logistics

MDPI

Article

The Vehicle Routing Problem with Time Window and Randomness in Demands, Travel, and Unloading Times

Gilberto Pérez-Lechuga [1,*] and Francisco Venegas-Martínez [2]

[1] Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero, División de Estudios de Posgrado e Investigación, Av. 1° de Mayo y Sor Juana I. de la Cruz S/N Col. Los Mangos, Ciudad Madero 89440, Mexico
[2] Escuela Superior de Economía (ESE), Instituto Politécnico Nacional, Av. Plan de Agua Prieta 66, Plutarco Elías Calles, Miguel Hidalgo, Ciudad de México 11350, Mexico; fvenegas1111@yahoo.com.mx
* Correspondence: perezlechuga59@gmail.com; Tel.: +52-7712367599

Abstract

Background: The vehicle routing problem (VRP) is of great importance in the Industry 4.0 era because enabling technologies such as the internet of things (IoT), artificial intelligence (AI), big data, and geographic information systems (GISs) allows for real-time solutions to versions of the problem, adapting to changing conditions such as traffic or fluctuating demand. *Methods*: In this paper, we model and optimize a classic multi-link distribution network topology, including randomness in travel times, vehicle availability times, and product demands, using a hybrid approach of nested linear stochastic programming and Monte Carlo simulation under a time-window scheme. The proposed solution is compared with cutting-edge metaheuristics such as Ant Colony Optimization (ACO), Tabu Search (TS), and Simulated Annealing (SA). *Results*: The results suggest that the proposed method is computationally efficient and scalable to large models, although convergence and accuracy are strongly influenced by the probability distributions used. *Conclusions*: The developed proposal constitutes a viable alternative for solving real-world, large-scale modeling cases for transportation management in the supply chain.

Keywords: vehicle routing problem; stochastic modeling; Monte Carlo simulation; supply chain management; metaheuristics; logistics optimization

.

1. Introduction

In practical applications of supply chain management, especially in logistics engineering and transportation scheduling, the problem of vehicle routing is one of the central axes of large-scale transportation management. The routing model is an outgrowth of the classical traveling salesman problem (TSP). The TSP was proposed in the 19th century by mathematicians William Rowan Hamilton and Thomas Kirkman. Subsequently, George Dantzig, Ray Fulkerson, and Selmer Johnson were the first to publish a solution to the problem in 1954 [1].

The vehicle routing model consists of identifying the most efficient route for a set of customers with fixed locations. The problem has many variants, and almost all of them have received great attention in the specialized literature due to the economic impact that their instances provide. Despite the variety of archetypes, almost all of them contain the following common elements [2].

1. One or more central warehouses from which the products to be transported are sent or centralized.

Check for updates

Academic Editors: Mladen Krstić and Hao Yu

Received: 31 October 2025
Revised: 1 January 2026
Accepted: 5 January 2026
Published: 7 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

2. A predetermined number of vehicles required to carry out the distribution of the products whose carrying capacity is known.
3. A group of consumers who need to satisfy their periodic demand.
4. Knowledge of a matrix of distances between the points to be linked.
5. The time windows imposed on the delivery times of the products.
6. A known number of delivery transport operators.

There are also several variants of the model that depend on the context in which it is applied. Some of the best-known ones are mentioned in [3]. Some authors have made an effort to unify the various versions of the routing problem into a single representation. For example, in [4], a set of practical cases are shown, illustrating a variety of transportation tasks that can be modeled, ultimately returning to a simplified model. The classic vehicle routing model has also been analyzed from a machine learning perspective using a non-autoregressive supervised deep learning approach [5]. The authors address a highly specialized, frontier problem, namely the time-window model, using unconstrained quadratic binary optimization.

Another interesting approach to the problem is found in [6]. Here, the authors address a model that includes vehicle synchronization, a problem consisting of carrying out the coordination and consistency of data and information between the vehicle and other external systems or devices in order to optimize its overall efficiency. Also, in [7], the document presents an interesting variant of the vehicle routing problem, the Flexible Periodic Vehicle Routing Problem (FPVRP). In this variant, the carrier must establish a distribution plan to serve its customers over a known planning horizon. Each customer has a total demand that must be met within the horizon and a limit on the maximum quantity that can be delivered on each visit using a fleet of homogeneous vehicles with a given capacity. An attractive approach to the problem is found in [8]. In this paper, the authors develop an algorithm considering multiple depots (MDVRP) under capacity and path length constraints.

Although interesting from a theoretical perspective, these varieties are of little practical use because real-life logistics and transport management problems are a mixture of the above variants, with a large number of variables and constraints imposed on theoretical models. Here, stochastic approaches are the most useful, as they reveal the uncertainty existing in each case.

The most practical models contain a depot for various products, which are then distributed to secondary depots called spray depots (peripheral depots). From here, the products are generally sent to retail distributors who deliver them to the end customers. Products are moved from one or more central warehouses to the processing nodes via a primary route that connects the source nodes to the first link in the chain. The items then continue their journey from the processing depots to the final consumers via a secondary route.

Models gain importance and complexity when considering distribution centers, where a wide variety of operations are carried out, such as receiving, storage, order picking, packaging, and shipping. These operations are frequently performed using highly specialized software with the help of radio frequency technologies (RF technologies), such as WMS, or Warehouse Management Systems, which optimize warehouse management from goods receipt to shipping [9,10]. For a succinct classification of the model, see [11]. In this paper, the authors analyze the urban routing problem, highlighting important considerations for cost-effective smart delivery services with applications to smart cities. Similarly, for an in-depth review of the vehicle routing model with time windows, see [12].

An interesting proposal related to the warehouse reallocation problem to minimize the total workload needed to reallocate products to their new locations is found in [13]. Finally, an approximate model of our proposal solved via Tabu Search can be consulted in [14].

In turn, there are two types of configurations for distributing to customers from the spraying points. The first is shown in Figure 1. Here, the final transport departs and returns to the same starting point on both types of routes. In the primary route, or network 1, a high-capacity transport system is responsible for transporting products to peripheral nodes according to demand. The second option consists of a set of sub-routes where they circulate. The second option consists of a set of sub-routes where a set of smaller capacity vehicles will circulate, responsible for delivering the products to peripheral depots. Finally, from here, the product is sent to the end consumer or retailer.

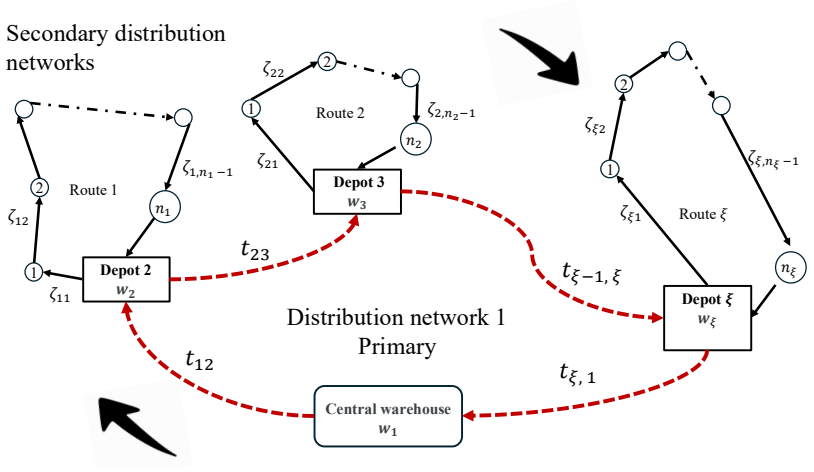


Figure 1. Network topology with a single centralized distributor warehouse and ξ spray points.

