OPTIMAL ROUTE DETERMINATION FOR POSTAL DELIVERY USING ANT COLONY OPTIMIZATION ALGORITHM

[1]Babatunde A. O, [2]Oladipo I. D., [3]Busari A. O., [4]Balogun G. B., [5]Taofeek-Ibrahim F. A., [6]Sheu M. O.

[1,2,3,4]Department of Computer Science, University of Ilorin, Ilorin, Nigeria
[5,6]Department of Computer Science, Federal Polytechnic Offa
[6]Department of Computer Science, Institute of Information and Communication Technology, Kwara State Polytechnic, Ilorin
[1]babatunde.ao@unilorin.edu.ng, [2]odidowu@unilorin.edu.ng; [3]ahmadinjo@gmail.com;
[4]balogun.gb@unilorin.edu.ng; [5]fatty_fatty2@yahoo.com.au, *[6]sheumusaolojeola@gmail.com*

ABSTRACT

There are a lot of optimization challenges in the world, as we all know. The vehicle routing problem is one of the more complex and high-level problems. Vehicle Routing Problem is a real-life problem in the Postal Delivery System logistics and, if not properly attended to, can lead to wastage of resources that could have been directed towards other things. Several studies have been carried out to tackle this problem using different techniques and algorithms. This study used the Ant Colony Optimization Algorithm along with some powerful APIs to find an optimal route for the delivery of posts to customers in a Postal Delivering System. When Ant Colony Optimization Algorithm is used to solve the vehicle routing problem in transportation systems, each Ant's journey is mere "part" of a feasible solution. To put it in another way, numerous ants' pathways might make up a viable solution. Routes are determined for a delivery vehicle, with the objective of minimizing customer waiting time and operation cost. Experimental results indicate that the solution is optimal and more accurate

Keywords: *Ant Colony, algorithm, route, postal delivery, optimization.*

INTRODUCTION

The Postal Delivery Systems (PDSs) are extensive networks with many moving parts. These networks create a variety of issues for decision-makers in logistics and express services. Indeed, the regional postal network is highly complex due to the large population of nodes, extended distances, large territories, intricate flow directions, and flow amounts, and its administration is critical (Fanti, Laraspata, Iacobellis, Mangini, Ukovich, &Abbatecola, 2014).

According to the Council of Logistics Management, CLM 2001, Planning, implementing, and controlling the effective, appropriate movement from the point of origin to the point of consumption to satisfy customers' demands and storage of such products, services, and associated information constitute the logistics component of the supply chain process. However, logistics activities in urban areas have distinct characteristics that distinguish them from general logistics activities, which is why freight transport in urban areas specifically freight flows associated with the supply of goods to city centres, is commonly referred to as "city logistics". City Logistics is defined by Taniguchi, Thompson, and Yamada (2001) as "the process of completely optimizing logistics and transportation activities by private companies in urban areas while taking into account within the context of a market economy, the traffic environment, traffic congestion, and energy usage". The distribution of costs in the chain of freight transportation also demonstrates the

relevance of urban freight transportation. Deliveries and pick-ups activities, which are common in urban areas, account for around 40% of total door-to-door costs in combined transportation. (Inner-City Freight Transportation). The weight of these expenditures is increased by reducing stocks, the decrease in consignment size and the growth in volume.

Vehicle routing, location routing, network architecture, and crew assignment are all issues that the Postal Delivery System must deal with (Song, Wang, Li, & Zhang, 2009). Thus, describing and optimizing decision problems in the Postal Delivery System entails complex tasks, and NP-hard combinatorial optimization problems (Ye et al., 2004).

METHODOLOGY

In the past, many scholars have made great efforts to solve the VRP, and both exact and heuristic methods have been suggested. For example, scientists have created exact algorithms based on integer programming (Almoustafa, Hanafi, &Mladenovi, 2013), column generation algorithm (Liberatore, Righini, & Salani, 2011), branch-and-bound (Fischetti, Toth, & Vigo, 1994), branch-and-price (Lysgaard &Wohlk, 2014) and Bellman-Ford algorithm (Rivera, Afsar, & Prins, 2015).

