reading data into memory
- The Shortest path calculation
- Assignment calculation.

The computation time for each step of TDVRP using VBA in Microsoft Excel is shown in table 16.

Table 16: Run time for each step of TDVRP using VBA

Run times	1	2	3	4	5	Average
Reading data into memory (seconds)	114	113	110	116	114	113
Shortest path calculation (seconds)	2,885	2,950	2,875	2,910	2,820	2,888
Vehicle assignment (seconds)	169	170	156	163	177	167
Total (seconds)	3,168	3,233	3,141	3,189	3,111	3,168
Total (minutes)	53	54	52	53	52	53

It took around one hour to calculate the TDVRP for a road network with 3,926 nodes using VBA. Most of the run time was spent on the calculation of time-dependent shortest path. Around 90% of computation time was spent on the calculation of the shortest path while

reading data into memory and assignment calculation consumed a small portion of total time (around 10%).

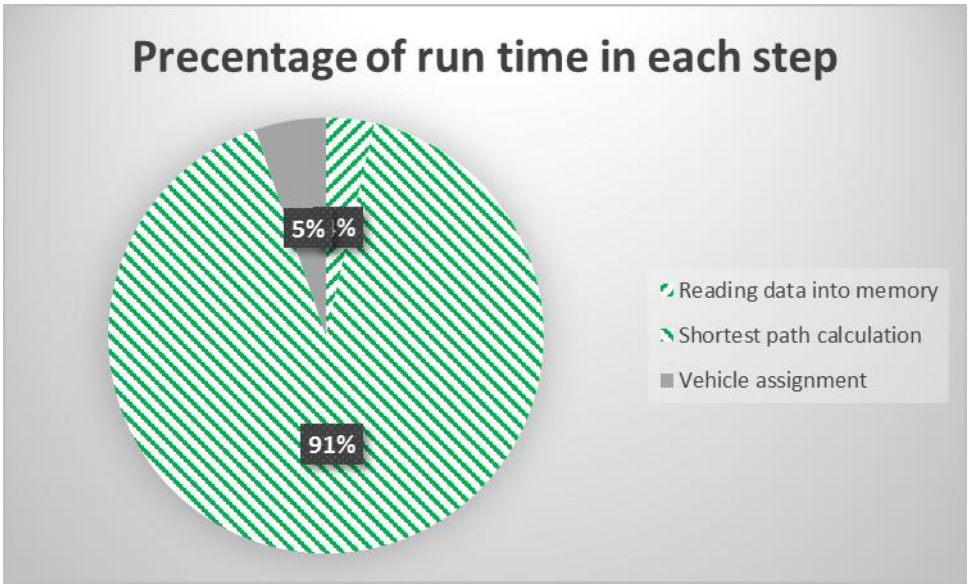

Figure 8: Percentage of run time in each step using VBA

The same VBA code was implemented in Visual Basic .NET Framework based Microsoft Visual Studio 2013. Table 17 summarizes the run time for each step of TDVRP using the VB program.

Table 17: Run time for each step of TDVRP using VB

Run times	1	2	3	4	5	Average
Reading data into memory (seconds)	145	145	144	144	144	144
Shortest path calculation (seconds)	238	239	242	244	238	240
Vehicle assignment (seconds)	3	2	3	3	3	3
Total (seconds)	385	386	389	390	385	387
Total (minutes)	6	6	6	7	6	6

Total computation time for the VB program decreased significantly to around 6 minutes from 52 minutes of the VBA program even though both programs have the same code. There was no big difference for time spent on data reading between VB and VBA programs; however, calculation for the shortest path and vehicle assignment took much less time for the VB

program than VBA. About 60% of computation time was spent on calculating the shortest path, while percentage of run time in reading data increased significantly to 37% of total time. Overall, TDVRP for a large road network which has around 4,000 nodes, could be solved in around 6 minutes by TDSP algorithms with arc labeling through the VB program.

