OPERATIONS AND SUPPLY CHAIN MANAGEMENT
Vol. 11, No. 2, 2018, pp. 55 - 65
ISSN 1979-3561 | EISSN 2759-9363

An Improved Efficient Algorithm for Time Dependent Vehicle Routing

Xin Chen
Industrial Engineering
Southern Illinois University Edwardsville, Illinois, United States
Email: xchen@siue.edu

ABSTRACT

A universal challenge in solving a variety of vehicle routing problems (VRPs) is the exponential increase of computation time when the number of entities such as roads, vehicles, and destinations increases. This article studies a class of VRPs in which multiple vehicles located at different locations are dispatched to multiple destinations. Real time VRP in large road networks with time dependent travel time remains a challenge because computation time for the optimal vehicle routes and assignment increases significantly as the size of road networks increases. This article (a) applies a shortest path algorithm with arc labelling to reduce required computer storage space; (b) develops a revised Hungarian method to minimize the latest arrival time and total travel time; and (c) uses appropriate computer programs and tools to reduce computation time for optimal vehicle routing. The algorithm developed in this article identifies the optimal vehicle routes and assignment in six minutes for large and dense road networks.

Keywords: *Hungarian method, shortest path algorithm, vehicle routing*

1. INTRODUCTION

In a vehicle routing problem (VRP) studied in this article, vehicles travel from multiple origins to destinations. The goal is to meet the demand (the number of vehicles) at destinations and minimize total travel time or the latest arrival time for vehicles. The VRP has been extensively studied since 1960s (Balas and Toth, 1985; Bräysy and Gendreau, 2005; Clarke and Wright, 1964; Drexl, 2012; Gendreau *et al.*, 1997; Haimovich *et al.*, 1988; Lai and Tong, 2012; Lai *et al.*, 2014; Laporte *et al.*, 1987; Laporte, 2009; Ozsoydan and Sipahioglu, 2013). During the last decade, research has shifted to time-dependent vehicle routing problems (TDVRPs; Almoustafa *et al.*, 2013; Ando and Taniguchi, 2006; Chen *et al.*, 2006; Chen *et al.*, 2013; Figliozzi, 2012; Gendreau *et al.*, 2015; Haghani and Jung, 2005; Ichoua *et al.*, 2003; Kritzinger *et al.*, 2012; Lecluyse *et al.*, 2009; Maden *et al.*, 2010; Rekersbrink *et al.*, 2009; Spliet and Gabor, 2012; Van Woensel *et al.*, 2008; Vidal *et al.*, 2012). In the TDVRP, travel time between two nodes connected by a road is dynamic and depends on traffic.

This article develops an efficient TDVRP algorithm for vehicle routing. A TDVRP algorithm comprises of two sequential steps. In the first step, a shortest path (a path with minimum travel time) between an origin where a vehicle is stationed and a destination is calculated. Because travel time is dynamic, a shortest path for each combination of vehicle, origin, and destination is computed. In the second step, an assignment problem is formulated using the shortest paths (vehicle routes) identified in the first step as input. The assignment problem is solved to determine vehicle assignments, i.e., which vehicles are dispatched (assigned) to which destinations. The goal is to minimize total travel time or the latest arrival time.

This article develops a time-dependent shortest path algorithm with arc labeling (TDSP-ARC), which efficiently computes the shortest paths for the TDVRP. The TDSP-ARC improves the classic Dijkstra's algorithm in two aspects. First, the TDSP-ARC uses dynamic travel time to calculate shortest paths. Secondly, the TDSP-ARC uses arc labeling to reduce computer storage space required for computation. The output of the TDSP-ARC is used in a revised Hungarian method to minimize both the latest arrival time and total travel time. This article also examines factors that affect time efficiency of the TDVRP algorithm and improves time efficiency using appropriate computer programs and tools.

The rest of this article is organized as follows. Section 2 reviews relevant literature. Section 3 introduces the TDSP-ARC algorithm. Section 4 describes the revised Hungarian method. Section 5 examines factors affecting time efficiency of the TDVRP algorithm and applies computer programs and tools to improve time efficiency. Section 6 concludes the article with main results and limitations. Section 7 discusses applications and future research directions. All computation results in this article are obtained using a Windows 8 x64 Laptop with Intel i7-4700 CPU @2.40 GHZ and 8.0 GB RAM.

The main contributions of this article include: (a) the TDSP-ARC algorithm using dynamic travel time to calculate the shortest paths and arc labeling to reduce required computer storage space; (b) the revised Hungarian method using the output of the TDSP-ARC algorithm to minimize the latest arrival time and total travel time; (c) analyses of factors affecting time efficiency of the TDVRP algorithm; and (d) application of computer programs and tools to improve time efficiency of the TDVRP algorithm.

2. BACKGROUND

The VRP has many input parameters, including the number of vehicles, origins of vehicles, size and structure of a road network, number of destinations, location of destinations, and road network traffic. Numerous methods were introduced in recent years to solve the VRP; these methods stipulated various conditions for one or more

Table 1 Summary of Literature on Vehicle Routing

Research Methods	Articles	Optimality
Branch-and-bound	Almoustafa *et al.*, 2013; Laporte *et al.*, 1987; Laporte *et al.*, 1988	Optimal
Branch-price-cut with metaheuristic	Alvarez and Munari, 2017	Good feasible solutions
Column generation	Spliet and Gabor, 2012; Wilhelm, 2001	Optimal
Dijkstra's algorithm	Kritzinger *et al.* (2012)	Optimal
Genetic algorithms	Chakroborty and Mandal, 2005; Haghani and Jung, 2005; Küçükoğlu and Öztürk, 2014; Lai *et al.*, 2014; Ozsoydan and Sipahioglu, 2013	Good feasible solutions
Heuristic algorithm	Maden *et al.*, 2010	Good feasible solutions
Mixed integer programming	Chen *et al.*, 2006	Good feasible solutions
Neighborhood search	Defryn and Sörensen, 2017	Good feasible solutions
Tabu search	Euchi and Chabchoub, 2010; Ichoua *et al.*, 2003; Lai and Tong, 2012; Ozsoydan and Sipahioglu, 2013	Good feasible solutions

parameters. This article studies a class of VRPs in which multiple vehicles located at different locations are dispatched to multiple destinations with time dependent travel time. **Table 1** is a summary of these methods and their solutions relevant to the class of VRPs studied in this article. One of the most common conditions was the upper limit for the size of road networks. Other conditions include static travel time and upper limit for the number of destinations. These methods become ineffective (solutions far from optimal) or inefficient (cannot 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 many parameters remains a considerable challenge (Vidal *et al.*, 2014).

