# A Routing Model for the Delivery of Perishable Products in Food Deserts

Jorge Nadjar

## Abstract

*Food deserts* are areas with limited access to healthy and affordable food, and they negatively affect low-income populations across the United States. A decreased access to transportation, low median household income, and a concentration of minority populations are just a few of the complex factors that generally characterize food deserts. This research proposes an algorithm that models the optimal route delivery vehicles should take to deliver food products to internally homogeneous clusters composed of households in designated food deserts. A one-to-one distributional model is used, where a single supplier serves a set of customers such that there is a single origin for delivery routes. Distributional and routing costs are considered, and we constrain for the perishability of the products being delivered during a discrete time period such that deliveries are done with products' shelf lives.

## 1 Introduction

A food desert is a geographical area in which the access to healthy and affordable food is limited. A defining aspect of food deserts is the lack of access to reliable transportation that would give a household the ability to obtain healthy food from supermarkets and grocery centers, which tend to be sparse in these areas to begin with. Food deserts are generally low-income and rural as well, thus exacerbating vehicular and transportation constraints.

The Vehicle Routing Problem (VRP) is a type of combinatorial problem that sets out to find an optimal route a fleet of vehicles can use to deliver products to different customers, and additional constraints are added to simulate realistic problems supply chain networks and companies have to deal with. Given the need for food deserts to have access to fresh and healthy food, food delivery systems for consumers are available solutions to fix many of the problems these places must confront. This paper considers a routing model for such food delivery systems, where a household or a cluster of households can order food products and have them delivered during a specific time frame related to the product's perishability.

Clustering involves creating subsets of households that are both internally homogeneous and externally different. Thus, the internal homogeneity is determined by income levels and access to vehicular transportation. Using clusters alleviates the amount of delivery locations needed to map while at the same time creating a less computationally intensive problem. This paper's solution to the VRP uses clusters of households as the delivery spots for food products, but the algorithm can operate under situations where food products are delivered to individual households as well.

## 2 Related Literature Review

### *2.1 Distribution Networks*

The distribution network VRPs deal with different types of distribution networks: *one-to-one*, *one-to-many*, and *many-to-many*. One-to-one involves a single supplier delivering to a customer, one-to-many involves a sing supplier delivering to a set of customers, and many-to-many involves a set of suppliers delivering to a set of customers. Perishability constraints are a newer addition to these types of problems. Soysal et al. [1,2] model inventory routing problems with perishable products and sustainability constraints. These studies also consider many other constraints, such as fuel, logistical, and emission costs. While these two studies consider a many-to-many distribution network and constrain heavily for fuel and emission costs, which differ from the main constraint involved in the research study, the inventory costs associated with these studies do factor in perishability concerns.

*2.2 Perishability Constraints*

Most algorithms that constrain for perishability consider product shelf lives that are fixed. Le et al. [3] measures perishability as a function of the number of time periods in which that good can still be consumed by a customer. Furthermore, a customer’s inventory, which in our study would be correspond to a regional cluster’s consumption bundle, i.e., the products this cluster demands, would thus have an upper-bound restriction relating to whether a product being delivered is within its designated shelf life. Additionally, the aforementioned study considers a one-to-many structure for a distribution network, and each transport vehicle conducts one route throughout a given time period. This latter constraint is common throughout the academic literature, such as in Soysal et al. [1].

*2.3 Inventory Routing Problems*

Inventory Routing Problems (IRP) differ from VRPs in the fact that IRPs constrain for the supplier’s inventory and the costs associated with it. Soysal et al. [1,2] and Le et al. [3] consider transportation costs from one node to another, inventory holding costs (i.e., products that become damaged or unsold), and vehicular capacities. Le et al. considers demand to be deterministic, while the two studies previously discussed [1,2] quantify demand to be uncertain. Stochastic or uncertain demand entails that a supplier does not know the future demand of a product, and deterministic demand [4], suppliers can predict consumer demand in the long run. The application of what type of demand is used changes depending on the type of distribution network. An IRP would generally assume that the demand between customers or a set of customers changes between time windows, but other problems [3] assume that the demand between a set of customers remains constant between deliveries.

*2.4 Genetic Algorithms in VRPs*

Vehicle Routing Problems have long used exact methods, but heuristic methods have become more common, given the intractability of VRPs. Heuristic methods are those that trade optimality for speed when it comes to finding a solution. Genetic algorithms are a heuristic method that has been applied to VRPs; genetic algorithms essentially mimic natural selection using chromosomes as a basic unit that is subject to evolutionary principles. Thus, it is able to iteratively produce better and better solutions as offspring with better genes and genetic variability are produced.