In the second case, the transport makes the round trip to each of the retail distributors, as shown in Figure 2. In this configuration, each transport visits a single peripheral node and returns to the central node, repeating the operation as often as necessary.

Another common way to allocate transportation is less analytical and involves creating an imaginary circle with a known radius, for example, 30 km around the central distribution warehouse, and setting a rate for delivery within the zone. The choice of the final delivery method is a company policy and is not always based on optimal rules.

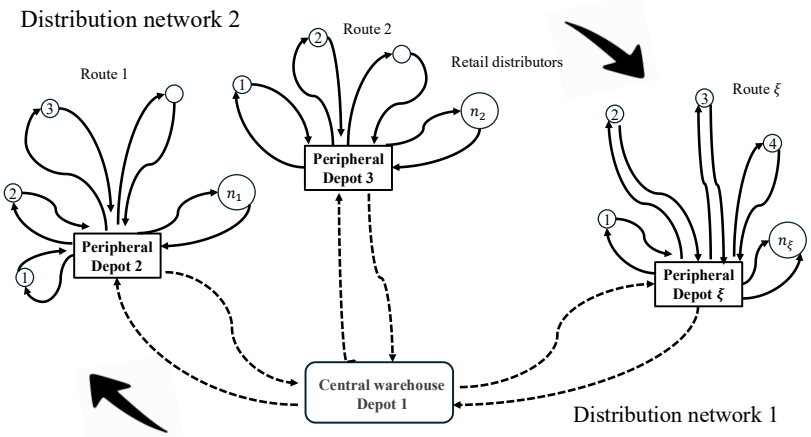


Figure 2. Sheet topology with a single centralized distributor warehouse and ξ spray points.

In our document, we develop a stochastic routing model that incorporates random demands, vehicles with different load capacities, and routes that must visit several depots

along the way. We also include time windows commonly used in perishable-product transportation models for cold-chain management. We exemplify our model with an optimized instance through a holistic approach inspired by two-phase mathematical programming but using the random search technique. The method uses random sampling of the search space to create random solutions within the defined limits of the problem, evaluating their effectiveness and retaining the best solution found so far. We include a full description of the solution algorithm.

In real-world logistics engineering models, the inherent randomness of demand, travel times, product unloading times, and more renders deterministic transportation planning and route selection models unreliable. These models fail to account for the variability of the results, leaving ample room for experimentation with the proposed solutions. This is without even considering the clustering problems inherent in vehicle routing models, which, by necessity, must be addressed from a random perspective.

However, simply including uncertainty in an optimization model is insufficient. The effect that random variables introduce on the model's topological properties (such as compactness, convexity, and more) is well-established, almost always undermining the scientific principles that guarantee finding an optimal value. For example, the Weierstrass theorem states that a continuous function on a closed and bounded interval, also known as a compact set, attains its absolute maximum and minimum values within that interval. Clearly, compactness in stochastic models is an attribute that cannot be guaranteed.

Therefore, we are interested in determining the effect of a probability density function in this context compared to another type of density. What is the effect of using an exponential distribution instead of a normal density to represent charging and discharging times? Does the spectrum of a density determine the overall convergence of the model? Based on the above, we have formulated the following research questions in our proposal.

Our research questions are formulated as follows:

1. Is it possible to approximate an average optimal solution to the stochastic routing problem with multiple random variables applicable to dual-routing topological configurations in large-scale models?
2. How do the density and instability of random variables affect network travel times?
3. Are the travel times of the main distribution network decisive in obtaining an approximately optimal solution?
4. How does product demand affect travel times and the number of vehicles required to satisfy it?
5. Is it possible to draw valid conclusions about a possible relationship between network travel times, required demands, and the time window imposed on the model?

It is important to emphasize that our proposal does not focus on solving a large-scale problem, i.e., one with thousands or millions of variables. Our approach is directed at studying the behavior of this type of models in configurations like the one show in Figure 1 and determining the influence of the random variables involved in finding an approximately optimal solution. Once the feasibility of an optimal solution has been evaluated under the proposed topology and working hypothesis, the model can be scaled with the assurance of finding optimal solutions.

2. Literature Review

Mathematical programming relates to the theory, use, and computational solution of mathematical models designed for decision-making, typically concerning the optimal use of scarce resources. Stochastic programming, as a by-product of generic mathematical programming, is a specialty that addresses decision-making problems where some parameters are uncertain and where uncertainty can be modeled through theoretical and/or empirical

probability distributions. Therefore, a stochastic routing problem (SVRP) is a particular case of stochastic optimization, which is an extension of the classic traveling salesman model.

The importance of the SVRP lies in its ability to reflect a more tangible reality than that traditionally studied in classical deterministic models. Due to its mathematical structure, the model allows for the inclusion of incomplete or even nonexistent information regarding supply chain management, especially transportation scheduling. For example, it is feasible to include travel time variability due to unpredictable traffic conditions, demand volatility, especially during peak seasons, unloading times, and more. Almost always, the lack of information or incomplete information is supplemented by using theoretical or empirical probability distributions that fit the sample data obtained in situ.

In real-world applications of large-scale models that handle millions of variables in logistics-related problems, the application of stochastic optimization techniques requires exhaustive variable sampling and the corresponding adjustment to probabilistic mathematical models, which significantly increases the complexity of the problem. In such situations, not only the travel times of the distribution networks matter, but also the number of vehicles used [15], the time windows, the operating costs of transport, the macro and micro strategic location of manufacturing facilities and more [16].

The presence of uncertainty forces us to approach the problem from the perspective of stochastic optimization, which allows us to incorporate potential future information into decision-making. Another advantage is that it allows us to recalibrate route selection as new information becomes available, allowing the system to react to unforeseen circumstances and optimize performance.

Traditional routing models typically aim to minimize total cost or distance traveled in the distribution network; however, there are situations where minimizing the travel time is more important. For example, in the distribution of perishable products in the cold chain, the delivery of short-lived medications, and the transport of human organs, semen, and blood. In other cases, there is interest in defining the optimal number of vehicle fleets for the delivery of goods at a minimum cost and time. Here, the problem is to determine the optimal quantity and type of vehicles needed to serve a set of customers or delivery points, minimizing costs and/or maximizing efficiency, which is why the objective is to minimize fixed costs (vehicle ownership and maintenance) and variable costs such as fuel, time, tolls, and more [17].

The stochastic routing problem considers the joint evolution of complex sequential routing decisions and stochastic information, and the solution alternatives constitute a major challenge for researchers in this field. An early approach to the problem can be found in [18]. The procedure given in this paper considers the demand at each delivery point to be a random variable and proposes that routes should be defined before the values of these are known. Another suggestive perspective is found in [19] , where the authors analyze the solution alternatives for the problem from the perspectives of conventional Markov decision processes (MDPs). An interesting point of view, similar to ours, is found in [20]. Here, the authors assume that the probability distribution of travel times is known and solve the problem of minimizing the sum of the probabilities of missing deadlines. The objective of the problem was to optimize a performance measure, called the delay index, which represents the risk of missing deadlines.

Another similar idea is found in [21]. This paper proposes an alternative way to model travel and service times. A routing engine is integrated into the model, and interesting results are reported in instances with different travel and service time distributions. Another perspective on the problem is to visualize it as a multi-objective stochastic vehicle routing model in a green environment. With this approach, ref. [22] presents and validates a new hybrid search algorithm to solve the proposed instances. Subsequently, a stochastic

multi-objective technique is used to solve the enigma, and as an important complement, an analysis of the effect of relaxing the customer time window is developed.

For a recent review of the approaches, authors, and techniques developed in this regard, see [23]. For a detailed analysis of the prospective research on this topic, see [24].