Since exact methods that identify optimal vehicle routes and assignments were either ineffective or inefficient for VRPs, heuristic methods including the genetic algorithm (Chakroborty and Mandal, 2005; Haghani and Jung, 2005; Küçükoğlu and Öztürk, 2014; Lai *et al.*, 2014), Tabu search (Euchi and Chabchoub, 2010; Ichoua *et al.*, 2003; Lai and Tong, 2012), branch and price (Almoustafa *et al.*, 2013), and column generation algorithm (Spliet and Gabor, 2012; Wilhelm, 2001) were studied. Ozsoydan and Sipahioglu (2013) compared performance of the genetic algorithm, Tabu search, and nearest neighborhood-based initial solution technique for capacitated VRP. Haghani and Jung (2005) presented a genetic algorithm to solve a pick-up or delivery VRP with soft time windows. Their study considered multiple vehicles with different capacities, real-time service requests, and dynamic travel time between destinations. Defryn and Sörensen (2017) developed a two-level heuristic algorithm to solve the clustered VRP.

Time windows in VRPs were also studied for various applications such as catering firms (Küçükoğlu and Öztürk, 2014) and multiple delivery men (Alvarez and Munari, 2017). 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 *et al.* (2013) improved a branch-and-bound method to solve the asymmetric distance-constrained VRP suggested by Laporte *et al.* (1987). Chen *et al.* (2006) 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 *et al.* (2012) applied variable neighborhood search algorithm to solve the TDVRP with time windows. Maden *et al.* (2010) proposed a heuristic algorithm for the VRP to minimize total travel time.

Road networks with different sizes were also 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 destinations over 30 time intervals. Almoustafa *et al.* (2013) solved an asymmetric distance–constrained VRP for a network of 1,000 destinations. In summary, most previous research focused on developing heuristic methods for VRPs and TDVRPs. Effective and efficient algorithms which may be applied to general TDVRPs to obtain optimal vehicle routes and assignment were not available. Many algorithms and methods developed in previous research were not tested or validated using real-world road networks.

3. TDSP-ARC

Most road networks follow the FIFO principle. The FIFO principle specifies that if two vehicles take the same route from the same origin to the same destination, the vehicle leaving the origin earlier always arrives at the destination earlier. According to the FIFO principle, a vehicle should leave its origin or other intermediate nodes whenever it is ready. Waiting at any node is never beneficial because a vehicle that leaves later always arrives later. Under the FIFO principle, two optimization objectives, minimizing total travel time and minimizing total arrival time, become equivalent. Another common practice in solving the TDVRP is the use of time intervals. In TDVRPS, a planning period is a time period during which vehicles must be dispatched to destinations to meet the demand. The planning period is "discretized" into sufficiently small, equal, and consecutive

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.

Let $\mathrm{A}, \mathrm{B}, \Gamma$ represent sets of vehicles, origins, and destinations, respectively, in a road network. There are total $|\mathrm{A}|$ vehicles stationed at $|\mathrm{B}|$ origins at the beginning of a planning period. Some or all of the $|\mathrm{A}|$ vehicles need to be dispatched to $|\Gamma|$ destinations, each of which requires d_γ vehicles, where γ represents a destination, $\gamma \epsilon \Gamma$. Let $c_{i,j,t}^{\alpha}$ be the cost (time) it requires for a vehicle α, $\alpha \epsilon \mathrm{A}$, to travel from node i at time t to node j. $i, j \in \mathrm{V}$, where V is the node set in the road network. $\mathrm{B}, \Gamma \subset \mathrm{V}$. Let (i, j) represent an arc that originates from node i and points at node j, $(i, j) \in \mathrm{E}$, where E is the set of arcs in the road network. $c_{i,j,t}^{\alpha} = \infty$ if $(i, j) \notin \mathrm{E}$. When $i = \beta$, β is an origin and $\beta \epsilon \mathrm{B}$, $c_{\beta,j,t}^{\alpha} = \infty$ for $\forall j$ if α is not ready to travel from β at time t. Eq. (1) is a model whose optimal solution is the TDSP between β and γ when α travels from β at t. Depending on time t, the TDSP between β and γ may be different. The optimal solution to Eq. (1) is time dependent.

$$Minimize \quad \sum_i \sum_j c_{i,j,t}^{\alpha} X_{i,j}$$

Subject to:

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

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

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

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

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

The second objective of the TDVRP is to minimize total travel time. Let $s_{\beta,\gamma}^{\alpha}$ represent the optimal value to Eq. (1), i.e., the minimum time for α to travel from β and arrive at γ. Eq. (2) models a general assignment problem (GAP) that determines which vehicles are dispatched to each demand point to meet the demand. Note that the objective of Eq. (2) is to minimizing total travel time, which is equivalent to minimizing total arrival time according to the FIFO principle. Both Eqs. (1) and (2) are pure integer programming problems. If $\sum_{\gamma=1}^{|\Gamma|} d_\gamma = |\mathrm{A}|$, Eq. (2) is a balanced transportation problem and both constrains may be changed to equality constraints. If $\sum_{\gamma=1}^{|\Gamma|} d_\gamma > |\mathrm{A}|$, Eq. (2) is infeasible.

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

Subject to:

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

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

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

Traditional shortest path algorithms, e.g., the Dijkstra's algorithm, use node labeling to compute shortest paths. These algorithms manipulate a three-dimensional matrix, $|\mathrm{V}| \times |\mathrm{V}| \times |\Delta|$. Each component in the matrix is travel time from one node to the other when a vehicle leaves the first node during a time interval. These travel times are often obtained through field observations (Rakha *et al.*, 2006). $|\mathrm{V}|$ is the size of the road network, i.e., the total number of nodes. $|\Delta|$ is the number of time intervals. For example, if the size of a road network increases by tenfold, the computer storage space required for the three-dimensional matrix increases by 100 times.