## 3 Problem Definition

Table 1. Variables in the Multi-objective Optimization Function

| **Variable** | **Meaning** |
|---|---|
| $H$ | Set of nodes; each node is a cluster composed of households and the origin depot |
| $H'$ | Set of nodes composed of only the cluster of households to have products delivered |
| $E$ | Set of weighted edges; $E = \{(i,j): i,j \in H, i \neq j\}$ |
| $c_{ij}$ | Cost of going from an arbitrary node *i* to node *j* |
| $h$ | Set of households inside a cluster $H_n$ |
| $P$ | Set of perishable goods |
| $s_p$ | Shelf life of perishable grocery product |
| $d_h$ | Demand of a set of households |
| $T$ | Discrete time period |
| $k$ | Transportation vehicle, $k \epsilon K$, where *K* is set of vehicles of a supplier |
| $Q$ | Vehicle capacity |
| $x_{ij}$ | Binary decision variable: $x_{ij} = 1 \iff$ is the arc going from *i* to *j* is part of the optimal route solution; otherwise, $x_{ij} = 0$ |

Modeling the logistics of supply chains is a complex optimization problem, and managing to incorporate both routing and inventory costs can exacerbate it. For this particular case, we will consider a one-to-many distribution network of consumer products. In supply chain and logistics, a **one-to-many distribution network** is one in which a single supplier serves a set of customers. This means that the distribution of products starts at a single origin and moves throughout multiple stops within a specified geographical area before returning to its point of origin.

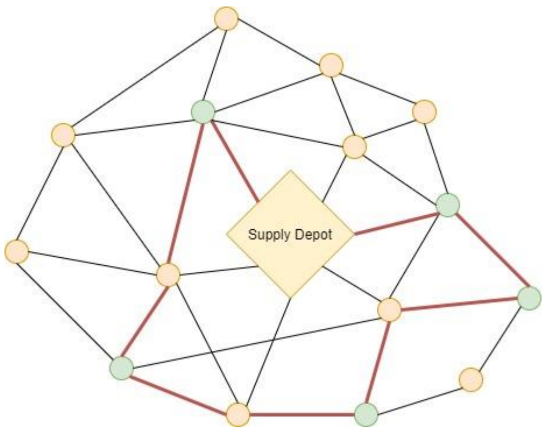

**Fig. 1**. An example of a one-to-many distribution network for a Vehicle Routing Problem. Green nodes represent customers to have products delivered and the red edges represent an optimal route for a vehicle.

Given this framework, we assume that all the vehicles used in transporting these goods are homogenous, meaning that they are identical to each other in terms of transportation capacity and distance mileage. We assume as well that the supplier can accommodate its production to satisfy the demand of the set of households in the food desert. The **planning horizon** of such deliveries is discrete, meaning that the deliveries done in the distribution network are done during a specified timeframe. Furthermore, we assume the vehicle fleet size of a given supplier to be of the same size and capacity.

We define our problem on a complete, undirected, weighted graph $G = \{H, E\}$, where $H$ is the set of nodes and node 0 and node $n + 1$ is the origin point, i.e., the supply depot from which deliveries start. Each node represents a cluster, each composed of a set of households $h = \{h_1, h_2, \dots, h_n\}$. $H'$ is the set of nodes that only contain the set of clusters of households to which products are delivered. Thus, we can express $H$ as $H = H' \cup \{0, n + 1\}$.

$E$ is the set of weighted edges for the graph representing the distance between clusters such that $E = \{(i, j): i, j \in H, i \neq j\}$. The supplier provides a set of products $P = \{p_1, p_2, \dots, p_m\}$, and each product has a shelf life that can vary between products, given that they consist of perishable food. Each route must start and end at the supply depot. $Q$ is the vehicular capacity of some vehicle $k \in K$, where $K$ is the set of vehicles of the supplier delivering these products. The value of $Q$ is the same for all of the vehicles, given the previous assumption of the homogeneity of vehicle capacities. Furthermore, the demand for a set of households, $d_h$, is deterministic, i.e., it is known by the supplier.

This particular problem involves minimizing the distance traveled by the suppliers and delivering fresh (non-spoilt) food, where $s_p$ is the average shelf life of a given set of products being delivered, all within a given time period $T$. This time period is constrained by the product's average shelf life. Thus, our main constraint is delivering consumer products in a discrete time window that is a function of the products' perishability.