Other approaches to the stochastic routing problem include computational optimization algorithms inspired by the social behavior of animals such as flocks of birds or schools of fish. In [25] , the routing problem is addressed from the perspective of particle swarm optimization (PSO), proposing randomness in the demands with variations. The results obtained suggest that the technique is viable in small-scale models. An interesting approach to solving instances of stochastic routing models is found in [26]. This proposal considers transportation costs and vehicle travel speeds to be stochastic. The authors propose a hybrid solution procedure based on Ant Colony Optimization (ACO) and Discrete Event Simulation (DES). The costs generated in the process are considered through digital discrete event simulation.

The application of routing and allocation models in smart cities has also been addressed. Two models are presented in [27]. One is a vehicle routing (VRP) model for directing a fleet of generated waste to separation facilities in the smart city, and the other is for allocating resources from the separation facilities to recovery plants or landfills. Both models employ programming with random constraints via a stochastic optimization model combined with four metaheuristic algorithms to identify the best solution. The results of the study suggest that the technique used is valid for medium-sized models (300 variables).

Artificial intelligence and machine learning have also been involved in solving the stochastic routing model. In [28], reinforcement learning (RL) and machine learning (ML) techniques are used to solve the stochastic vehicle routing problem (SVRP). The stochastic perspective of the model uses an RL agent with a simple training architecture. The results are encouraging and appear to be more promising than traditional heuristics. This article presents a reinforcement learning approach to optimize the Stochastic Vehicle Routing Problem (SVRP) with time windows, focusing on reducing travel costs in goods delivery. For their solution, the authors use an attention-based neural network, trained through reinforcement learning, to minimize routing costs. A similar alternative is found in [29]. This article presents a reinforcement learning approach to optimize the Stochastic Vehicle Routing Problem(SVRP) with time windows, focusing on reducing travel costs in goods delivery. For their solution, the authors use an attention-based neural network, trained through reinforcement learning, to minimize routing costs.

An interesting alternative can be found in [30] . The paper addresses the Inventory Routing Problem (IRP) and the Vehicle Routing Problem (VRP), aiming to simultaneously optimize distribution, inventory, and vehicle routes. The problem with the maximum-level replenishment (ML) policy is modeled as a two-stage stochastic programming model to minimize total cost. In the first stage, inventory management and routing decisions are made, and in the second stage, the corresponding resource actions are implemented.

In more traditional approaches, the dynamic stochastic vehicle routing problem (DSVRP) is addressed in [31]. The problem is first approached as a two-stage stochastic programming model with resource constraints. In the first stage, the cost of the routing plan is minimized a priori, while in the second stage, the average of the highest moments for the resource cost, calculated using a set of scenarios, is minimized.

3. Materials and Methods

Stochastic linear programming (SLP) addresses the problem of optimizing linear models under conditions of uncertainty, i.e., when there is little or no information in some of their parameters. Typically, to cover this deficit, these parameters are represented by

theoretical or empirical probability distributions. The technique consists of making an initial decision that optimizes a utility function plus an expected value generated once certain events have occurred. Therefore, this approach is called two- or multi-stage optimization. In the first stage, the initial decision is made, and in subsequent stages, the succession of random events inherent to the model is allowed. Finally, the information obtained is incorporated and the initial decision is recalibrated in light of the results obtained [32]. The importance of these models is that they allow for the inclusion of more rational aspects found in engineering and science practice, where randomness is common in parameters such as demands, time, costs, and more. The archetype proposed in this document is the following.

3.1. Building the Routing Model

The problem is to find the most efficient routes to assign a fleet of vehicles with known capacity to distribute a given quantity of products among the system's clustered customers. The routing must satisfy time constraints defined by the products' shelf life. Our working hypotheses in building our model are as follows [33]:

1. Each deposit is visited only once in each cycle.
2. The consolidated demand D for each product in the distribution network is obtained by summing the demands δ_i from peripheral depots. Similarly, the consolidated demand for peripheral depots is obtained by summing the demand of the customers assigned to the depot cluster. In this proposal, customer demand is considered a random variable.
3. The requested products are transported in designated vehicles (with known capacity) without exceeding their capacity.
4. For product distribution, our model is based on Figure 1 and considers two types of routes. Route 1 (primary route) is the path followed by delivery trucks carrying the required quantity of products to the peripheral depots marked as Depot 2, Depot 3, ... , Depot ξ. At each depot, the delivery truck delivers the requested quantity of products and continues on its way until it is completely empty and returns to the central warehouse $W1$. The travel times T_{ij} from node i to node j of route 1 are random quantities with known probability density.
   The depots located along the network are intended to deconsolidate (split) orders placed on the routes designated as route 1, route 2, ... , and route ξ. The circles placed on the peripheral routes indicate the customers located along that path. We will refer to the set of routes mentioned as secondary routes. The same consideration applies to travel times $\mathcal{T}_{ij}$ from node i to node j on route 2.
5. There is a unique time window τ^* on the network such that

$$\tau^* = \min\{\tau_1, \ldots, \tau_\kappa\}$$

   where τ_r is the shelf life of the product r. There are κ products that require transportation. Note that τ^* uniquely defines the required time window without detriment to the value of κ. This means that we will take a single time window, which is completely defined by obtaining the shortest shelf line of all transported products, under the premise that the shelf life of freshness of the rest of the products is guaranteed.
6. The clusters that make up the peripheral depots are well defined and the assigned clients do not swap between clusters.
7. The distance from node i to node j is not necessarily equal to the distance from node j to node i.

8. Given the nature of our model, penalties for exceeding deadlines do not apply here to discourage late submissions.
9. The total distance traveled by all vehicles is linear with the costs generated by the trip.
10. Our model does not consider vehicle operating costs and wear and tear.
11. Operator salary costs are also not considered.
12. Our primary goal is to minimize the total travel time on both networks so that it is less than the time window imposed on the system and thereby minimize the magnitudes associated with travel time such as travel costs, time window violation penalties, and vehicle costs.

Table 1 shows the notation used in the construction of the mathematical model.

Table 1. Notation used in the model.

$G = (V, A)$	$\equiv$	Complete undirected graph.
$V_1 = \{v_0, v_1, \ldots, v_\xi\}$	$\equiv$	Set of nodes in the distribution network 1.
$V_2^\iota = \{v_0, v_1, \ldots, v_n\}$	$\equiv$	Set of nodes belonging to the secondary networks ι, $\iota = 1, 2, \ldots, \xi$
$A_1 = \{(i,j) \mid i,j \in V_1, i \neq j\}$	$\equiv$	Set of arches in the distribution network 1.
$A_2 = \{(i,j) \mid i,j \in V_2^\iota, i \neq j\}$	$\equiv$	Set of arches belonging to the secondary networks ι, $\iota = 1, 2, \ldots, \xi$
$x_{ij},\ y_{ij},\ i,j \in (V_1 \cup V_2^\iota)$	$\equiv$	Binary variables that decides whether the arche (i,j) is used. $(\cdot)_{ij} = 1$: yes it is used; $(\cdot)_{ij} = 0$ otherwise
$T_{ij} = t_{ij}, \mathcal{T}_{ij} = \zeta_{ij}\ \ i,j \in (V_1 \cup V_2)$	$\equiv$	Travel time matrices in networks 1 and 2, respectively (in hours).
$\Psi = \psi_i,\ i \in (V_1 \cup V_2^\iota)$	$\equiv$	Vector of download times (in hours) at node i.
τ^*	$\equiv$	Time window required by the system in hours.
$q_{ij}\ \ i,j \in (V_1 \cup V_2^\iota)$	$\equiv$	Quantity of product requested by the customer i in the depot j in pieces.
$D = \delta_{ij},\ i,j \in (V_1 \cup V_2^\iota)$	$\equiv$	Demand required by the client i and in the depot j in pieces.
$L_c^i,\ i \in (V_1 \cup V_2^\iota)$	$\equiv$	Load capacity of vehicle i in kg.
$K_i,\ i \in V_2^\iota$	$\equiv$	Number of customers served in cluster $\mathcal{C}^\iota \in V_2^\iota$.