In TDVRPs, most roads allow two-way traffic. During emergencies and evacuations, one-way roads may allow emergency vehicles to travel in both directions. It is assumed that all roads allow two-way traffic in the TDVRP. The degree of a node is the number of roads 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 roads. Because the total number of roads is $\frac{|\mathrm{E}|}{2}$, where $|\mathrm{E}|$ is the number of arcs in the road network, the mean degree is $\frac{\frac{|\mathrm{E}|}{2} \times 2}{|\mathrm{V}|} = \frac{|\mathrm{E}|}{|\mathrm{V}|}$. When arc labeling is used, the TDSP-ARC algorithm manipulates a two-dimensional matrix $|\mathrm{E}| \times |\Delta|$. Each component in $|\mathrm{E}| \times |\Delta|$ is travel time along an arc when a vehicle begins traveling during a time interval. Travel times in $|\mathrm{E}| \times |\Delta|$ are the same as those in $|\mathrm{V}| \times |\mathrm{V}| \times |\Delta|$, but are organized in a different format that reduces required computer storage space. Since $2 \leq \frac{|\mathrm{E}|}{|\mathrm{V}|} \leq 4$, $2|\mathrm{V}| \leq |\mathrm{E}| \leq 4|\mathrm{V}|$. Computer storage space required for the TDSP-ARC algorithm 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-ARC algorithm described below is used to identify the shortest paths, i.e., the earliest arrival times (EATs) of vehicles at destinations. The TDSP-ARC algorithm replaces node labeling in the Dijkstra's algorithm with arc labeling without changing other parts of the Dijkstra's algorithm. The TDSP-ARC algorithm is guaranteed to identify the time-dependent shortest paths between nodes. The algorithm below describes how the shortest path for one vehicle to arrive at a destination is calculated. **Figure 1** expands the algorithm to identify multiple shortest paths in a road network. Steps 1 through 5 in **Figure 1** correspond to Steps 1 through 5 described below.

TDSP-ARC algorithm for identifying the shortest path:

1. Assign to every arc (i, j), $(i, j) \in \mathrm{E}$, in a road network a value representing the arrival time of a vehicle α, $\alpha \epsilon \mathrm{A}$, at j, $i, j \in \mathrm{V}$. For (i, j) in which $i = \beta$, an origin 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, $c_{\beta,j,\delta}^{\alpha}$, 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, $c_{1,2,15}^{\alpha}$, 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 $i \neq \beta$;

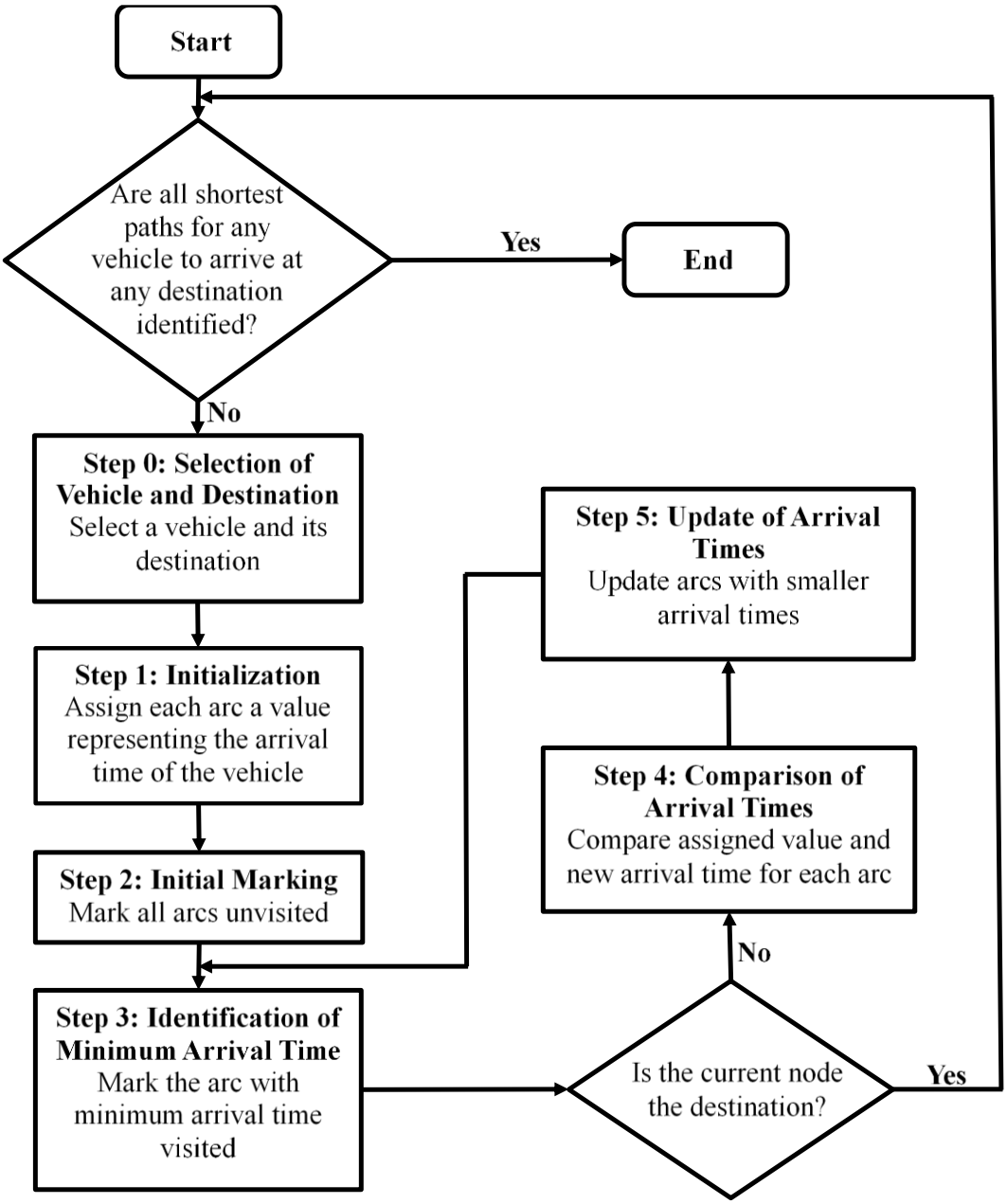

Figure 1 TDSP-ARC Algorithm for Identifying Multiple Shortest Paths

2. Mark all (i,j) unvisited;
3. Identify the unvisited (i,j) with the smallest value. Mark (i,j) visited. Set the end node, i.e., the second node j, in (i,j) as the current node. If j is the desired destination γ, $\gamma \epsilon \Gamma$, stop. The value set for (i,γ) is the EAT of α travelling from β to 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, $c_{i,j,\delta}^{\alpha}$, 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 already set for (i,j). Go to Step 3.