We can represent a constrained optimization function using a vehicle flow formulation with time windows, adapted from Munari et al. [5]:

$$min \sum_{k \in K} \sum_{i \in H} \sum_{j \in H} x_{ij} c_{ij} \ \ subject\ to$$

$$\sum_{i \in H} x_{ij} = 1\,, \forall j \in H' \tag{2}$$

$$\sum_{j \in H} x_{ij} = 1\,, \forall i \in H' \tag{3}$$

Constraints (2) and (3) ensure that a cluster of households has a bundle of products delivered only once. $x_{ij}$ is a binary decision variable representing whether such edge is part of the optimal solution; as such, $x_{ij} \in \{0,1\}, \forall i, j \in H$.

$$\sum_{i \in H} x_{i0} = K \tag{4}$$

$$\sum_{i\in H} x_{0j} = K \qquad (5)$$

Constraints (4) and (5) ensure the fleet of delivery vehicles that exit and return to the supply depot remains the same in terms of its number of vehicles.

## 4 Proposed Algorithm

### *4.1 Introduction*

VRPs are intractable problems; as such, as the size of the input grows, the time to solve them grows exponentially. Multi-objective combinatorial optimization problems, such as this study's VRP, would work poorly using exact methods such as column generation [5]. Given that our problem is NP-hard, we propose a genetic algorithm based on those of the algorithms proposed by Baker et al. [6], Ombuki et al. [7], and Zhang et al. [8]. Genetic algorithms simulate evolutionary principles over a set of generations, ensuring that the strongest genes are passed down to subsequent generations.

### *4.2 Algorithm Pseudocode*

Table 2. Algorithm Variables and Data Structures

| **Variables** | **Meaning** |
|---|---|
| *CH* | A chromosome represented as a character string representing a permutation of all households $h \in H'$ that have perishable food products being delivered |
| $I_m$ | Assigned generation span |
| $\alpha$ | Weight parameter given transportation and distance costs |
| $\beta$ | Weight parameter given the perishability index of products |
| $P_n$ | A parent chromosome. We usually consider two parent chromosomes. |
| $C_n$ | A child chromosome |
| $P_m$ | Probability a mutation will occur |
| $P_f$ | Probability a child chromosome will not be discarded if its parents are a better fit |
| $CR$ | Data structure of chromosomes considered for an optimal route; we call this the *Genotype* |

**VRP-Algorithm**

```
1  $SetConstraints(Q, s_p)$
2  $CR \leftarrow GeneratePop(H')$
3  $PopSize = CR.length$
4  while i < $I_m$
5    $EvaluateFit(CR)$
6    $sort(CR)$
7    $CR.remove(CR.length - PopSize)$
8    for ($i \leftarrow 0; i < N; i{+}{+}$) do
9      $P_1, P_2 = ParentSelection(CH)$
10     $C_1 \leftarrow CrossOver(P_1, P_2)$
11     $C_2 \leftarrow CrossOver(P_2, P_1)$
12     $Mutate(C_1, P_m)$
13     $Mutate(C_2, P_m)$
14     if $C_1.fit > P_1$ or $C_1.fit > P_2$ then
15       $CR.add(C_1)$
16     if $C_2.fit > P_1$ or $C_2.fit > P_2$ then
17       $CR.add(C_2)$
18     if neither $CR.add(P_1, P_f)$ or $CR.add(P_2, P_f)$
19 return CR.optimalRouteSet
```

We represent each solution as a chromosome *CH* represented as a character string of the permutation of households that are having food products delivered for a given route. For example, if we have a route, we can represent the character string as follows, where the nodes in $h_v$ represent the nodes to be visited in that route: $h_{v_1} h_{v_2} \dots h_{v_n}$. Multiple routes can exist in a given problem as well.

Each gene in a chromosome thus represents an assigned node number to a customer, and the order of the genes is the sequence of nodes that a given vehicle would deliver to. Generating this sequence of nodes, which we denote as genes, must be done such that the established constraints are not violated. We parametrize based on the vehicular capacity $Q$ and the shelf life of the products being delivered $s_p$.