Thus, taking Figure 1 as reference,

$$\Upsilon_1 = t_{12} + \sum_{j \in V_2^1}^{n_1} \zeta_{1j} + \sum_{i \in V_1} \psi_i^1 + \sum_{i \in V_2^1} \psi_i^1$$

$$\Upsilon_2 = t_{11} + t_{12} + \sum_{j=1}^{n_2 - 1} \zeta_{2j} + \sum_{i \in V_1} \psi_i^2 + \sum_{i \in V_2^2} \psi_i^2$$

in general

$$\Upsilon_\xi = \sum_{(i,j) \in V_1, j > i} t_{ij} + \sum_{(i,j) \in V_2^\iota j > i} \zeta_{ij} + \sum_{i \in V_1} \psi_i^\xi + \sum_{i \in V_2^\iota} \psi_i^n \tag{1}$$

where the random variable Υ^* defines the maximum network travel time in each run of the model, that is,

$$\Upsilon^* = \max\left[\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_\xi\right]. \tag{2}$$

Then for $\iota = 1, 2, \ldots, \xi$ and for each cluster assigned to the peripheral depot, we have the following conditional stochastic problem

$$\mathcal{Z}_2 = \min \mathbf{E}_{\Psi} \left[\sum_{i \in V_2^{\iota}} \sum_{j \in V_2^{\iota}} \zeta_{ij} \, x_{ij} \;\middle|\; \mathcal{Z}_1 \right] \tag{3}$$

subject to

$$\sum_{i \in V_2^{\iota}} x_{ij} = 1, \, \forall \; j \; \in \; V_2^{\iota} \setminus \{0\} \tag{4}$$

$$\sum_{j \in V_2^{\iota}} x_{ij} = 1, \, \forall \; i \; \in \; V_2^{\iota} \setminus \{0\} \tag{5}$$

$$\sum_{i \in V_2^{\iota}} x_{i,n+1} = 1, \tag{6}$$

$$\sum_{i \in V_2^{\iota}} x_{ij} - \sum_{j \in V_2^{\iota}} x_{ji} = 0, \; \forall \; i \; \in \; V_2^{\iota} \setminus \{0\} \tag{7}$$

$$\sum_{i \in V_2^{\iota}} x_{ij} \, (\psi_i + t_{ij}) \leq 0, \forall \; j \; \in \; V_2^{\iota} \setminus \{0\} \tag{8}$$

$$\sum_{i \in V_2^{\iota}} \delta_{ij} = D_p, \forall \; j \; \in \; V_2^{\iota} \setminus \{0\} \tag{9}$$

$$\sum_{i \in V_2^{\iota}} q_{ij} \geq D_p, \forall \; j \; \in \; V_2^{\iota} \setminus \{0\} \tag{10}$$

$$\sum_{i \in V_2^{\iota}}^{\xi} \omega_i \, q_{ij} \, x_{ij} \leq PL_c, \;\; c = 1, 2, \ldots, C, \;\; j \; \in V_2^{\iota} \setminus \{0\} \tag{11}$$

$$\tau^* \leq \Upsilon^* \tag{12}$$

$$x_{ij} \; \in \; \{0, 1\}, \, \delta_i \in D, \, t_{ij} \in T, \, \psi_i \in \Psi. \tag{13}$$

In our formulation, the objective function, Equation (3), minimizes the average time of transportation travel times by ensuring the shortest route to guarantee deliveries within the time window, Equation (12). Equations (4)–(6) guarantee that only one vehicle type is assigned to customer i. Constraint (7) represents the balance equation of the system. Inequality (8) states the allocation of download times when visiting node i.

Similarly, Equations (9) and (10) ensure the consolidation of demand in peripheral depots. Equation (11) determines the number of vehicles required to meet the required demand, where P is the maximum fraction of total capacity for each vehicle type [34].

Again, it is important to note that the constraint (12) represents the time window imposed on the system; that is, it is satisfied that within the final travel time plus the unloading time, the product can still be delivered without exceeding its shelf freshness. Although real-world models require travel times from node i to node j to be known, the reality is that they are never exact due to various factors such as traffic delays, detours, accidents, and more. Therefore, we impose the extra conditions $t_{ij} \neq t_{ji}$ and $\psi_{ij} \neq \psi_{ji}$. We have considered these times as random variables in an effort to make our proposal more realistic. The same situation applies to the service times δ_i^k for each client assigned to the system's depots.

Thus, each t_{ij}, ζ_{ij}, ψ_i and δ_{ij} is a lottery from the random variables T, $\mathcal{T}$, Ψ and D defined in its own probability space $(\Omega, \mathcal{F}, \mathcal{S})$.

From the above, we will assume that there exists a continuous function such that

$$F_{D_i}(\delta) = \int_0^{\delta} f_D(\vartheta)\, d\vartheta, \tag{14}$$

satisfying

$$\mathbf{E}[D_i] = \int_0^{\infty} \delta\, dF_{D_i}(\delta) < \infty, \text{ and } \mathbf{Var}[D_i] = \mathbf{E}[D_i]^2 - (\mathbf{E}[D_i])^2 < \infty \tag{15}$$

By the continuity hypothesis on D, $F_{D_i}(\delta)$ is a nondecreasing function, and therefore, there is an inverse function $\varphi = F_{D_i}^{-1}(u)$ defined for any value of u in $[0,1]$ such that

$$\varphi_i = F_{D_i}^{-1}(u) = \inf\{\delta \mid F_{D_i}(\delta) \geq u\},\ 0 \leq u \leq 1. \tag{16}$$

The same assumptions will be made about the random variables T, $\mathcal{T}$ and Ψ. The above implies that we can create lotteries or samples from the given random variables.

Regarding the objective function, Equation (3), the value of $\mathcal{Z}_1$ is given by

$$\mathcal{Z}_1 = Min\left\{\sum_{i \in V_1}\sum_{j \in V_1} t_{ij}\, y_{ij} \mid D_c\right\},\ i \neq j \tag{17}$$

and is subject to constraints (4) to (13) of the previous model by changing V_2^l to V_1, x_{ij} by y_{ij}, $D_c = \sum_i \delta_i$ represents the consolidated demand at depot $i \in V_1$.

3.2. Solution Strategy

The constitution of the model is very similar to a two-phase mathematical programming model with uncertainty ($TFMP$), so the solution strategy will be somewhat similar to the solution methodology for this type of problem. A $TFMP$ model can be written as

$$\text{Minimize}\left[c'x + \mathbf{E}_d\left[\min d'y\right] \mid Ax + By = b, x, y \geq 0\right] \tag{18}$$

where $By = b - Ax$, $y \geq 0$ which is equivalent to the deterministic program given by

$$\text{Minimize}\left[c'x + Q(x), x \geq 0\right] \tag{19}$$

where $Q(x) = \mathbf{E}[Q(x,b)]$. Here $Q(x,b) = \infty$ if the constraints are inconsistent, and $Q(x,b) = -\infty$ if the solutions of the constraints are unbounded below [35].

Like classic two-phase problems, the proposed model also requires decision-making in two distinct phases. The decision in the second stage depends on the decision in the first stage and the perceived uncertainty. In the first stage, a "here-and-now" decision must be made, and in the second stage, the model reacts to the uncertainty present in the first stage.

However, we do not opt for an optimization process based on this theory. Our proposal consists of generating, using a Monte Carlo Method, the demand values associated with peripheral depots, as well as the travel and unloading times of transport vehicles. Once this information is established, we proceed to optimize the routes of distribution network 2.