4. ASSIGNMENT PROBLEM: REVISED HUNGARIAN METHOD

After shortest paths are identified using the TDSP-ARC algorithm, the next step is to determine which vehicle is dispatched to which destination to meet the demand. This is a process to assign (dispatch) vehicles to destinations. Different assignments have different total travel time or latest arrival time. In a typical TDVRP, a vehicle is assigned to one destination but a destination may require one or more vehicles. Following previous definitions, there are total $|\mathrm{A}|$ vehicles to be dispatched to $|\Gamma|$ destinations, each of which requires d_γ vehicles, where γ represents a destination, $\gamma = 1, 2, 3, \ldots$ and $\gamma \epsilon \Gamma$, and α represents a vehicle, $\alpha = 1, 2, 3, \ldots$ and $\alpha \epsilon \mathrm{A}$. Let total vehicle demand $D = \sum_{\gamma=1}^{|\Gamma|} d_\gamma$. It is expected that $D \leq |\mathrm{A}|$; otherwise the TDVRP is infeasible.

When the objective is to minimize total travel time for all vehicles, the assignment of vehicles to destinations may be modelled as a GAP and solved using the classic Hungarian method. In the GAP, a square cost matrix must be constructed such that each task is assigned to exactly one agent and each agent is assigned exactly one task (Nauss, 2003). To model the TDVRP as the GAP using the EATs obtained from the TDSP-ARC algorithm, three steps must be followed:

1. Construct a cost matrix $|\mathrm{A}| \times |\Gamma|$. Each component in $|\mathrm{A}| \times |\Gamma|$ represents the EAT of a vehicle at a destination;
2. If a destination γ's demand for vehicles is greater than one, i.e., $d_\gamma > 1$, duplicate the column γ for $d_\gamma - 1$ times. Repeat this process for each destination γ with $d_\gamma > 1$ such that the number of columns in the cost matrix for a destination γ is equal to d_γ. A cost matrix $|\mathrm{A}| \times D$ is constructed at the end of this step;
3. For $|\mathrm{A}| \times D$, if $|\mathrm{A}| > D$, add $|\mathrm{A}| - D$ dummy columns; each component in the dummy columns has a value of zero. The square cost matrix $|\mathrm{A}| \times |\mathrm{A}|$ is constructed.

The Hungarian method may be applied to $|\mathrm{A}| \times |\mathrm{A}|$ to identify the optimal assignment of vehicles to destinations that minimizes total travel time. A more practical goal of the TDVRP, however, is to minimize the latest arrival time for all vehicles; this belongs to the class of bottleneck assignment problem (BAP; Ford and Fulkerson, 1966; Gross, 1959; Ravindran and Ramaswami, 1977). The same square matrix $|\mathrm{A}| \times |\mathrm{A}|$ may be used to identify the optimal solution for the BAP. To the best of the author's knowledge, the Hungarian method was not applied in previous research to solve the BAP. Let $EAT_{\alpha,\gamma'}$ be the component at the intersection of row α and column γ' in the cost matrix $|\mathrm{A}| \times |\mathrm{A}|$. $EAT_{\alpha,\gamma'}$ is the earliest arrival time of vehicle α, $\alpha \in \mathrm{A}$, at γ', $\gamma' \in \Gamma'$, where Γ' is a set of destinations including all destinations in a road network, duplicated destinations, and dummy destinations. $\Gamma \subseteq \Gamma'$ and $|\mathrm{A}| = |\Gamma'|$. The revised Hungarian method minimizes both the latest arrival time and total travel time.

Revised Hungarian method:

1. Find the smallest component in each column of the square cost matrix $|\mathrm{A}| \times |\Gamma'|$, $EAT_{\gamma'}$, where $EAT_{\gamma'} = \min_{\gamma'} EAT_{\alpha,\gamma'}$;
2. Find the largest of the smallest components, $EAT_{max} = \max EAT_{\gamma'}$;
3. Subtract EAT_{max} from all components in $|\mathrm{A}| \times |\Gamma'|$;
4. Draw lines through rows and columns to cover all non-positive components in $|\mathrm{A}| \times |\Gamma'|$. Each line must cover only one entire row or column. Use the minimum number of such lines to cover all non-positive components;
5. Check the number of lines in Step 4. If it is equal to $|\mathrm{A}|$ or $|\Gamma'|$, go to Step 7;
6. Subtract the smallest positive component from all components and go to Step 4;
7. Replace all positive components with infinity ∞;
8. Apply the Hungarian method to find the optimal solution that minimizes total travel time.

Proof of the revised Hungarian method:

The $EAT_{\gamma'}$ calculated in Step 1 is the earliest arrival time at the destination γ'. No vehicle arrives at γ' before $EAT_{\gamma'}$. The EAT_{max} is the earliest time when demand at all destinations is met. The minimum latest arrival time at this point is EAT_{max}. After subtracting EAT_{max} from all components in Step 3, a non-positive component indicates the corresponding vehicle arrives at the destination before or at EAT_{max}. For example, a component of "0" indicates the corresponding vehicle arrives at the destination at EAT_{max}. For another example, a component of "-7" indicates the corresponding vehicle arrives at the destination at $EAT_{max} - 7$. Non-positive components may be part of the optimal solution. Positive components may not be part of the optimal solution because a positive component indicates the corresponding vehicle arrives at the destination after EAT_{max}.