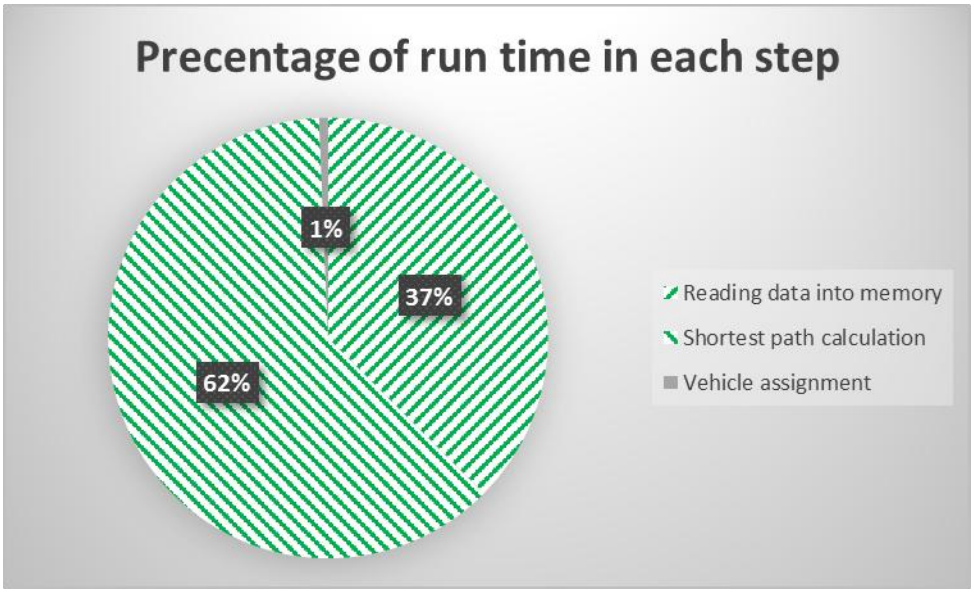

Figure 9: Percentage of run time in each step using VB

Furthermore, in order to implement parallel computing, VB code was revised in Microsoft Visual Studio 2013. Run time of each step for the revised VB program is shown in table 18.

Table 18: Run time for each step of TDVRP using the revised VB program

Run times	1	2	3	4	5	Average
Reading data into memory (seconds)	8	7	7	8	7	7
Shortest path calculation (seconds)	350	398	385	320	310	353
Vehicle assignment (seconds)	2	3	2	2	3	2
Total (seconds)	360	408	394	330	320	362
Total (minutes)	6	7	7	6	5	6

Total run time almost did not vary too much between the VB program and the revised VB program. However, run time for calculating the shortest path took 2 minutes longer than that

of the VB program. From figure 10 we can see, calculation of the shortest path almost consumed 97% of total computation time.