*CR* is the set of solutions, such that $CR = \{CH_1, CH_2, \dots, CH_n\}$. We base off this initial population on $H'$, the set of nodes that we plan to deliver to, excluding the supply depot. The initialization of the population is handled by the *GeneratePop* procedure. *CR* can be represented as an array; it is kept sorted in an increasing fashion based on overall fitness. Thus,

$CH_1$ would be an ideal solution given a set of solutions. We then evaluate the fitness of a chromosome based on the different constraints parametrized in the multi-objective function. We can represent a fitness function as follows, adapted from Ombuki et al. [7]:

$$Fitness = \alpha \cdot |K| + \beta \sum_{k \in K} \sum_{i \in H} \sum_{j \in H} x_{ij} c_{ij}$$

As previously mentioned, at each iteration, we sort the population of chromosomes based on their fitness. Each chromosome holds their fitness score, and this allows us to sort the set of solutions. After sorting in this fashion, we remove all those solutions that do not fit a minimum standard regarding their fitness. In this case, we remove those solutions that do not meet the average fitness given the overall set of chromosomes. Following this process, we use a tournament selection strategy to select two parents using the *ParentSelection* procedure, based on [7]. We randomly select a set of chromosomes which we call the tournament set. The procedure followed in this method is used to simulate the fact that the fittest individuals are sometimes not the ones that end up producing offspring.

Once we select two suitable parents from the two binary tournaments, we produce two offspring by mating the parent chromosomes. We do this through the *CrossOver* procedure. We adapt this crossover method it from Baker et al. [6], which uses a two-point crossover; two points are randomly selected from a chromosome. Regarding the first offspring, we choose those genes that are to the left of the first point and to the right of the second point. The second parent's inner gene values relative to those two points is then added to that first offspring. The reverse procedure is then replicated with the second offspring.

After generating the offspring, our next step is to add them to the overall set of feasible routes. However, to add genetic variability into the gene pool, we allow the offspring chromosomes the chance to mutate with the *Mutate* procedure. Mutations have a 10% chance of occurring for each of the offspring, and this probability is standard in various VRPs that use genetic algorithms. Once both *Mutate* procedures are applied on both offspring chromosomes, we then check the fitness of both offspring. If the first child chromosome is better fit than either of its parent chromosomes, then we add it to the set of feasible routes. We apply the same procedure to the second offspring chromosome. If neither are better than their parents in terms of overall fitness, then we add one of the parent chromosomes to the set of feasible solutions. However, given that we want to mimic evolutionary principles, we add the probability that a child chromosome will not be discarded if its parents are a better, and thus it will be added to the set of feasible solutions. Finally, we iterate doing the following procedures until we have reached a maximum number of generations and a maximum in time elapsed is reached as well. We then return the optimal feasible routes for each vehicle from the given set.

## *4.3 Case Analysis*

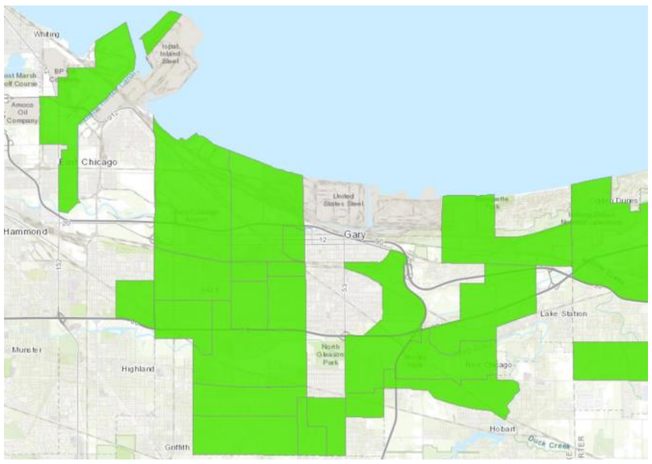

**Fig. 2**. Clusters of food deserts in the East Chicago area, shown in light green. Node locations have been arbitrarily selected within those clusters in this example.

To better illustrate the proposed algorithm, we will outline the different steps using a geographical example. Figure 2 shows the Gary/East Chicago area in the border of Indiana and Illinois, which contains numerous food deserts (measured by census tract). We represent ten arbitrary locations where food products are being delivered to. Two trucks are available for deliveries, each one having a capacity of 100 units of food products. Each cluster node we wish to deliver to has a different demand, and that demand is known by the supplier. The supply depot is represented as a node, but its demand is zero given that no deliveries are done to it.

We first generate a random genotype *CR* based on the ten locations we wish to deliver, adding the necessary constraints as well. These constraints come in the form of penalties to the fitness function. We can represent the nodes with a network topology as well to show the clustered delivery locations.