We will now introduce the concept of critical route time (CTT). This variable measures the actual critical time required in the network to deliver the product with the shortest freshness period. This magnitude is determined from the Y* value by subtracting the time of the last segment of the route for the cluster in question. This is because this time is no

longer significant for product delivery, and it is used for the vehicle assigned to the route to return to its base, usually empty.

$$CTT_i = \left[\sum_{(i,j) \in V_1, j > i} t_{ij} + \sum_{(i,j) \in V_2, j > i} \zeta_{ij} + \sum_{i \in V_1} \psi_i^{\xi} + \sum_{i \in V_2} \psi_i^{n} \right] - \sum_{i=1}^{\xi} t_{i,n_i} \tag{20}$$

This is the sum of the travel times on route 1 plus the travel times on route 2 minus the critical travel times on route 2.

4. Numerical Results

Below, we describe the sequence of steps used to solve the proposed instance.

4.1. Obtaining the First Feasible Solution

To illustrate our proposal, we consider a network topology as shown in Figure 1.

In our example, there is a single central warehouse (number 1) with $\xi = 4$ peripheral warehouses (typically, in cold chain models, the maximum number of peripheral warehouses to be serviced does not exceed this amount). Each peripheral warehouse has $K_i = 4$, $i = 1, \ldots, 4$ customers to serve. For the purpose of using the model, we employ $\tau^* = 50$ h. Similarly, we assume that $D \sim \mathcal{N}(\mu = 1000, \sigma = 150)$, $T \sim Exp(\lambda)$, $\mathcal{T} \sim Exp(\lambda)$ and $\Psi \sim Exp(\lambda)$.

In this instance, we assume that peripheral depots also have customers who come to make their purchases in the same way as the regular customers assigned to their cluster.

As sample estimators in the normal distribution, we use the sample mean and the corrected sample standard deviation using $(n-1)$ instead of n in the formula. In the same way, to approximate λ_{ij}, $i, j \in (V_1 \cup V_2)$ we use the biased estimator $\hat{\lambda}_{ij} \approx (\bar{t})^{-1}$. The capacity of the vehicles is $L_c^1 = 50{,}000$ kilos on network 1 and $L_c^2 = 10{,}000$ kilos on network 2.

Tables 2 and 3 show the parameters used in the central depot and the associated clusters, respectively. Here, $N_{ij} \equiv$ network i, cluster j, $\psi_{ij} \equiv$ download time on network i node j, $j = 1, \ldots, 5$.

Table 2. Travel times and demands parametric estimators in network 1.

N_1	$\hat{\lambda}_1$	$\hat{\lambda}_2$	$\hat{\lambda}_3$	$\hat{\lambda}_4$	$\hat{\lambda}_5$
$\hat{\lambda}_1$	0	18.00	35.00	17.00	15.00
$\hat{\lambda}_2$	17.12	0	21.00	28.00	11.00
$\hat{\lambda}_3$	34.85	20.00	0	17.00	13.00
$\hat{\lambda}_4$	18.00	27.10	16.33	0	15.00
$\hat{\lambda}_5$	14.68	11.30	12.61	14.33	0
$\hat{\psi}_{1j}$	1.00	1.20	1.60	1.20	0.90

The parameter values used in the demand $\delta_{ij} \equiv$ requested in the peripheral depots j were the following: $(\mu_2 = 2.8, \sigma_2 = 1.5)$; $(\mu_3 = 3, \sigma_3 = 2)$; $(\mu_4 = 3.5, \sigma_4 = 1.8)$; $(\mu_5 = 4, \sigma_5 = 1.7)$. By the hypothesis of normality, we have that $\mu_1 = 2.8 + 3 + 3.5 + 4 = 13.3$ and $\sigma_1^2 = 1.5^2 + 2^2 + 1.8^2 + 1.7^2 = 12.38$.

Table 3. Travel times and demands parametric estimators in network 2.

N_{21}	λ_1	λ_2	λ_3	λ_4	λ_5	N_{22}	λ_1	λ_2	λ_3	λ_4	λ_5
λ_1	0.00	9.00	6.00	2.00	4.00	λ_1	0.00	7.00	4.00	3.00	3.00
λ_2	8.13	0.00	7.00	5.00	2.00	λ_2	7.04	0.00	8.00	3.00	5.00
λ_3	6.18	7.50	0.00	2.00	1.00	λ_3	4.14	8.80	0.00	3.00	2.00
λ_4	1.81	5.70	2.95	0.00	2.00	λ_4	2.56	2.84	3.81	0.00	1.00
λ_5	3.69	1.21	1.23	2.00	0.00	λ_5	2.62	4.09	2.54	0.35	0.00
ψ_{2j}	0.90	0.71	1.00	1.20	0.90	ψ_{2j}	0.20	0.70	0.10	0.90	1.20
N_{23}	λ_1	λ_2	λ_3	λ_4	λ_5	N_{24}	λ_1	λ_2	λ_3	λ_4	λ_5
λ_1	0.00	6.00	2.00	7.00	4.00	λ_1	0.00	2.00	5.00	9.00	8.00
λ_2	5.31	0.00	7.00	2.00	3.00	λ_2	2.53	0.00	3.00	5.00	4.00
λ_3	1.21	6.72	0.00	6.00	1.00	λ_3	4.71	2.30	0.00	6.00	8.00
λ_4	6.50	2.90	5.07	0.00	4.00	λ_4	9.08	5.23	6.35	0.00	5.00
λ_5	4.62	2.01	0.23	3.76	0.00	λ_5	8.33	4.99	7.03	4.76	0.00
ψ_{2j}	0.80	1.21	0.60	0.40	0.40	ψ_{2j}	1.00	1.30	1.20	0.90	0.70

The parameter values shown were used to construct the inverse function given by Equation (16) as follows to obtain the travel times and unloading times, respectively.

$$\varphi_i = (\lambda)^{-1} \ln(U),\ U \in (0,1) \tag{21}$$

Similarly, the construction of the respective demands was carried out using Algorithm 1 shown below.

Algorithm 1 Generating samples from $Y \sim (\mu, \sigma^2)$

Input: μ, σ, U_1, U_2 from $\mathcal{U} \sim (0,1)$

$X \leftarrow \sqrt{\frac{\pi}{8}} \ln\left[\frac{1+U_1}{1-U_1}\right]$

if $U_2 \leq 0.5$ **then** $Z = -X$

$Z \leftarrow X$

$Y \leftarrow \mu + \sigma Z$

end if

Output: Y

Where μ and σ are the corresponding mean and standard deviation of a normal distribution with density

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}},\ x \in \mathbb{R} \tag{22}$$

U_1 and U_2 are the random variables uniformly distributed on the closed interval $[0,1]$ with density function

$$h_X(x) = 1,\ x \in [0,1]. \tag{23}$$

4.2. Moving Toward an Approximate Convergent Solution

The next step in the process is to create the respective lotteries from the already defined parameter values and, based on Algorithm 2, move toward an improved solution. The results are as follows.

Here, μ_{ij} and σ_{ij} are the mean and standard deviation values corresponding to a normally distributed random variable, Equation (22). Furthermore, ψ_i and λ_{ij} are the

discharge times of the paths and the intensity of the system that measures the inter-arrival times, respectively, with an exponential density given by

$$g_X(x) = \lambda\, e^{-\lambda x}\ x \geq 0. \tag{24}$$

The process used for instance optimization is shown in Algorithm 2.