An implementation of the integer L-shaped method to arrive at the correct answer is investigated in (Laporte, Louveaux, & Hamme, 2002), which is heavily reliant on the computation of lower bounds and the construction of more efficient lower bounding functionals. Their paper looks at a scenario where some of the demands are stochastic. This means that amount of demand at each client is unknown before their arrival. Even if the predicted demand for the route does not surpass the vehicle's capacity, the vehicle might not be able to accommodate the customer's requirement every time. A failure is a term used to describe a situation like this. The capacitated vehicle routing problem with stochastic demands (SVRP) then requires reducing both the total cost of the routes proposed and the cost of failures anticipated. Failures result in visits back to the depot as a result of the penalty. The van is rerouted back to the depot to unload first before resuming its journey as scheduled. The implementation of the Integer L-shaped approach for the exact solution of the SVRP is investigated in this paper. Numerical studies show that some cases with up to 100 customers and a small number of cars can be solved to perfection in a relatively short amount of time.

As for the heuristic approaches, Prins, Labadi, and Reghioui (2009) propose a tour splitting heuristic, which first designs a "large" trip that stops at every consumer, then divides the tour into capacity-feasible vehicle trips. Because of their reputation for poor performance, they are rarely utilized alone. Prins, Labadi, and Reghioui (2009), in their paper describe how to make them better to get better answers or deal with more restrictions. Randomized versions outperform classical constructive algorithms in numerical testing on the capacitated arc routing problem (CARP) and the capacitated vehicle routing problem (CVRP). Based on these ideas, a greedy randomized adaptive search procedure (GRASP) and an iterated local search (ILS) can compete with present metaheuristics while being more efficient and straightforward.

Campbell and Savelsbergh (2004) propose an insertion heuristic and study the effect that adding complicated restrictions has on how effective insertion heuristics are. The basic insertion heuristic for the standard vehicle routing problem has a time complexity of $O(n^3)$. However, straightforward implementations of handling complicated constraints result in a time complexity of $O(n^4)$ that is undesirable. They show that with careful

implementation, it is possible to keep the $O(n^3)$ complexity in most situations or, in a few circumstances, increase the time complexity to $O(n^3 \log n)$ in a few cases.

Gong, Zhang, Liu, Huang, Chung, and Shi (2012) propose a set-based particle swarm optimization (PSO) algorithm in which the operators are defined on the set. By using the PSO framework, an optimal subgraph could be selected out of the complete graph, and the problem is solved. They suggest a set-based PSO in addressing the discrete combinatorial optimization problem in their study, although the particle swarm optimization (PSO) algorithm was created to solve continuous optimization problems. The standard approach is to use the PSO framework to choose an optimum subset from the universal set. As a result, instead of the arithmetic operators used in the original PSO method, the algorithm's operators are defined on the set. Furthermore, the algorithm's position updating process is constructive, as it takes into account the problem's restrictions and employs a time-oriented nearest neighbour heuristic. In order to address the primary and secondary objectives of the problem, a normalizing mechanism is proposed.

In Lau, Chan, Tsui, and Pang (2010), the guide of fuzzy logic is added to the genetic algorithm (GA) to dynamically adjust the crossover rate and mutation rate of GA. Their study discusses the optimization of vehicle routing issues that take into account numerous depots, multiple consumers, and different items. Because total travelling time is not necessarily a time window constraint, the aim considered in their research includes not only the cost of total travelling distance but also the cost of total travelling time. They propose to use a stochastic search technique called fuzzy logic guided genetic algorithms (FLGA) to solve the problem. The purpose of fuzzy logic is to alter the crossover and mutation rates dynamically after ten generations.