Steps 4 and 5 examine whether feasible solutions having the minimum latest arrival time of EAT_{max} exist. If the minimum number of lines that cover all non-positive components is equal to $|\mathrm{A}|$ or $|\Gamma'|$, feasible solutions that have the minimum latest arrival time of EAT_{max} exist. In other words, each vehicle may be assigned to one destination and each destination may be assigned one vehicle. Step 7 assigns all positive components in $|\mathrm{A}| \times |\Gamma'|$ a value of infinity so that they are excluded from feasible solutions to assure any feasible solution must have the minimum latest arrival time of EAT_{max}. Step 8 applies the classic Hungarian method to minimize total travel time. Indeed, any feasible assignment connecting pairs of vehicles and destinations using the non-positive components in $|\mathrm{A}| \times |\Gamma'|$ is an optimal solution that has the minimum latest arrival time of EAT_{max}, but such optimal solutions must be identified. The classic Hungarian method not only finds an optimal solution but also minimizes total travel time, which is an extra benefit.

If the minimum number of lines drawn in Step 4 that cover all non-positive components is less than $|\mathrm{A}|$ or $|\Gamma'|$, feasible solutions that have the minimum latest arrival time of EAT_{max} do not exist. The EAT_{max} must be updated (increased) incrementally to admit feasible solutions. Step 6 updates EAT_{max} by subtracting the smallest positive component from all components in $|\mathrm{A}| \times |\Gamma'|$. For example, if the smallest positive component is 8, the EAT_{max} is updated as follows: $EAT_{max} = EAT_{max} + 8$. Step 6 repeats until the minimum number of lines drawn in Step 4 is equal to $|\mathrm{A}|$ or $|\Gamma'|$. At that time feasible solutions that have the minimum latest arrival time of EAT_{max} exist and Steps 7 and 8 are performed to find the optimal assignment.

The cost matrix $|\mathrm{A}| \times |\Gamma'|$ is updated in three steps: Step 3, Step 6, and Step 7. Steps 3 and 6 subtract a value from all components in $|\mathrm{A}| \times |\Gamma'|$, which does not have any impact on the optimal assignment. Step 7 excludes positive components from any feasible solution because positive components are not part of an optimal assignment. The revised Hungarian method therefore minimizes the latest arrival time and total travel time.

This completes the proof.

5. COMPUTATION TIME EFFICIENCY OF THE TDVRP ALGORITHM

In real time transportation planning and emergence response, the TDVRP algorithm must be solved within a reasonable amount of time to support timely decision making. For example, in a no-notice evacuation, a large population must be evacuated to a safe area within a time period ranging from a couple of hours to days. Vehicles stationed at multiple yards (origins) must be dispatched within a planning period to various intersections and ramps to control traffic and facilitate evacuation. Many factors affect how fast the TDVRP algorithm may be executed to find the optimal solution. Among these factors, computer tools and complexity of the TDVRP play a major role in the computation time efficiency of the TDVRP algorithm.

As the size of a road network increases, computation time of the TDVRP algorithm increases (Harwood *et al.*, 2013). It is expected that the density of a road network also affects time efficiency. A dense road network has more roads (arcs) and may require more computation time to find the shortest paths and assign vehicles to destinations. On the other hand, a variety of computer programs and tools may be used to implement the TDVRP algorithm, e.g., Visual Basic for Applications (VBA), Visual Basic (VB), and parallel computing. The next two subsections examine the impact of road networks and computer tools on the computation time efficiency of the TDVRP algorithm.

5.1 Road networks and computation time efficiency of the TDVRP algorithm

The objective of the experiments is to determine the computation time of the proposed algorithms when the size and density of road networks change. Four road networks in the City of San Francisco (Brinkhoff, 2002) are used to examine the computation time efficiency of the TDVRP algorithm. **Table 2** summarizes the four networks, I, II, III, and IV. Two factors of a road network, Size and Density, are studied. Each factor has two levels: Size may be small or large and Density may be sparse or dense. The Size of a network is represented by the total number of nodes, and Density of a network is represented by the mean degree. Thirty-eight experiments are performed for each road network. In total, 152 experiments (= 38 x 4) are completed. In each experiment, the TDVRP algorithm is implemented in VBA to find the optimal vehicle assignments that minimize the latest arrival time and total travel time. Computation times of the algorithm are recorded in **Table 3**. In each experiment, 100 vehicles are uniformly and randomly stationed at origins. The number of origins is uniformly and randomly chosen between one and 100. There are 20 destinations each of which requires two vehicles. Locations of origins and destinations are uniformly and randomly chosen from nodes in a road network.

The planning period is 600 minutes with one minute time interval. As defined in Section 3, a planning period is a time period during which vehicles must be dispatched to destinations to meet the demand. While the planning period may be large, the minimum latest arrival time can be small (less than 30 minutes in this case) depending on the geographical region and traffic of the road network. A large planning period provides a sufficiently large solution base from which the minimum latest arrival time may be identified.

Computation times are analyzed using SAS 9.4 (SAS Institute Inc., 2014). The normal probability plot of residuals is approximately linear, indicating residuals conform to normality assumption. Residual is arithmetic difference between actual computation time and computation time predicted by the regression model. The 2-way analysis of variance (ANOVA) procedure shows that main effects, Size and Density, and their interaction are significant (P-value < 0.001). The interaction plot (**Figure 2**) indicates there is an interaction between Size and Density. When Size is large, Density has a substantial positive effect on computation time whereas when Size is small, Density has a minor positive effect on computation time.

The interaction between Size and Density may partially attribute to the TDSP-ARC algorithm, which uses arc labelling to calculate shortest paths. When Size is small, the difference in Density does not have a significant impact on the number of arcs. For example, the numbers of arcs in I, a small and sparse road network, and II, a small and dense road network, are 571 and 828, respectively. Since the increase in the number of arcs in II compared to I is small (257 arcs), the computation time of the TDSP-ARC algorithm increases slightly. When Size is large, the difference in Density has a significant impact on the number of arcs. For example, the numbers of arcs in III, a large and sparse road network, and IV, a large and dense road network, are 4,375 and 6,299, respectively. The large increase in the number of arcs in IV compared to III (1,924 arcs) leads to a significant increase in computation time of the TDSP-ARC algorithm. Other parts of the TDVRP algorithm also contribute to its computation time and these are discussed in the next section.