Algorithm 2 Convergence algorithm

$s \leftarrow 0$ ▷ Obtaining an initial feasible solution
$\mathcal{Z}_1^s \leftarrow 0$
$\mathcal{Z}_2^s \leftarrow 0$
Initial step ▷ Parameters for V_1
$D_i \leftarrow \mu_i,\ \sigma_i,\ \forall\ i \in V_1$
$T_{ij} \leftarrow \lambda_{ij}, \forall\ (i,j) \in V_1$
$\Upsilon_\xi^1 \leftarrow \sum_{j \in V_1} T_{ij} + \sum_{i \in V_1} \psi_i^k,\ k = 1, \ldots$
$\mathcal{Z}_1^s$ is computed based on Υ_ξ^1 ▷ Parameters for V_2
$\delta_i \leftarrow \mu_i,\ \sigma_i,\ \forall\, i \in V_2$
$\mathcal{T}_{ij} \leftarrow \lambda_{ij}, \forall\ (i,j) \in V_2$
$\Upsilon_\xi \leftarrow \sum_{j \in V_2} T_{ij} + \sum_{j \in V_2} t_{ij} + \sum_{i \in V_2} \psi_i^k,\ k = 1, \ldots$
$\mathcal{Z}_2^{s+1} \leftarrow \{CTT_i\}\ \forall\, i \in V_2$ ▷ $\mathcal{Z}_2^{s+1}$ is computed based on Υ_ξ
if $\mathcal{Z}_2^{s+1}$ is unfeasible $(\mathcal{S} = \emptyset)$ **then** ▷ Return to the initial step
  get new parameter values
else if $\mathcal{Z}_2^{s+1}$ is feasible $(\mathcal{S} \neq \emptyset)$ **then**
  $X_s \leftarrow (D_s, T_s, \Psi_s, \mathcal{Z}_2^{s+1}) \mid X_s \in \mathcal{S}$
  while $\mathcal{Z}_2^{s+1} - \mathcal{Z}_2^s \leq \varrho$ **do**
    $s \leftarrow s + 1$ ▷ Convergence to the optimum ▷ Calculate values for the next iteration $s+1$
    ... ▷ Include necessary computations here for next $\mathcal{Z}_2^{s+1}$ and $\mathcal{Z}_2^s$
  end while
end if

Table 4 shows the lotteries obtained as a first instance to create the travel times from the central warehouse to the peripheral warehouses in the network. Similarly, Table 5 shows the same procedure but applied to the travel times between the nodes belonging to the clusters created in each peripheral warehouse by using Equation (21) and Algorithm 1. The rows at the end of each table show the draws associated with the demands required in the instance, where $D = \sum_{j \in V_1} \delta_i$.

Table 4. Lottery $\mathcal{T}$ matrices of demand and travel times at peripheral depots.

w_1	n_1	n_2	n_3	n_4	n_5
n_1	0.00	27.32	75.39	31.50	7.39
n_2	3.45	0.00	24.14	34.14	1.92
n_3	31.21	16.69	0.00	45.09	3.06
n_4	5.34	12.63	9.29	0.00	4.03
n_5	6.66	25.28	8.04	3.93	0.00
D	15.01	17.51	12.06	21.24	20.07

Table 5 shows the lotteries associated with the demand matrices in networks one and two, respectively.

Table 5. $\mathcal{T}$ travel time lottery matrices and demands at peripheral depots.

w_2	n_1	n_2	n_3	n_4	n_5	w_3	n_1	n_2	n_3	n_4	n_5
n_1	0.00	7.79	6.20	2.34	3.21	n_1	0.00	2.21	2.71	8.38	1.11
n_2	4.19	0.00	20.04	15.13	2.37	n_2	5.27	0.00	6.88	0.20	2.92
n_3	0.91	2.18	0.00	2.20	1.54	n_3	6.14	12.31	0.00	2.83	1.48
n_4	1.59	4.41	3.37	0.00	2.54	n_4	6.24	2.65	10.01	0.00	0.37
n_5	2.37	1.62	1.16	1.50	0.00	n_5	1.03	18.46	1.40	1.73	0.00
δ_j	5.21	4.36	2.52	0.51	2.41	δ_j	0.66	6.12	2.91	3.42	4.39
w_4	n_1	n_2	n_3	n_4	n_5	w_5	n_1	n_2	n_3	n_4	n_5
n_1	0.00	4.70	8.58	6.02	2.24	n_1	0.00	3.19	2.61	0.59	10.04
n_2	1.51	0.00	4.42	5.43	7.54	n_2	3.44	0.00	6.19	15.15	2.50
n_3	0.29	1.18	0.00	10.31	2.61	n_3	5.29	4.91	0.00	4.18	37.34
n_4	5.79	0.48	2.36	0.00	12.94	n_4	14.18	7.30	5.64	0.00	2.62
n_5	4.34	0.38	1.81	8.08	0.00	n_5	8.62	3.09	13.14	2.49	0.00
δ_j	2.06	2.27	3.72	0.50	3.51	δ_j	4.79	5.20	3.90	0.90	6.44

Table 6 shows the results obtained after applying Algorithm 2 to the information provided. Columns 1 to 5 show the demands considered in this iteration. Column 6 shows the cumulative demand for the analyzed reservoir, and finally, columns 7 and 8 provide the optimal solution in each case. The total number of transports required in this instance is as follows.

1. Route 1 (w_1): Two transports with a load capacity of 50,000 kilos each.
2. Route 2 (w_2): Two transports with a load capacity of 10,000 kilos.
3. Route 3 (w_3): Two transports with a load capacity of 10,000 kilos each.
4. Route 4 (w_4): Two transports with a load capacity of 10,000 kilos.
5. Route 5 (w_5): Two transports with a load capacity of 10,000 kilos each.

Table 6. Results of the first iteration.

Central depot demands (w_1)						Solution	
D_{w_1}	$D_{w_{12}}$	$D_{w_{13}}$	$D_{w_{14}}$	$D_{w_{15}}$	$\sum D_i$	$w_1 \xrightarrow{22.14} w_{12}$	$\mathcal{Z}_1 = 75.86$
0.00	14.32	17.24	18.77	15.54	65.88	$\xrightarrow{8.35} w_{14} \xrightarrow{5.69} w_{15} \xrightarrow{12.53} w_1$	$L_c = 50$
Required transportation: 2						$w_1 \xrightarrow{10.18} w_{13} \xrightarrow{16.97} w_1$	
First peripheral warehouse demands (w_{12})						Solution	
w_{12}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{12}$	$w_{12} \xrightarrow{4.85} 2 \xrightarrow{1.78} 5 \xrightarrow{0.37} w_{12}$	$\mathcal{Z}_1 = 16.25$
1.27	4.20	2.49	2.95	3.41	14.32	$w_{12} \xrightarrow{0.28} 3 \xrightarrow{4.57} 4 \xrightarrow{4.40} w_{12}$	$L_c = 10$
Required transportation: 1							
Second peripheral warehouse demands (w_{13})						Solution	
w_{13}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{13}$	$w_{13} \xrightarrow{4.73} 3 \xrightarrow{1.22} 4 \xrightarrow{4.38} 2$	$\mathcal{Z}_1 = 13.71$
6.52	3.54	2.27	1.84	3.07	17.24	$\xrightarrow{0.85} w_{13}$	$L_c = 10$
Required transportation: 2						$w_{13} \xrightarrow{1.91} 5 \xrightarrow{0.62} w_{13}$	
Third peripheral warehouse demands (w_{14})						Solution	
w_{14}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{14}$	$w_{14} \xrightarrow{1.66} 2 \xrightarrow{1.96} 4 \xrightarrow{5.02} w_{14}$	$\mathcal{Z}_1 = 20.36$
3.49	3.61	1.93	4.37	5.38	18.77	$w_{14} \xrightarrow{7.41} 5 \xrightarrow{1.68} 3 \xrightarrow{2.63} w_{14}$	$L_c = 10$
Required transportation: 2							
Fourth peripheral warehouse demands (w_{15})						Solution	
w_{15}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{15}$	$w_{15} \xrightarrow{1.39} 2 \xrightarrow{1.47} 5$	$\mathcal{Z}_1 = 19.80$
5.20	1.44	−1.41	5.99	4.32	15.54	$\xrightarrow{1.49} 3 \xrightarrow{4.59} w_{15}$	$L_c = 10$
Required transportation: 2						$w_{15} \xrightarrow{7.50} 4 \xrightarrow{3.36} w_{15}$	
$\mathcal{Z}_2 \approx \max\{28.77, 26.51, 39.16, 39.58, 40.53, 47.04\} = 47.04 < \tau^*$.							