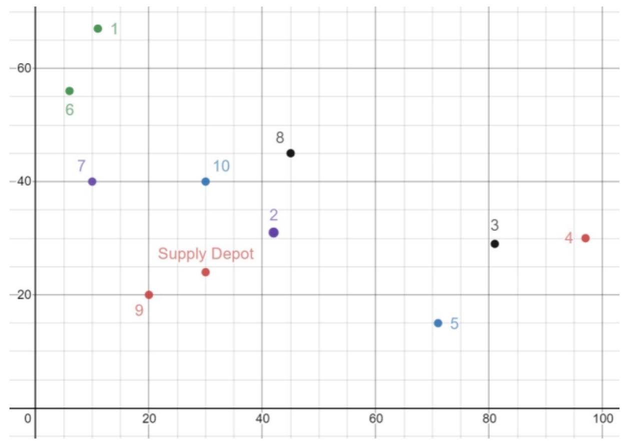

**Fig. 3**. Nodes for the delivery of products based in the East Chicago area using a Euclidean network topology.

The next step is to iterate through the genotype, stopping at the maximum number of generations, which we refer to as the generation span. In each iteration, we sort and find the chromosome with the greatest fitness. After selecting the parents and performing both the genetic and mutation operations, chromosomes are then essentially able to evolve. When we reach a set number of evolutions, we stop iterating and a feasible set of routes is produced as shown in Figure 4.

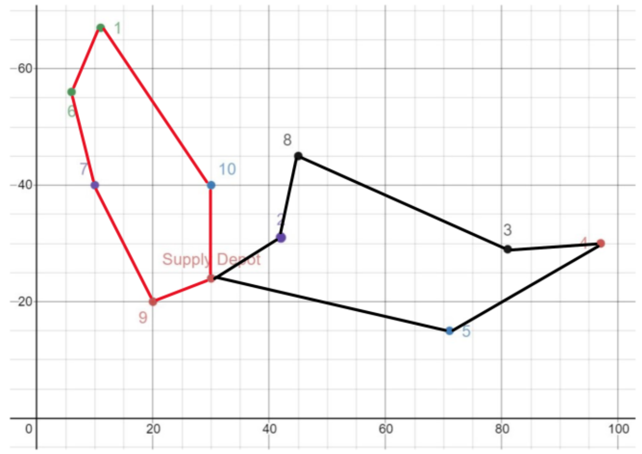

**Fig. 4**. Set of vehicular routes the two delivery trucks can utilize as derived from the proposed algorithm.

*4.4 Time Complexity Analysis*

Generating a population strand based on the households that have perishable food products being delivered requires $\Theta(H')$, which is the number of customers, or sets of households, having those products delivered. Evaluating the fitness of the set of chromosomes is dependent on the number of chromosomes in a given set of feasible network solutions. Given that each chromosome is a literal character string of size, we must iterate over the entire set of feasible solutions *CR*. If we let *N* be the length of *CR*¸ then the total time required to calculate the fitness of the set of chromosomes is $O(N)$. The same time complexity applies when we sort the set of chromosomes.

When selecting the parent chromosomes, the subset from where suitable parents are selected remains fixed. Thus, we approximate a time complexity of $\Theta(1)$. Calculating the time complexity of a mutation is dependent on the length of a chromosome *CH*. If we let *C* be the length of an arbitrary chromosome, then a mutation would have a time complexity of $O(C)$, given that we must iterate over the literal character string. Crossing over the genes from two parent chromosomes to create an offspring becomes $O(C)$ as well.

## 5 Experimental Results

### *5.1 Performance Metrics*

Throughout our testing of the performance of the proposed algorithm, we generally focus on time metrics, particularly those relating to the time taken for the set of generations to converge to a feasible and optimal solution. Given that we are using a genetic algorithm, the assessment of an optimal solution varies depending on the constraints and penalties associated with the fitness of each chromosome. Multiple parameters can be adjusted, such as the generation span, the number of evolutions allowed, and other genetic and mutation operators so that different optimal solutions are presented; as such, our performance metrics have a qualitative nature to them.

### *5.2 Datasets and Input*

Throughout our computations, we used Euclidean distances for finding the routing and transportation costs, and we set the population size at 300 and the generation span at 2000. We also set the crossover rate at 80% and the mutation rate at 10% throughout our test cases, as mentioned in previous sections. These values are generally standard for genetic algorithms relating to the Vehicle Routing Problem. Furthermore, we set a deterministic demand such that demand for food products for each location is known by the supplier before a delivery. To simulate realistic geographical and economic constraints, we set up network topologies that are sparse and non-uniformly distributed.