Experimental (Material and Methods)

Generally, VRP can be roughly defined as finding a route or numerous routes that connect a depot to a crowd of clients while keeping the overall cost as low as possible. In previous decades, the majority of articles used an undirected graph $G = (V, E)$ to establish a mathematical model. In the model, $V = \{V_0, V_1, \ldots, V_n\}$ represents the vertex set and $E = \{(V_i, V_j) \mid V_i, V_j \in V, i<j\}$ is an edge set. A set of m homogeneous automobiles (having a constant and consistent capacity Q) leave from a single point of origin, which is represented by the vertex V_0, and must pay a visit to all of the customers who are represented by n vertexes $\{V_1, \ldots, V_n\}$. In E, we calculate the distance of customers V_i and V_j and get distance matrix $C = (c_{ij})$. Every customer V_i has a demand q_i and needs to be visited just once by only one vehicle. V is divided into m routes $\{R_1, \ldots, R_m\}$ that include all customers. The distance of route $R_i = \{V_0, V_1, \ldots, V_{k+1}\}$ where $V_k \in V$ and the depot $V_{k+1} = V_0$ is calculated by

$$\text{Cost}(R_i) = \sum_{j=0}^{k} c_j, j + 1 \quad (1)$$

and calculating cumulative Cost(R_i) to the total cost of solutions S is as follows:

$$f_{\text{VRP}}(S) = \sum_{i=1}^{m} Cost(R_i) \quad (2)$$

Overall, the VRP needs to observe the following constraints:

i. Each vehicle departs from the same depot and returns to the depot.

ii. Each customer is only to be visited by one vehicle at a time.

iii. Each vehicle has a capacity limit (assuming all vehicles are the same model).

iv. The vehicle arrives, and the service time must meet the deadline.

A one-dimensional array of customers is used to represent the route. The delivery vehicle delivers through this route in the order they appeared in the array.

Ants deposit pheromone on arcs of a node-arc graph, which is the basis for the ACO algorithm, so that the shortest-distance arcs between the source and destination nodes receive more pheromone, leading to more ants picking those arcs in subsequent rounds. Ants' likelihood of selecting certain nodes to visit is based on:

i. The pheromone content of an arc and

ii. The length of the arc.

The quantity of pheromone each ant leaves behind on the arcs of its route are regulated by pheromone update rules once the tour is completed. The pseudo-code described later in this section depicts the algorithm's structure and includes:

i. an outer loop relating to cycles of evolutionary pheromone updating,

ii. An inner loop executing for all node-arc (time) steps used, and

iii. The innermost loop involves the calculation of node-choice probabilities and tour-nodes updates.

Algorithmic skeleton for ACO algorithms applied to combinatorial optimization problems. The application of a local search algorithm is a typical example of a possible daemon action in ACO algorithms (Dorigo &Stützle, 2019):

procedure ACO algorithm for combinatorial optimization problems

Initialization

while (termination condition not met) do

ConstructAntSolutions

ApplyLocalSearch % optional

GlobalUpdatePheromones

end

end ACO algorithm for combinatorial optimization problems

The main loop consists of three main phases after initializing settings and pheromone trails. First, m ants build solutions to the problem instance in question, influenced by pheromone information and possibly available heuristic information. After the ants have finished their solutions, they can choose to improve them in a local search phase. Finally, before the next iteration begins, the pheromone trails are modified to reflect the ants' search experience. The ACO metaheuristic's main steps are detailed in the following sections below.

Initialization. The value of all pheromone variables is set to τ_0, which is a parameter of the algorithm at the start of the process.

ConstructAntSolutions. A group of *m* ants creates solutions to the problem instance in question. To do so, each Ant begins with an originally empty solution $s_p = \emptyset$. An ant expands its present partial solution s_p at each construction stage by deciding on a single viable solution component $c_i^j \in N(s_p) \subseteq C$ and incorporating it into the partial solution currently in place. $N(s_p)$ is the set of solution components that can be added while keeping the solution feasible, as determined implicitly by the ants' solution construction process. If a partial solution cannot be extended while being feasible, the solution is either abandoned or rendered infeasible, in which case a complete solution is constructed, depending on the building technique. In

the latter situation, infeasible solutions may be penalized based on how much the problem constraints are violated.