Similarly, Tables 7 and 8 show the results of the second and third iterations. For reasons of space, from here on, the results of the lotteries associated with each iteration are no longer shown. Likewise, we only show the first two recurrences, as well as the best response found in the process.

Table 7. Results of the second iteration.

Central depot demands (w_1)						Solution	
D_{w_1}	$D_{w_{12}}$	$D_{w_{13}}$	$D_{w_{14}}$	$D_{w_{15}}$	$\sum D_i$	$w_1 \xrightarrow{10.96} w_{12}$	$\mathcal{Z}_1 = 112.39$
0.00	21.50	23.44	25.81	26.70	97.46	$\xrightarrow{21.12} w_{15} \xrightarrow{2.27} w_1$	$L_c = 50$
Required transportation: 2						$w_1 \xrightarrow{24.29} w_{13}$	
						$w_{14} \xrightarrow{1.27} w_1$	
First peripheral warehouse demands (w_{12})						Solution	
w_{12}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{12}$	$w_{12} \xrightarrow{0.40} 2 \xrightarrow{3.49} 3 \xrightarrow{5.06} w_{12}$	$\mathcal{Z}_1 = 29.70$
3.69	5.70	0.60	4.59	6.93	21.50	$w_{12} \xrightarrow{3.22} 4 \xrightarrow{11.03} w_{12}$	$L_c = 10$
Required transportation: 3						$w_{12} \xrightarrow{3.22} 5 \xrightarrow{3.22} w_{12}$	
Second peripheral warehouse demands (w_{13})						Solution	
w_{13}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{13}$	$w_{13} \xrightarrow{3.09} 2 \xrightarrow{0.61} 4 \xrightarrow{5.53} w_{13}$	$\mathcal{Z}_1 = 14.30$
5.85	3.22	6.80	4.64	2.93	23.44	$w_{13} \xrightarrow{1.24} 5 \xrightarrow{2.62} 3 \xrightarrow{1.21} w_{13}$	$L_c = 10$
Required transportation: 2							
Third peripheral warehouse demands (w_{14})						Solution	
w_{14}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{14}$	$w_{14} \xrightarrow{5.42} 2 \xrightarrow{4.77} 3 \xrightarrow{3.09} w_{14}$	$\mathcal{Z}_1 = 28.05$
6.53	4.82	5.07	5.29	4.11	25.81	$w_{14} \xrightarrow{1.05} 4 \xrightarrow{7.89} w_{14}$	$L_c = 10$
Required transportation: 3						$w_{14} \xrightarrow{0.37} 5 \xrightarrow{2.55} w_{14}$	
Fourth peripheral warehouse demands (w_{15})						Solution	
w_{15}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{15}$	$w_{15} \xrightarrow{0.46} 3 \xrightarrow{2.03} 2 \xrightarrow{1.39} 1$	$\mathcal{Z}_1 = 19.80$
4.03	3.02	4.46	6.97	8.23	26.70	$w_{15} \xrightarrow{1.49} 4 \xrightarrow{4.59} w_{15}$	$L_c = 10$
Required transportation: 2						$w_{15} \xrightarrow{7.50} 5 \xrightarrow{3.36} w_{15}$	
$\mathcal{Z}_2 \approx \max\{28.77, 26.99, 39.58, 34.11, 43.68, 40.06, 20.51, 12.09\} = 43.68 < \tau^*$.							

The results of the third iteration are shown in Figure 3, and a more detailed explanation of the meaning of the solution is provided below. At the center of the graph, the central depot is shown with a consolidated demand in kilograms (in thousands) resulting from the sum of the demands from the four peripheral depots. The green numbers represent the demands of the customers assigned to each cluster. Likewise, the consolidated demand for each cluster is observed in the system's peripheral warehouses. The black numbers below the arcs represent the travel times from node i to node j. Therefore, this solution is feasible. The graphical solution is shown in Figure 3

The interpretation of the results in Table 8 is as follows. Columns 1 to 5 show the demands considered in this iteration. Column 6 shows the cumulative demand for the analyzed reservoir, and finally, columns 7 and 8 provide the optimal solution in each case. The total number of transports required in this instance is as follows.

1. Route 1 (w_1): Two transports with a load capacity of 50,000 kilos each.
2. Route 2 (w_{12}): One transport with a load capacity of 10,000 kilos.
3. Route 3 (w_{13}): Two transports with a load capacity of 10,000 kilos each.
4. Route 4 (w_{14}): Two transports with a load capacity of 10,000 kilos each.
5. Route 5 (w_{15}): Two transports with a load capacity of 10,000 kilos each.

Table 8. Results of the third iteration (optimal).

Central depot demands (w_1)						Solution	
D_{w_1}	$D_{w_{12}}$	$D_{w_{13}}$	$D_{w_{14}}$	$D_{w_{15}}$	$\sum D_i$	$w_1 \xrightarrow{21.28} w_{15} \xrightarrow{4.16} w_{13}$	$\mathcal{Z}_1 = 97.74$
0.00	11.33	15.97	15.06	14.60	56.96	$\xrightarrow{9.93} w_{14} \xrightarrow{26.88} w_1$	$L_c = 50$
Required transportation: 2						$w_1 \xrightarrow{21.28} w_{12} \xrightarrow{14.01} w_1$	
First peripheral warehouse demands (w_{12})						Solution	
w_{12}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{12}$	$w_{12} \xrightarrow{2.29} 2 \xrightarrow{6.07} 5 \xrightarrow{1.29} 4$	$\mathcal{Z}_1 = 11.36$
2.40	4.19	2.53	1.43	2.45	11.33	$\xrightarrow{1.42} 3 \xrightarrow{0.29} w_{12}$	$L_c = 10$
Required transportation: 1							
Second peripheral warehouse demands (w_{13})						Solution	
w_{13}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{13}$	$w_{13} \xrightarrow{1.46} 2 \xrightarrow{1.39} 3 \xrightarrow{5.01} w_{13}$	$\mathcal{Z}_1 = 13.09$
5.44	1.71	3.55	1.91	3.38	15.97	$w_{13} \xrightarrow{3.67} 5 \xrightarrow{0.96} 4 \xrightarrow{0.60} w_{13}$	$L_c = 10$
Required transportation: 2							
Third peripheral warehouse demands (w_{14})						Solution	
w_{14}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{14}$	$w_{14} \xrightarrow{2.59} 3 \xrightarrow{2.37} w_{14}$	$\mathcal{Z}_1 = 13.24$
3.26	0.82	4.27	1.47	6.88	15.06	$w_{14} \xrightarrow{5.26} 4 \xrightarrow{1.08} 5$	$L_c = 10$
Required transportation: 2						$\xrightarrow{0.51} 2 \xrightarrow{1.43} w_{14}$	
Fourth peripheral warehouse demands (w_{15})						Solution	
w_{15}	δ_2	δ_3	δ_4	δ_5	$\sum \delta_i + w_{15}$	$w_{15} \xrightarrow{0.59} 2 \xrightarrow{1.01} 5$	$\mathcal{Z}_1 = 14.41$
4.12	1.11	1.15	3.82	4.40	14.60	$\xrightarrow{1.14} 3 \xrightarrow{2.16} w_{15}$	$L_c = 10$
Required transportation: 2						$w_{15} \xrightarrow{7.51} 4 \xrightarrow{2.00} w_{15}$	
$\mathcal{Z}_2 \approx \max\{24.42, 28.99, 28.49, 30.27, 38.16, 42.45, 32.35\} = 42.45 < \tau^*$.							