Table 2 Four Road Networks

Network	Total Number of Nodes	Size	Mean Degree	Density
I	512	Small	2.23	Sparse
II	516	Small	3.21	Dense
III	3,916	Large	2.23	Sparse
IV	3,926	Large	3.21	Dense

Table 3 Computation Times (seconds) of the TDVRP Algorithm for the Four Road Networks Described in Table 2

Experiment	Network				Experiment	Network			
	I	II	III	IV		I	II	III	IV
1	64	105	1,481	2,972	20	59	80	1,555	3,058
2	58	97	1,512	2,986	21	57	78	1,552	3,108
3	66	106	1,521	3,000	22	56	93	1,552	3,057
4	63	87	1,515	2,963	23	63	100	1,530	3,104
5	66	100	1,510	2,958	24	60	82	1,505	3,049
6	52	87	1,492	2,968	25	68	69	1,529	2,966
7	56	98	1,531	3,004	26	54	108	1,548	3,060
8	54	95	1,493	3,023	27	55	103	1,579	3,158
9	47	105	1,564	3,008	28	51	101	1,495	3,103
10	45	98	1,486	2,996	29	54	102	1,587	2,959
11	58	92	1,543	2,931	30	56	100	1,580	3,037
12	55	169	1,477	3,162	31	48	90	1,561	3,111
13	56	114	1,454	2,934	32	57	82	1,561	3,106
14	55	101	1,514	2,942	33	61	106	1,638	3,154
15	58	104	1,484	2,997	34	58	78	1,551	3,138
16	70	100	1,463	3,017	35	56	99	1,571	3,045
17	66	94	1,522	3,071	36	61	102	1,530	3,168
18	50	91	1,512	2,956	37	69	90	1,525	2,916
19	58	95	1,524	2,990	38	55	87	1,504	3,117

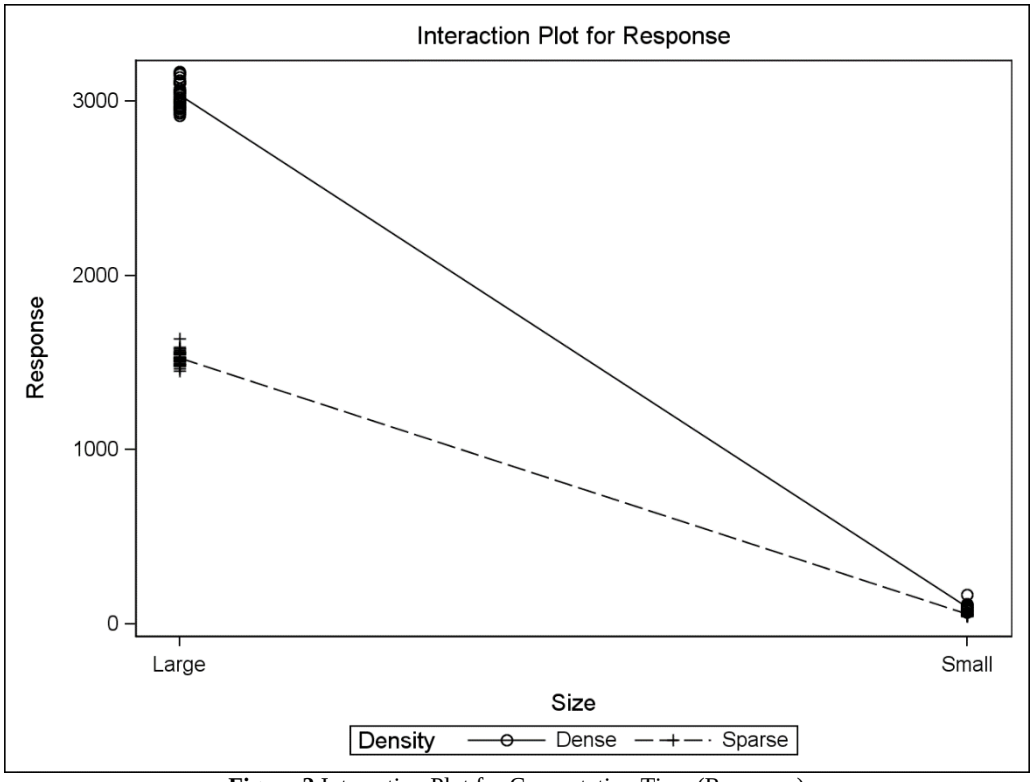

Figure 2 Interaction Plot for Computation Time (Response)

5.2 Computer tools and computation time efficiency of the TDVRP algorithm

The computation time of the TDVRP algorithm comprises of three parts: data processing (including reading data from the cost matrix, storing temporary data, and results output), TDSP-ARC algorithm, and revised Hungarian method. Three computer programs and tools, VBA, VB, and VB with parallel computing are used to implement the TDVRP algorithm for road network IV, the large and dense road network. The objective is to determine which computer tool is the most time efficient for the TDVRP algorithm. **Table 4** shows the breakdown of the computation time of the TDVRP algorithm in five experiments using VBA.

It takes a little less than an hour to execute the TDVRP algorithm using VBA for a large and dense road network with 3,926 nodes, 6,299 arcs, and a mean degree of 3.21. The column "Mean" in **Table 4** calculates the mean computation time in five experiments. The data processing time is about 4% of the total computation time. The computation time of the TDSP-ARC algorithm is about 91% of the total computation time. The computation time of the revised Hungarian method is about 5% of the total computation time. The computation time in VBA for large and dense road networks is too large for real time vehicle routing. **Table 5** shows the breakdown of the computation time of the TDVRP algorithm in five experiments using VB .NET Framework in Microsoft Visual Studio 2013.