At each building step, the decision on which solution component to add is made probabilistically. The probability distributions have been defined in a variety of ways. The Ant System (AS) is the most often utilized rule (Dorigo et al., 1996):

$$p\left(c_j^i \middle| s_p\right) = \frac{\tau_{ij}^{\alpha} \cdot \left[\eta\left(c_i^j\right)\right]^{\beta}}{\sum_{c_l^l \in N(s_p)} \tau_{il}^{\alpha} \cdot \left[\eta\left(c_i^l\right)\right]^{\beta}}, \forall c_i^j \in N\left(s_p\right) \quad (3)$$

where $\eta(\cdot)$ is a function that assigns a heuristic value η_{ij} to each feasible solution component $c_i^j \in N\left(s_p\right)$, which is usually called, heuristic information. The algorithm behaviour is influenced by parameters α and β, which govern the relative importance of pheromone trails and heuristic information as follows; If α = 0, the selection probabilities are proportional to $\left[\eta_{ij}\right]^{\beta}$ and a solution component with a high heuristic value will more likely be selected: this scenario corresponds to a stochastic greedy algorithm. If β = 0, pheromone amplification is the only thing going on.

ApplyLocalSearch. After obtaining complete candidate solutions, local search techniques might be used to optimize them further. In fact, when combined with local search algorithms, ACO algorithms provide the best results for a wide range of combinatorial optimization problems (Dorigo, Maniezzo, & Colorni, 1996). Local search is an example of what has been dubbed "daemon actions" in general (Dorigo & Caro, 1999). These are employed to carry out problem-specific or centralized tasks that individual ants cannot carry out.

GlobalUpdatePheromones. The purpose of the pheromone update is to make solution components from good solutions more appealing in subsequent iterations. To achieve this purpose, there are essentially two systems in play. The first is the pheromone deposit, which raises the level of the pheromone of solution components linked to a certain set S_{upd} of good solutions. In subsequent stages, the idea is to make these solution components more appealing to ants. The second mechanism is pheromone trail evaporation, which is the mechanism that causes the pheromone deposited by preceding ants to degrade over time. Practically, pheromone evaporation is necessary to stop the algorithm from rapidly converging to a sub-optimal region. It makes use of a useful type of forgetting that promotes the identification of fresh search territories. The pheromone update is commonly implemented as:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s) \quad (4)$$

where S_{upd} is the set of solutions that are used to deposit pheromone, $\rho \in (0,1]$ is a parameter called evaporation rate, $g(\cdot): S \mapsto \boldsymbol{R}^{+}$ is a function such that:

$$f(s) < f(s) \Rightarrow g(s) \geq g(s).$$

It is usually referred to as the evaluation function because it determines the quality of a solution.

SQLite is the database storage technology that is being used in this project. It's a contemporary database system that's utilized to run databases. It's widely utilized on the web and other server-based applications. Because it employs a standard Structured Query Language (SQL) for accessing and manipulating databases, it is excellent for both small and large numbers of applications.

A self-contained, serverless, and configuration-free transactional SQL database engine is produced by the in-process library known as SQLite. The source code for SQLite is open source., which means it can be used for any purpose, commercial or personal.

The most widely used database worldwide is SQLite, has an infinite number of uses, some of them are for high-profile initiatives.

SQLite is an SQL database engine for embedded systems. SQLite does not have a separate server process like most other SQL databases. SQLite reads and writes to regular disk files directly. A complete SQL database, comprising several tables, indices, triggers, and views, is included on a single disk file. You can freely copy a database across different platforms or between big-endian and little-endian architectures because the database file format is cross-platform. These features make SQLite a popular Application File Format. SQLite database files are suggested by the US Library of Congress as a storage format.

RESULTS AND DISCUSSION

The Ant Colony Optimization Algorithm (ACO) was implemented in Java, an object-oriented programming language. In order to enhance the development, editing, and proper debugging of the equivalent java source code, IntelliJ IDEA 2019 Integrated Development Environment (IDE) tool was employed. The java implementation is driven by four major classes (Ant, ACO, DBDriver, Customer, and Route classes), of which the DBDriver class is the executable class containing the main method and a class, CreateDB Class, for storing, retrieving, and managing the data needed for the solution computation. The Ant class is a model class of the natural Ant that holds the attributes and behaviours (methods) of an Ant (Solution). The ACO class implements the steps in the Ant Colony Optimization Algorithm in Java. The DB class handles the establishment of a connection with the database.