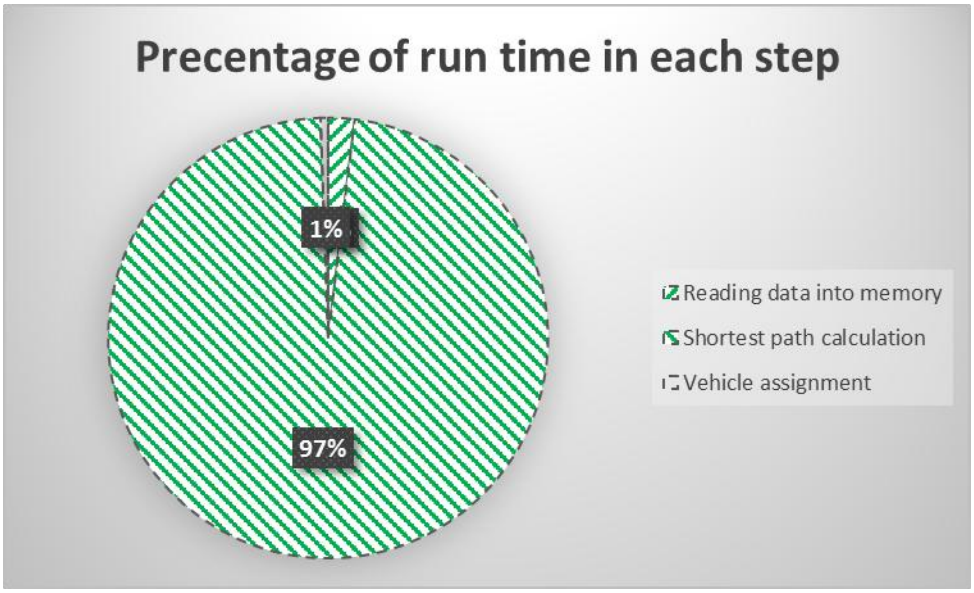

Figure 10: Percentage of run time in each step using the revised VB program

Part of the reason could be that, even though the code ran in a parallel way, slack time of CPU was not being utilized. Generally, long operation process like calculating the shortest path, requires a high level of coordination and synchronization of resources in operation system. The overhead of using parallel computing is expensive. The same reasoning could also be found in literature as Barney (2010) stated that parallel applications are much more complex than corresponding serial applications, perhaps an order of magnitude. The amount of time required to coordinate parallel tasks, as opposed to doing useful work. Parallel overhead can include factors such as: task start-up time, synchronizations, data communications and etc. Figure 11 shows the comparison of run time in the VBA, VB and revised VB programs.

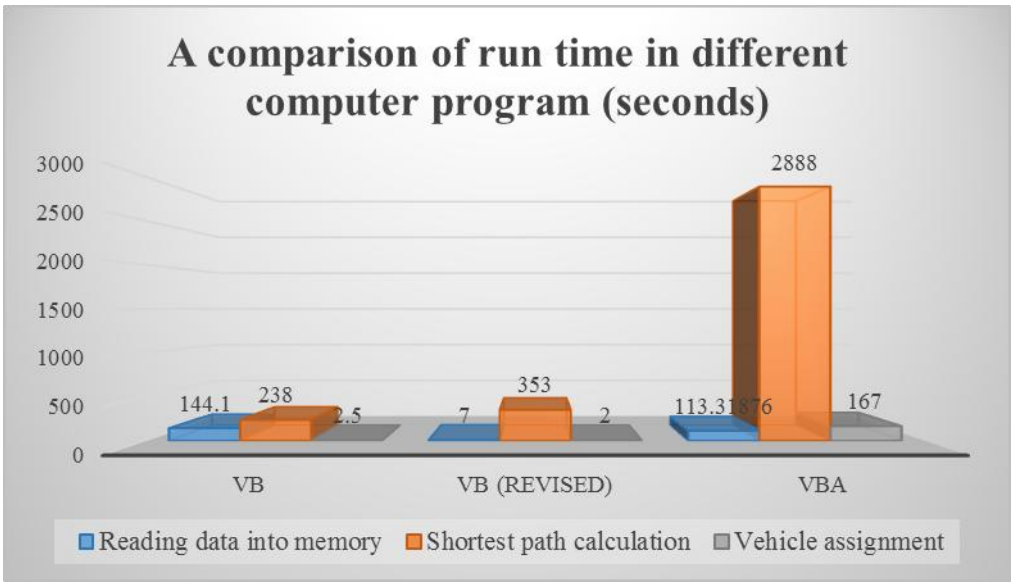

Figure 11: A comparison of run time in different computer programs (seconds)

Genetic Algorithm for Solving TDVRP

The idea of Genetic Algorithms originates from evolutionary theory. It incorporates natural selection, genetic crossover and mutation to select the offspring with the highest fitness. Genetic Algorithms have been widely applied in computer science, engineering, economics, manufacturing and physics, etc. This section will discuss the applicability of Genetic Algorithms in solving TDVRP. The program will be coded in MATLAB.

Considering the complexity of TDVRP in this thesis, using Genetic Algorithms to solve static shortest path problem in a small road network was tested ahead of applying for TDVRP. Take one of the road networks from figure 4 which has 512 nodes and 571 arcs as an example to calculate the shortest path. Figure 12 shows the trend of average shortest path (upper curve) and optimal shortest path in each generation (lower curve).