Compared to the total computation time in VBA, the total computation time of the TDVRP algorithm in VB decreases significantly; it takes a little over six minutes to execute the TDVRP algorithm using VB for a large and dense road network. The data processing time in VB is greater than that in VBA. VBA is embedded in Microsoft Excel 2013 and the cost matrix is also stored in Microsoft Excel 2013. The seamless integration of VBA and Microsoft Excel 2013 leads to a smaller data processing time in VBA. Both the computation times for the TDSP-ARC algorithm and revised Hungarian method, however, decrease significantly in VB. Based on the column "Mean" in **Table 5**, the data processing time is about 37% of the total computation time. The computation time of the TDSP-ARC algorithm is about 62% of the total computation time. The computation time of the revised Hungarian method is only 1% of the total computation time. The total computation time in VB is acceptable for most real time TDVRPs.

A major development contributing to the acceleration and quality of algorithms for real time decision support is parallel computing (Ghiani *et al.*, 2003). The TDVRP algorithm is implemented in VB with parallel computing. **Table 6** shows the breakdown of the computation time of the TDVRP algorithm in five experiments using VB with parallel computing. Compared to the total computation time in VB, the total computation time in VB with parallel computing is smaller but with a larger standard deviation, indicating performance of parallel computing is not stable. The data processing time decreases significantly whereas the computation time of the TDSP-ARC algorithm increases in VB with parallel computing compared to those in VB without parallel computing. The computation time of the revised Hungarian method is almost the same in VB with or without parallel computing.

Based on the column "Mean" in **Table 6**, the data processing time is about 2% of the total computation time. The computation time of the TDSP-ARC algorithm is about 97% of the total computation time. The computation time of the revised Hungarian method is only 1% of the total computation time. Compared to VB without parallel computing, VB with parallel computing significantly reduces data processing time but increases the computation time of the TDSP-ARC algorithm; both have similar total computation time. Parallel computing requires a high level of coordination and synchronization of resources in the computer operating system. The overhead of using parallel computing is expensive (Barney, 2010). Within the TDVRP algorithm, the TDSP-ARC algorithm requires the most complex calculations, which explains why the computation time of the TDSP-ARC algorithm increases in VB with parallel computing. **Figure 3** compares the mean computation times of the three different computer tools: VBA, VB, and VB with parallel computing. VB with or without parallel computing should be used to execute the TDVRP algorithm for real time vehicle routing in large road networks.

Table 4 Computation Time Breakdown (seconds) for the TDVRP Algorithm Implemented in VBA

Classification of Computation Time	Experiment					Mean	Standard Deviation
	1	2	3	4	5		
Data Processing	114	113	110	116	114	113	2
TDSP-ARC Algorithm	2,885	2,950	2,875	2,910	2,820	2,888	48
Revised Hungarian Method	169	170	156	163	177	167	8
Total	3,168	3,233	3,141	3,189	3,111	3,168	46

Table 5 Computation Time Breakdown (seconds) for the TDVRP Algorithm Implemented in VB

Classification of Computation Time	Experiment					Mean	Standard Deviation
	1	2	3	4	5		
Data Processing	145	145	144	144	144	144	< 1
TDSP-ARC Algorithm	238	239	242	244	238	240	3
Revised Hungarian Method	3	2	3	3	3	3	< 1
Total	385	386	389	390	385	387	2

Table 6 Computation Time Breakdown (seconds) for the TDVRP Algorithm Implemented in VB with Parellel Computing

Classification of Computation Time	Experiment					Mean	Standard Deviation
	1	2	3	4	5		
Data Processing	8	7	7	8	7	7	< 1
TDSP-ARC Algorithm	350	398	385	320	310	353	39
Revised Hungarian Method	2	3	2	2	3	2	< 1
Total	360	408	394	330	320	362	39

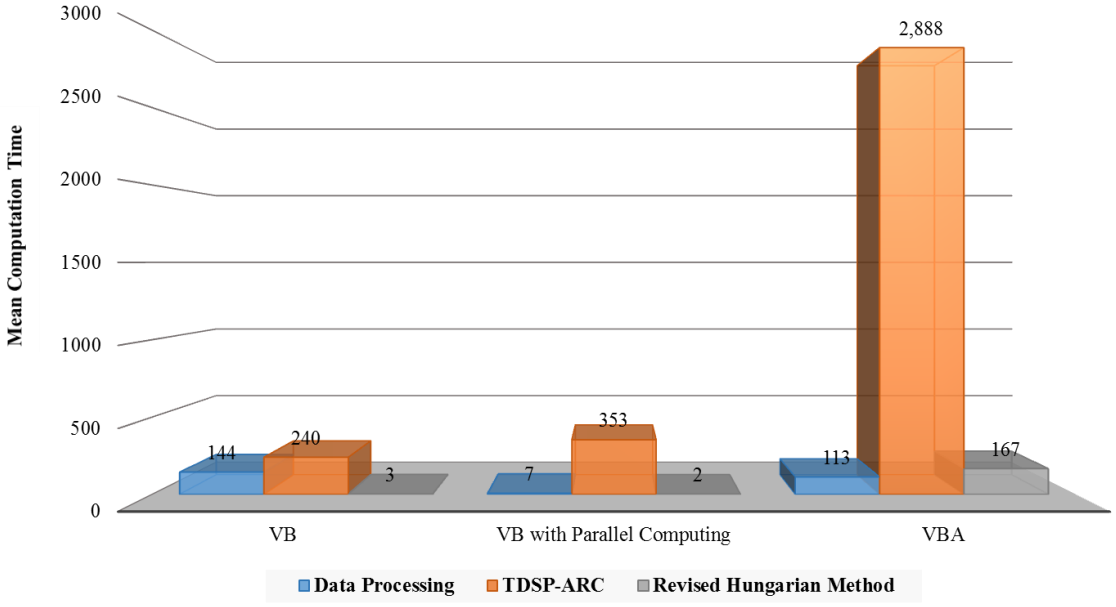

Figure 3 Mean Computation Times of the TDVRP Algorithm

6. CONCLUSIONS

This article develops an improved TDVRP algorithm, which identifies the optimal vehicle routes and assignments in road networks. The TDVRP algorithm integrates the TDSP-ARC algorithm and a new Hungarian method, which is revised based on the classic Hungarian method. The TDVRP algorithm minimizes the latest arrival time and total vehicle travel time. Experiments results show that the TDVRP algorithm implemented in VB finds the optimal solution in six minutes for large and dense road networks. The TDVRP algorithm implemented in VB may be used for real time time-dependent vehicle routing.