The Ant class models the Ants (agents for finding optimum route) as objects containing attributes to initialize the Ants, control the pheromone features of the Ants and their corresponding methods for each Ant Route construction.

The Route Class models a route object containing one dimensional array of customers in a particular order to form routes (solutions) as found by individual Ants.

The Ant Colony Optimization Algorithm (ACO) class implements each step in the algorithm as a series of modules or methods. It works on the objects of the Ant Class by performing optimization and improvements on the routes found by the ants. It serves the following functions:

i. Creates Matrix to keep track of the pheromone levels between customers' locations.

ii. Creates Matrix to keep track of the distances between customers' locations.

iii. Initializes the initial route and customers size (i.e., number of customers).

iv. It exposes methods to get both the pheromone levels and distances between customers' locations.

v. It also exposes methods to initialize the distances and the pheromone levels between customers' locations.

The Customer class is a container for the customer, one of the critical resources of this software project. It models a customer as an object containing the customer’s name and location as attributes.

The CreateDB class is used to connect to a database or create one if it does not exist. It also creates and fills the customers' table with the necessary information needed for the algorithm.

The DBDriver class is a class containing the main method where inputs are retrieved from the database and supplied to the Ant Colony Optimization Algorithm, multi-objective optimization operations

(modules) of the algorithm are being invoked, as well as displaying relevant results (output). In this class, the details of the program problem to be solved are initialized appropriately, and the Ant Colony Optimization Algorithm (ACO) is applied to the problem to obtain a near-optimal solution.

The input data to the program are:

i. The customers, their names and locations as pairs of latitude and longitude for better accuracy.

ii. The variable that is used to modify the quantity of pheromones deposited along each route (edges).

iii. The variable that is changed to alter the rate of pheromone evaporation along each pathway.

iv. The variable used to regulate the pheromone trail's significance.

v. The variable used to regulate how important the gap between source and destination is.

vi. Ants' population.

The output of the program includes:

i. The optimized delivery route with the total distance in kilometres.

The algorithm is coded using Jetbrains IntelliJ IDEA. JetBrains' IntelliJ IDEA is a Java integrated development environment. It is one of the best Java IDEs available. The Java IDE has an ergonomic design and focuses on developer efficiency. Groovy and Kotlin, for example, get support for JVM-based languages. IntelliJ IDEA allows programmers to develop Java desktop applications quickly and easily, along with mobile and web applications, and in addition to HyperText Markup Language (HTML), JavaScript, Cascading Style Sheets (CSS) ("Jetbrains IntelliJ Features", 2019). It also provides an avenue for NodeJs, ReactJs, AngularJs, and other developers to write clean code.

The algorithm is executed and tested on a Hewlett Packard (HP) EliteBook Folio 1040 G3 Ultrabook Computer with an Intel® Core™ i5 vPro CPU at 2.8GHz, running a 64-bit Windows 10 Operating System (OS). The test computer system is also equipped with an 8 GigaBytes (GB) Random Access Memory (RAM) and a 256 GigaBytes (GB) Sandisk Solid State Drive.

The proposed Ant Colony Optimization Algorithm (ACO) is tested on a randomly generated vehicle routing problem consisting of five (5) to eight (8) customers with their locations, a delivery vehicle, and randomly generated pheromone levels for each path. The program builds matrices for the pheromone levels, and distances of the paths involved in a complete Ant solution. The distances between any pair of locations are equal to the distance between them by road as obtained from the Google Maps API to make the program implementable to a real-life problem that involves routing of vehicles while taking into consideration the traffic volume.

The result is an optimal route through the customers for deliveries which can then be visualised on the map and followed in the order suggested by the algorithm to reduce time and cost of delivery. Since this experiment used the Ant Colony Optimization on a randomly generated set of locations and tried to compute the optimal route for delivery through the locations, its result and how well the optimization is done is always time varied.

Therefore, it is necessary to run the algorithm several times before averaging the results to have a better understanding of how the algorithm performs on a particular problem and how well the possible alternatives are improved before choosing the best option.