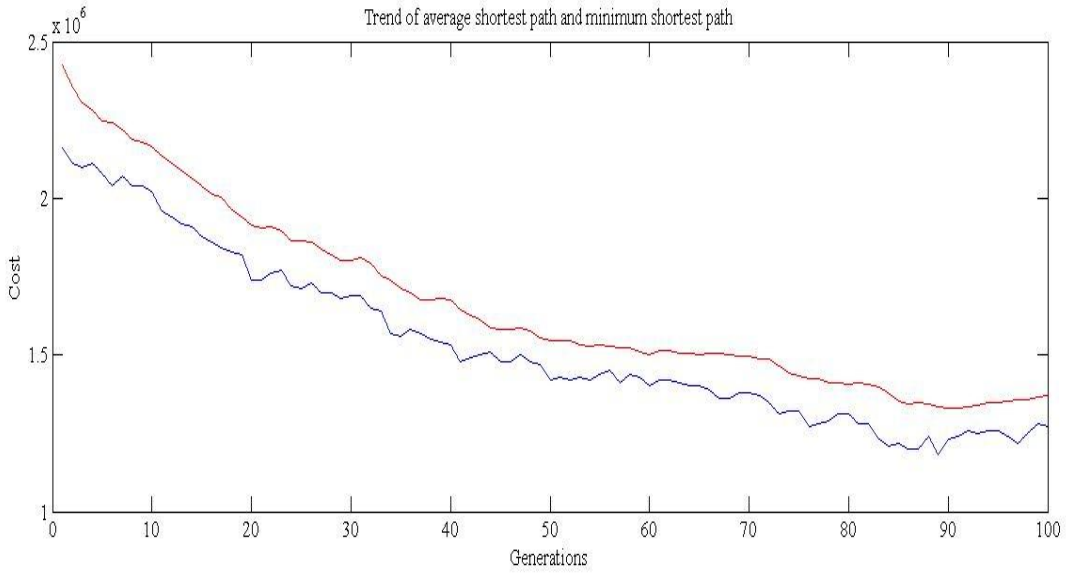

Figure 12: Trend of average shortest path (upper curve) and minimum shortest path in each generation (lower curve) (population size is 50, generation size is 100, crossover probability 0.9 and mutation probability 0.01 with random initial population)

The trend indicated randomized initial population did not contain a feasible solution. The Genetic Algorithm improved solutions through generation by generation. Similarly, while we increased population size and generation size, the trend did not vary too much and result was still far from optimality.

Before we reached the conclusion that it is not suitable to apply Genetic Algorithms to the shortest path problem, we made further investigation about Genetic Algorithms in the case that initial population was selected among the feasible solutions instead of random generation. We found that under no circumstance there will be feasible solutions for a large network, if population is randomly generated.

Case 1: If setting the probability of mutation to positive number, the trend of average shortest path for the population is shown in figure 13. Trend shown in figure 13 is right opposite of result shown in figure 12. The population started at a very good point. Along with generation,

more and more infeasible solutions made the overall population worse. It is obvious that initial feasible population could not guarantee feasible computation results when mutation rate is positive.