The input data is added to set Vehicle Routing Problem configuration with the help of *org.jgap* Java package, which allows for easier manipulation of chromosomes, fitness function design, mutation operators, and genetic operators. We vary the size of the input, which is the number of nodes we wish to deliver to. The vehicle fleet size also changes as we increase the number of spots we are delivering to; thus, we keep this number at a constant multiple. Thus, if we have 10 nodes to deliver, we keep the fleet size at 2 vehicles. At 20 nodes, we keep the fleet size at 4 vehicles, and so on. The combined demand of the set of clusters to deliver to must also increase as we increase the size of the input. These two aforementioned parameters must increase as the size of the input increases, since the vehicle fleet and the determined demand for such a region must be congruent with the number of nodes where products are being delivered to.

### *5.3 Results and Performance Analysis*

Throughout the test runs, multiple datasets were ran to obtain averages for each set number of nodes. In Figure 5, as previously mentioned, the fleet size was adjusted depending on the amount of delivery locations. Given the relatively small input sizes and the increasing fleet size per node dimension, results show a linearly increasing running time. These approximations are similar to the expected time complexity described in Section 4. However, although the datasets used in Figure 5 are realistic given supply chain logistics, given that suppliers can increase the size of the vehicle fleets to meet up with demand, several parameters are not kept constant.

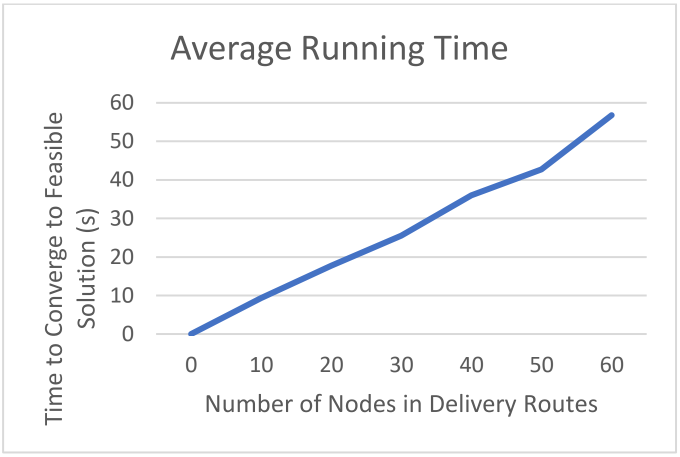

**Fig. 5**. Average Running Time while varying the number of nodes and the fleet size.

The results found throughout the experimentation show consistentency with our time complexity analysis. Adjusting for other parameters, such as keeping the vehicle fleet size constant as we increase the number of delivery locations should not change the time complexity of the proposed algorithm by a consequential factor. Iterating over the set of vehicles, when computing covered distance and demand, becomes a constant operation when conducting such analysis. Further testing done while adjusting for other parameters is essential to developing a more accurate time complexity of such algorithm.

## References

[1] M. Soysal, J. M. Bloemhof-Ruwaard, R. Haijema, J. G.A.J. van der Vorst, “Modeling an Inventory Routing Problem for perishable products with environmental considerations and demand uncertainty”, International Journal of Production Economics, Volume 164, 2015

[2] M. Soysal, J. M. Bloemhof-Ruwaard, R. Haijema, J. G.A.J. van der Vorst, “Modeling a green inventory routing problem for perishable products with horizontal collaboration”, Computer and Operations Research, Volume 89, 168-182, 2018

[3] T. Le, A. Diabat, J. Richard, Y. Yih, “A column generation-based heuristic algorithm for an inventory routing problem with perishable goods”, Optimization Letters, 7, 1481-1502, 2013

[4] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, A. Lokketangen, “Industrial aspects and literature survey: Combined inventory management and routing”, Computer and Operations Research, Vol 37, 1515-1536, 2010

[5] P. Munari, T. Dollevoet, R. Spliet, “A generalized formulation for vehicle routing problems”, Optimization and Control, Sept. 2016

[6] B. Baker, M.A. Ayechew, “A genetic algorithm for the vehicle routing problem”, Computer and Operations Research Vol. 30, 787-800, 2003

[7] B. Ombuki, B. Ross, F. Hanshar, “Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows”, Applied Intelligence 24, 17-30, 2006

[8] Y. Zhang, X.D. Chen, “An Optimization Model for the Vehicle Routing Problem in Multi-product Frozen Food Delivery”, Journal of Applied Research and Technology, Vol. 12, 239-250, April 2014