The table below shows the results of the experiments carried out on the vehicle routing problem involved in the delivery of posts to customers scattered around the city in various locations. A total of five (5) runs were performed on each case of the variation, and the result is averaged:

Table 4.1: Experimental Results

Number of Customers	**No of Vehicles**	**Average Distance (km)**
5	1	15.67
6	1	39.20
7	1	30.56
8	1	38.45

The experimental results analysis presented in the table above reveals that the algorithm performs better and provides more accurate results using Google Maps API integration to calculate distances to that of the Euclidean method but comes with a computational cost in term of time. This is because of the time spent fetching the distances data from the Google Maps API as opposed to calculating it statically.

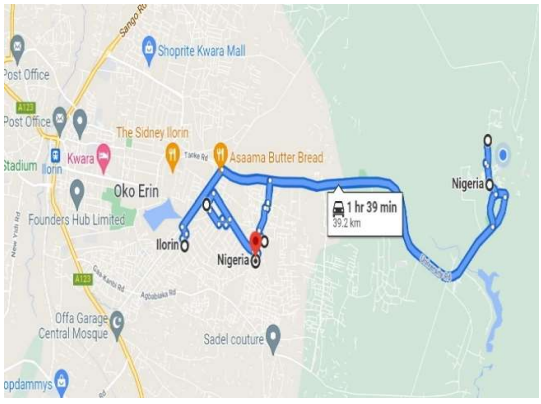

Figure 4.1: An Example of the ComputedRoute displayed using Google Maps API

CONCLUSION

The study produces an exemplary data structure that enforces dependency constraints among customers. The study also discovered a mathematical model for determining the best path that is simple to use. This would enable the adoption of a good data structure in representing a route (solution) constructed by ants from which the optimal route would be chosen. Therefore, the study was not only able to implement the Ant Colony Optimization Algorithm (ACO) in Java but also used SQLite for coding the database of the program and adopt the power of the implemented Ant Colony Optimization Algorithm to find the near-optimal solution for randomly generated Vehicle Routing Problem (VRP) in Postal Delivery System. The Ant Colony Optimization Algorithm approach in routing also provides a new, effective and efficient perspective to determine optimal routes to deliver posts. The result analysis of the study shows that it performs excellently in solving Vehicle Routing Problems.

REFERENCES

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). doi:10.1016/j.ejor.2012.11.040

Birattari, M., Pellegrini, P., & Dorigo, M. (2007). On the invariance of ant colony optimization. *IEEE Transactions on Evolutionary Computation, 11*(6). doi:10.1109/TEVC.2007.892762

Campbell, A. M., & Savelsbergh, M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science, 38*(3). doi:10.1287/trsc.1030.0046

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science, 6*(1). doi:10.1287/mnsc.6.1.80

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America, 2*(4). doi:10.1287/opre.2.4.393

Dorigo, M., & Caro, G. D. (1999). Ant colony optimization: A new meta-heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, 2*. doi:10.1109/CEC.1999.782657

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization.* MIT Press.

Dorigo, M., & Stützle, T. (2019). Ant colony optimization: Overview and recent advances. *International Series in Operations Research and Management Science, 272*. doi:10.1007/978-3-319-91086-4_10

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine, 1*(4). doi:10.1109/CI-M.2006.248054

Dorigo, M., Caro, G. D., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life, 5*(2). doi:10.1162/106454699568728

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 26*(1). doi:10.1109/3477.484436

El-Sherbeny, N. A. (2010). Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University - Science, 22*(3). doi:10.1016/j.jksus.2010.03.002

Fanti, M. P., Laraspata, R., Iacobellis, G., Mangini, A. M., Ukovich, W., & Abbatecola, L. (2014). A Decision Support System approach for the postal delivery operations. *IEEE International Conference on Automation Science and Engineering.2014-January.* Institute of Electrical and Electronic Engineering. doi:10.1109/CoASE.2014.6899386

Fischetti, M., Toth, P., & Vigo, D. (1994). Branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research, 42*(5). doi:10.1287/opre.42.5.846