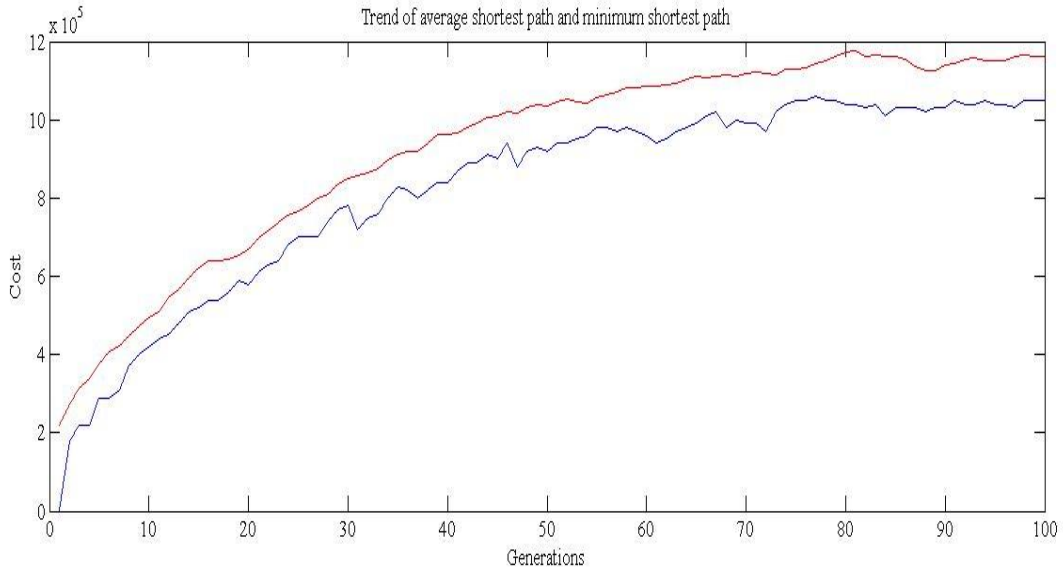

Figure 13: Trend of average shortest path (upper curve) and minimum shortest path in each generation (lower curve) (population size is 50, generation size is 100, crossover probability 0.9 and mutation probability 0.01 with selected initial population)

Case 2: If setting probability of mutation to 0, the population turned better and better along with generation till all the individuals turned to the best initial point by genetic assimilation. The result shows that the Genetic Algorithms are able to pick up the best solution if all the solutions in initial population are feasible and the mutation rate is 0.

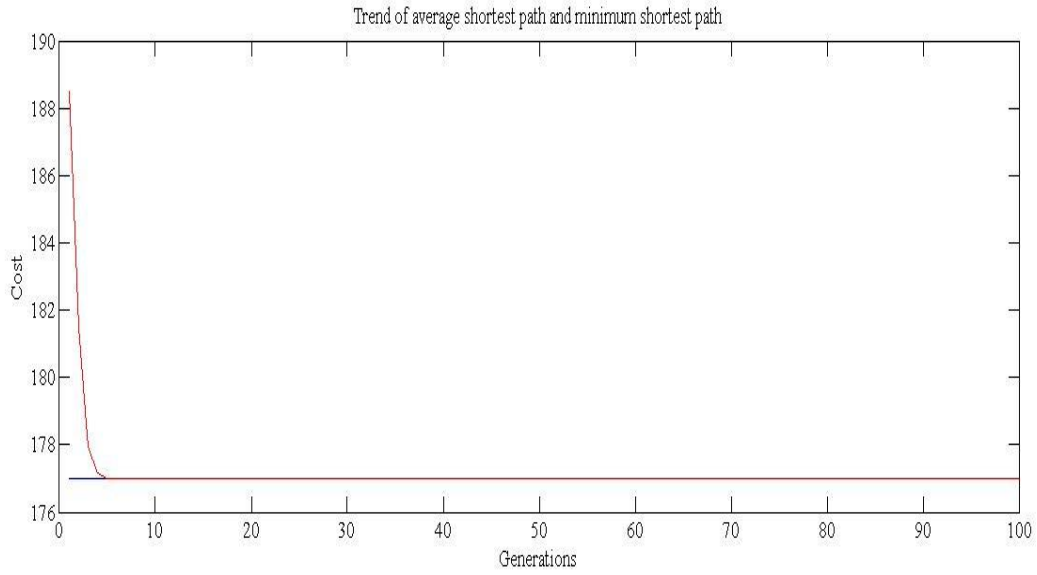

Figure 14: Trend of average shortest path (upper curve) and minimum shortest path in each generation (lower curve) (population size is 50, generation size is 100, crossover probability 0.9 and mutation probability 0 with selected initial population)

Combining all the scenarios we discussed above, though it has been extensively applied in solving traveling salesman problem (TSP) (Baker, 2003; Vidal et al., 2012 and etc.), we can conclude that Genetic Algorithms are not suitable for solving the shortest path problem, not to mention those with time dependencies.

Summary of Findings

In this chapter, we investigated and implemented TDSP algorithms with node labeling and arc labeling, assignment algorithms as well as Genetic Algorithms. Computation programs were developed in several software environments like GAMS, MATLAB, Microsoft Excel 2013 and Microsoft Visual Studio 2013. TDSP algorithms with arc labeling and node labeling both worked efficiently for TDVRP in a small network. In a large road network, TDSP algorithm with arc labeling showed great competence in terms of memory efficiency.