The TDVRP algorithm may be further validated with other large road networks. If the computation time is too large, the implementation of the algorithm in VB should be improved, or other programming languages can be used to shorten the computation time. The TDVRP algorithm efficiently identifies the optimal routes and assignments for a class of VRPs in which multiple vehicles located at different locations are dispatched to multiple destinations. For other types of VRPs, e.g., VRPs with time windows, the algorithm developed in this article may be customized to find optimal routes and assignments.

7. FUTURE RESEARCH

Future research may validate the TDVRP algorithm using other road networks and computer tools. Abundant data for various road networks are available online (e.g., Brinkhoff, 2002). The TDVRP algorithm developed in this article may be further tested for time efficiency using these road networks. Other computer tools, e.g., C++ and Java, may be used to improve computation time efficiency. Although parallel computing tested in this research does not have a dominant advantage, it remains an effective tool to improve computation time efficiency of the TDVRP algorithm. For example, parallel computing using multiple computers may be tested in future research. In addition, super computers may be used to test the TDVRP algorithm.

ACKNOWLEDGEMENTS

The author would like to thank Zhu Zhang for conducting the experiments and helping prepare this article with the support of the Southern Illinois University Edwardsville Research Grants for Graduate Students.

REFERENCES

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

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

Alvarez, A., and Munari, P. (2017). An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen. *Computers and Operations Research* 83, pp. 1 - 12.

Balas, E., and Toth, P. (1985). Branch and bound methods. *The traveling salesman problem*, Lawer *et al.* (Eds.), John Wiley & Sons, 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.

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

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

Chakroborty, P., and Mandal, A. (2005). An asexual genetic algorithm for the general single vehicle routing problem. *Engineering Optimization* 37 (1), pp. 1 - 27.

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

Chen, B. Y., Lam, W. H. K., Sumalee, A., Li, Q, Shao, H., and Fang, Z. (2013). Finding reliable shortest paths in road networks under uncertainty. *Networks and Spatial Economics* 13, pp. 123 - 148.

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

Defryn, C., & Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers & Operations Research*, *83*, pp. 78-94.

Drexl, M. (2012). Rich vehicle routing in theory and practice. *Logistics Research* 5, pp. 47 - 63.

Euchi, J., and Chabchoub, H. (2010). A hybrid tabu search to solve the heterogeneous fixed vehicle routing problem. *Logistics Research* 2, pp. 3 - 11.

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), pp. 616 - 636.

Ford, L. R. Jr., and Fulkerson, D. R. (1966), *Flows in Networks*, Princeton University Press, Princeton, New Jersey.

Gendreau, M., Laporte, G., and Yelle, S. (1997). Efficient routing of service vehicles. *Engineering Optimization* 28 (4), pp. 263 - 271.

Gendreau, M., Ghiani, G., and Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & Operations Research* 64, pp. 189 - 197.

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

Gross, O. (1959). The bottleneck assignment problem. *Proceedings of the RAND Symposium on Mathematical Programming (Linear Programming and Extensions)*, P-1630.

Haghani, A., and Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research* 32 (11), pp. 2959 - 2986.

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

Harwood, K., Mumford, C., and 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), pp. 34 - 47.

Ichoua, S., Gendreau, M., and Potvin, J. Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* 144 (2), pp. 379 - 396.

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

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

Küçükoğlu, İ., and Öztürk, N. (2014). A differential evolution approach for the vehicle routing problem with backhauls and time windows. *Journal of Advanced Transportation* 48 (8), pp. 942 - 956.

Lai, M., and Tong, X. (2012). A metaheuristic method for vehicle routing problem based on improved ant colony optimization and Tabu search. *Journal of Industrial and Management Optimization* 8 (2), pp. 469 - 484.

Lai, M., Yang, H., Yang, S., Zhao, J., and Xu, Y. (2014). Cyber-physical logistics system-based vehicle routing optimization. *Journal of Industrial and Management Optimization* 10 (3), pp. 701 - 715.

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

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

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science* 43, pp. 408 - 416.

Lecluyse, C., Van Woensel, T., and Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times. *4OR-Q Journal of Operations Research* 7 (4), pp. 363 - 377.

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

Nauss, R. M. (2003). Solving the generalized assignment problem: An optimizing and heuristic approach. *INFORMS Journal on Computing* 15 (3), pp. 249 - 266.

Ozsoydan, F. B., and Sipahioglu, A. (2013). Heuristic solution approaches for the cumulative capacitated vehicle routing problem. *Optimization: A Journal of Mathematical Programming and Operations Research* 62 (10), pp. 1321 - 1340.

Rakha, H., El-Shawarby, I., Arafeh, M., and Dion, F. (2006). Estimating path travel time reliability. *Proceedings of 2006 IEEE Intelligent Transportation Systems Conference*, Toronto, Canada, pp. 236 - 241.

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

Rekersbrink, H., Makuschewitz, T., and Scholz-Reiter, B. (2009). A distributed routing concept for vehicle routing problems. *Logistics Research* 1, pp. 45 - 52.

SAS Institute Inc. (2014), SAS 9.4.

Spliet, R., and Gabor, A. F. (2012). The time window assignment vehicle routing problem. *Erasmus School of Economics (ESE)*, No. EI 2012-07, pp. 1 - 19.

Van Woensel, T., Kerbache, L., Peremans, H., and Vandaele, N. (2008). Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research* 186 (3), pp. 990 - 1007.

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

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2014). A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research* 234 (3), pp. 658 - 673.

Wilhelm, W. E. (2001). A technical review of column generation in integer programming. *Optimization and Engineering* 2 (2), pp. 159 - 200.

Xin Chen, Ph.D., is an Associate Professor of Industrial Engineering at Southern Illinois University Edwardsville. His research focuses on network- and knowledge-centric collaborative control with applications in air and airport traffic control, critical infrastructure protection, energy distribution, social networks, and supply chains. He received a B.S. in Mechanical Engineering from Shanghai Jiao Tong University, an M.S. and a Ph.D. in Industrial Engineering from Purdue University.