Golden, B. L., Assad, A. A., & Wasil, E. A. (2002). 10. Routing Vehicles in the Real World: Applications in the Solid Waste, Beverage, Food, Dairy, and Newspaper Industries. In B. L. Golden, A. A. Assad, & E. A. Wasil, *10. Routing Vehicles in the Real World: Applications in the Solid Waste, Beverage, Food, Dairy, and Newspaper Industries.* doi:10.1137/1.9780898718515.ch10

Gong, Y. J., Zhang, J., Liu, O., Huang, R. Z., & Hung Chung, H. S. (2012). Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 42*(2). doi:10.1109/TSMCC.2011.2148712

Ji, P., & Chen, K. (2007). The vehicle routing problem: The case of the Hong Kong postal service. *Transportation Planning and Technology, 30*(2-3). doi:10.1080/03081060701390841

Kovacs, A. A., Golden, B. L., Hartl, R. F., & Parragh, S. N. (2014). Vehicle routing problems in which consistency considerations are important: A survey. *Networks, 64*(3). doi:10.1002/net.21565

Kuo, R. J., Zulvia, F. E., & Suryadi, K. (2012). Hybrid particle swarm optimization with genetic algorithm for solving capacitated vehicle routing problem with fuzzy demand - A case study on garbage collection system. *Applied Mathematics and Computation, 219*(5). doi:10.1016/j.amc.2012.08.092

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research, 59*(3). doi:10.1016/0377-2217(92)90192-C

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics, 54*(8). doi:10.1002/nav.20261

Laporte, G., Louveaux, F. V., & Hamme, L. V. (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research, 50*(3). doi:10.1287/opre.50.3.415.7751

Lau, H. C., Chan, T. M., Tsui, W. T., & Pang, W. K. (2010). Application of genetic algorithms to solve the multi depot vehicle routing problem. *IEEE Transactions on Automation Science and Engineering, 7*(2). doi:10.1109/TASE.2009.2019265

Liberatore, F., Righini, G., & Salani, M. (2011). A column generation algorithm for the vehicle routing problem with soft time windows. *4OR, 9*(1). doi:10.1007/s10288-010-0136-6

Liu, R., Jiang, Z., & Geng, N. (2014). A hybrid genetic algorithm for the multi-depot open vehicle routing problem. *OR Spectrum, 36*(2). doi:10.1007/s00291-012-0289-0

Lummus, R. R., Krumwiede, D. W., & Vokurka, R. J. (2001). The relationship of logistics to supply chain management: Developing a common industry definition. *Industrial Management and Data Systems, 101*(8). doi:10.1108/02635570110406730

Lysgaard, J., & Wøhlk, S. (2014). A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *European Journal of Operational Research, 236*(3). doi:10.1016/j.ejor.2013.08.032

Niroomand, I., Khataie, A. H., & Galankashi, M. R. (2014). Vehicle routing with time window for regional network services-Practical modelling approach. *IEEE International*

Conference on Industrial Engineering and Engineering Management, 2015-January. doi:10.1109/IEEM.2014.7058769

Prins, C., Labadi, N., & Reghioui, M. (2009). Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research, 47*(2). doi:10.1080/00207540802426599

Rivera, J. C., Afsar, H. M., & Prins, C. (2016). Mathematical formulations and exact algorithm for the multi-trip cumulative capacitated single-vehicle routing problem. *European Journal of Operational Research, 249*(1). doi:10.1016/j.ejor.2015.08.067

Song, Q., Wang, X., Li, X., & Zhang, C. (2009). Optimization of postal express mail network based on swarm intelligence. *Proceedings of the IEEE Conference on Decision and Control.* Institute of Electrical and Electronic Engineering. doi:10.1109/CDC.2009.5400879

Sun, X., & Wang, Y. (2011). An optimal real-time route choice model based on route traffic volume. *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering, TMEE 2011.* Transportation, Mechanical, and Electrical Engineering. doi:10.1109/TMEE.2011.6199423

Taniguchi, E., Thompson, R. G., & Yamada, T. (2004). Visions for city logistics. *Logistics Systems for Sustainable Cities*. doi:10.1108/9780080473222-001

Wang, Z., & Sheu, J. B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological, 122*. doi:10.1016/j.trb.2019.03.005