With regard to memory limitation of current technology, TDSP algorithm with arc labeling is strongly suggested to solve time-dependent shortest path problem.

Both Node number and mean degree of road network play important roles in affecting computation time. When the road network size is given, computer programs should be carefully reviewed before implementation of algorithms. The TDVRP, discussed in this thesis, could not be solved by Genetic Algorithms.

CHAPTER V

CONCLUSIONS AND FUTURE RESEARCH

This thesis investigated TDSP algorithm with node labeling in a time-dependent shortest path problem. Even though node labeling algorithm is very efficient in solving TDVRP in a small road network, it could not be applied to a large road network which has more than 1000 nodes due to memory limitation. Hence, more memory-efficient algorithm was proposed in order to solve TDSP problems in a larger road network. Multiple experiment results validated the accuracy of TDSP algorithm with arc labeling; also, they showed that arc labeling algorithm is both efficient and effective in solving TDVRP for a road network which has more than 3000 nodes within around 6 minutes. Besides TDSP algorithms, this paper also applied assignment algorithms to assignment problems. Revise Hungarian Algorithm was developed in order to minimize the maximum cost of assignment while minimizing the total cost. Two examples about the revised Hungarian Algorithm were shown and calculation results were compared to Hungarian Algorithm. Combining TDSP algorithms and Hungarian Algorithms investigated and developed in this thesis, optimal results of TDVRP could be obtained.

Moreover, multiple factors that affect computation efficiency were discussed. The size of road network, including node number and mean degree, substantially determines the complexity of TDVRP for a road network. Computation time increases tremendously along with the increase of node number when mean degree is high. A simple mathematical model was proposed by a two-factor two-level design of experiment to describe the effects of node number and mean degree on computation time. Besides road network size, computation time

of TDVRP through several computer programs was presented. When different software is used, computation performance may vary significantly.

Genetic Algorithms, even though very popular in the literature, were found not suitable in solving the shortest path problem, not to mention TDVRP in this thesis. The reason could be the difficulties of generating feasible solutions as initial population. However, other meta-heuristic algorithms, such as tabu search, could be tested in future studies. Parallel computing still remains an intriguing and challenging area to be discovered to improve computation efficiency for repetitive tasks.

REFERENCES

A. Orda and R. Rom (1990). Shortest-path and minimum-delay algorithms in networks with time-dependent edge length. Journal of the ACM 37 (3), 607-625.

A. Ravindran, V. Ramaswami (1977), On the bottleneck assignment problem, Journal of Optimization Theory and Applications 21 (4) 451–458.

Almoustafa, S., Hanafi, S., & Mladenović, N. (2013). New exact method for large asymmetric distance-constrained vehicle routing problem. European Journal of Operational Research, 226(3), 386-394.

Ando, N., & Taniguchi, E. (2006). Travel time reliability in vehicle routing and scheduling with time windows. Networks and Spatial Economics, 6(3-4), 293-311.

Andres Figliozzi, M. (2012). The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. Transportation Research Part E: Logistics and Transportation Review, 48(3), 616-636.

Archetti, C., Speranza, M. G., & Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science, 40(1), 64-73.

Baker, B. M., & Ayechew, M. A. (2003). A Genetic Algorithm for the vehicle routing problem. Computers & Operations Research, 30(5), 787-800.

Balas, E., Toth, P. (1985). Branch and bound methods. In: Lawer et al. (Eds.), The Traveling Salesman Problem. Wiley, Chichester, pp. 361–401.

Barabasi, A. L. (2002). Linked: the new science of networks. Perseus Publishing, Cambridge, Massachusetts.

Barney, B. (2010). Introduction to parallel computing. Lawrence Livermore National Laboratory, 6(13), 10.

Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation science, 39(1), 104-139.

Brinkhoff, T. (2002). A framework for generating network-based moving objects. GeoInformatica, 6(2), 153-180.

Chen, H. K., Hsueh, C. F., & Chang, M. S. (2006). The real-time time-dependent vehicle routing problem. Transportation Research Part E: Logistics and Transportation Review, 42(5), 383-408.

Clarke, G., Wright, J.V. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12 (4), 568–581.

Dantzig, G.B., Fulkerson, D.R., Johnson, S.M. (1954). Solution of a large-scale traveling salesman problem. Operations Research 2, 393–410.

Dreyfus, S. E. (1969). An appraisal of some shortest-path algorithms. Operations research, 17(3), 395-412.

GAMS Development Corporation (2013). GAMS Distribution 24.0.2.

Ghiani, G., Guerriero, F., Laporte, G., & Musmanno, R. (2003). Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. European Journal of Operational Research, 151(1), 1-11.

Haghani, A., & Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. Computers & operations research, 32(11), 2959-2986.

Haimovich, M., Rinnooy Kan, A.H.G., Stougie, L. (1988). Analysis of heuristic routing problems. In: Golden et al. (Eds.), Vehicle Routing: Methods and Studies. North Holland, Amsterdam, pp. 47–61.

Harwood, K., Mumford, C., & Eglese, R. (2013). Investigating the use of metaheuristics for solving single vehicle routing problems with time-varying traversal costs. Journal of the Operational Research Society, 64(1), 34-47.

Ichoua, S., Gendreau, M., & Potvin, J. Y. (2003). Vehicle dispatching with time-dependent travel times. European journal of operational research, 144(2), 379-396.

Jeong, H. (2003). Complex scale-free networks. Physica A: Statistical Mechanics and Its Applications, 321, 226-237.

Kaufman, D. E., & Smith, R. L. (1993). Fastest paths in time-dependent networks for intelligent vehicle-highway systems application∗. Journal of Intelligent Transportation Systems, 1(1), 1-11.

Kok, A. L., Hans, E. W., & Schutten, J. M. J. (2012). Vehicle routing under time-dependent travel times: the impact of congestion avoidance. Computers & operations research, 39(5), 910-918.

Kritzinger, S., Doerner, K. F., Hartl, R. F., Kiechle, G. Ÿ., Stadler, H., & Manohar, S. S. (2012). Using traffic information for time-dependent vehicle routing. Procedia-Social and Behavioral Sciences, 39, 217-229.

Kuhn, H.W. (1955). The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–97.

L. Cooke and E. Halsey (1966). “The shortest route through a network with timedependent internodal transit times”. Journal of Mathematical Analysis and Applications 14, 492-498.

L.R. Ford Jr., D.R. Fulkerson (1966), Flows in Networks, Princeton University Press, Princeton, NJ, (Section II.6).

Laporte, G., Nobert, Y., & Taillefer, S. (1987). A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. Mathematical Modelling, 9(12), 857-868.

Laporte, G., Nobert, Y., & Taillefer, S. (1988). Solving a family of multi-depot vehicle routing and location-routing problems. Transportation science, 22(3), 161-172.

Lecluyse, C., Van Woensel, T., & Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times. 4OR, 7(4), 363-377.

Maden, W., Eglese, R., & Black, D. (2010). Vehicle routing and scheduling with time-varying data: A case study. Journal of the Operational Research Society, 61(3), 515-522.

Gross, O. (1959). The bottleneck assignment problem (No. P-1630). RAND CORP SANTA MONICA CALIF. Pentico, D. W. (2007). Assignment problems: A golden anniversary survey. European Journal of Operational Research, 176(2), 774-793.

Spliet, R., & Gabor, A. F. (2012). The Time Window Assignment Vehicle Routing Problem (No. EI 2012-07, pp. 1-19). Erasmus School of Economics (ESE).

Van Woensel, T., Kerbache, L., Peremans, H., & Vandaele, N. (2008). Vehicle routing with dynamic travel times: A queueing approach. European journal of operational research, 186(3), 990-1007.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid Genetic Algorithm for multidepot and periodic vehicle routing problems. Operations Research, 60(3), 611-624.

Ziliaskopoulos, A.K., Mahmassani, H.S. (1993). Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. Transportation Research Record 1408, 94±100.