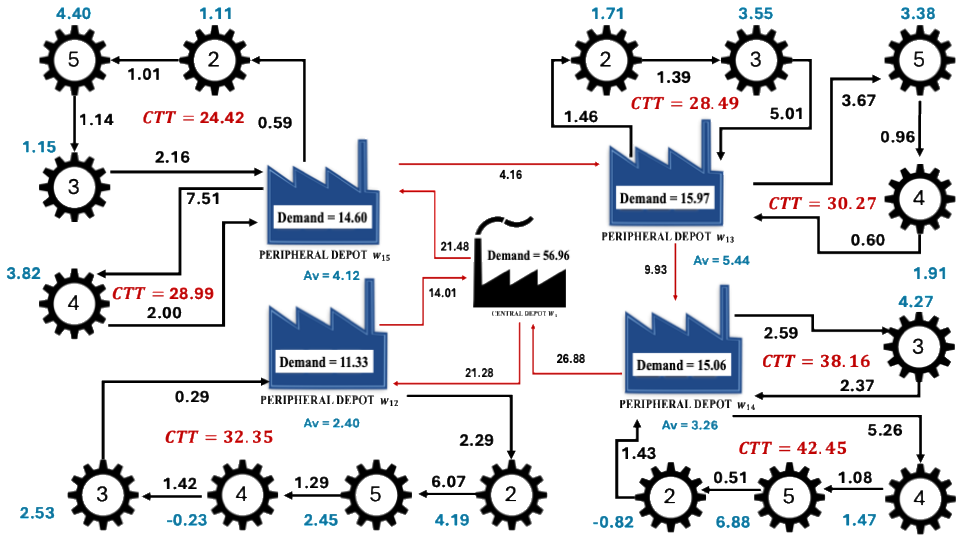


Figure 3. Optimal topology last iteration.

The optimal results are displayed in Figure 3 and are interpreted as follows:

1. Five components in the system: One central store (W_1) and four peripheral stores ($W_{12}, W_{13}, W_{14}, W_{15}$).
2. Demands registered by warehouse (in tons): $\delta_{12} = 11.33, \delta_{13} = 15.97, \delta_{14} = 15.06$, $\delta_{15} = 14.60$. Total consolidated demand = 56.96.
3. Route along the primary route:
   (a) First tour: $w_1 \xrightarrow{21.48} w_{15} \xrightarrow{4.16} w_{13} \xrightarrow{9.93} w_{14} \xrightarrow{26.88} w_1$

(b) Second tour $w_1 \xrightarrow{21.28} w_{12} \xrightarrow{14.01} w_1$

(c) Total travel time on the primary route: 62.45 + 35.29 = 97.74

4. Path followed on the first secondary route:

(a) Peripheral Deposit w_{15}:

i. First tour: $w_{15} \xrightarrow{0.59} w_2 \xrightarrow{1.01} w_5 \xrightarrow{1.14} w_3 \xrightarrow{2.16} w_{15}$

ii. Second tour: $w_{15} \xrightarrow{7.51} w_4 \xrightarrow{2} w_{15}$

iii. Download times: 1.11, 4.40, 1.15, 3.82, 4.12

(b) Recorded travel and unloading times: $\Upsilon_1 = [t_{w_1 \to w_{15}}] + [\sum_{i \in V_2^1} \zeta_i] + [\sum_{j \in V_2^1} \psi_{ij}] + [\Psi_5] = [21.48] + [0.59 + 1.01 + 1.14 + 2.16 + 7.51 + 2] + [1.11 + 4.40 + 1.15 + 3.82] + [4.12] = 50.49$

5. Path followed on the second secondary route:

(a) Peripheral Deposit w_{13}:

i. First tour: $w_{13} \xrightarrow{1.46} w_2 \xrightarrow{1.39} w_3 \xrightarrow{5.01} w_{13}$

ii. Second tour: $w_{13} \xrightarrow{3.67} w_5 \xrightarrow{0.96} w_4 \xrightarrow{0.96} w_{13}$

iii. Download times: 1.71, 3.55, 3.38, 1.91, 5.44

(b) Recorded travel and unloading times: $\Upsilon_2 = [t_{w_3 \to w_{15}} + t_{w_{15} \to w_{13}}] + [\sum_{i \in V_2^2} \zeta_i] + [\sum_{j \in V_2^2} \psi_{ij}] + \Psi_5 = [21.48 + 4.16] + [1.46 + 1.39 + 5.01 + 3.67 + 0.96 + 0.60] + [1.71 + 3.55 + 3.38 + 1.91] + 5.44 = 57.72$

The remaining peripheral nodes are calculated similarly. Therefore, the maximum travel time of the entire network under this scenario is given by

$$\mathcal{Z}_2 \approx \max\{50.49, 57.72, 63.87, 41.69\} = 63.87$$

This means that any product with a shelf life of less than 64 hours cannot go on the trip.

It is important to note that the required demand at each peripheral warehouse is fully met by the proposal, and the excess quantity is stored in the peripheral warehouses as available material (identified in the figure with green letters). These surpluses result from maximum capacity loads of transport vehicles and are generally managed as inventory for customer service.

Although the results shown are encouraging, it is also possible to find infeasible solutions in the model. An infeasible solution is one that violates at least one constraint of the problem; it is a solution that does not satisfy all the requirements or constraints defined in the model. In our case, the most common infeasibility manifests itself when the time window imposed on the model is not met.

5. Comparative Aspects and Discussion of the Results

To illustrate the proposal with larger-scale examples, a comparative analysis of the results is presented below, contrasting our proposal with three well-known metaheuristics in the literature: Ant Colony Optimization (ACO), Simulated Annealing (SA) and Tabu Search Optimization (TS). The size of the cities visited on the primary route has been increased to 15 (although it can actually be scaled to any manageable size of cities visited), as, in real-world cold chain models, a larger number is difficult to manage due to the shelf life of the transported food. Appendix A shows the corresponding Python programs used as a contrast to this proposal [36].

5.1. Model Optimization Using Ant Colony Optimization (ACO)

The first comparative method we will use is a well-known probabilistic metaheuristic called the ant colony algorithm. This technique is classified as a stochastic search algorithm

because ants' decisions about which path to take are based on probabilities (influenced by the amount of pheromones), allowing them to explore different routes and avoid getting stuck in local optima. The method was selected because the process depends not on centralized control but rather on the interaction of multiple agents (ants) working in parallel. Furthermore, it is robust, flexible, and capable of finding favorable solutions in this model.

Algorithm 3 shows the source code using a Python programming language through the corresponding pseudocode.

Algorithm 3 Ant Colony Optimization (ACO) for VRP with Stochastic Data

Input: N (Total Points), Vehicle Capacity C_{max}
Time Distribution $\leftarrow$ Exponential($\lambda = 20$)
Demand Distribution $\leftarrow$ Normal($\mu = 1000, \sigma = 150$)
Ants $\leftarrow 10$, Iterations $\leftarrow 30, \alpha \leftarrow 1.0, \beta \leftarrow 2.0, \rho \leftarrow 0.5$
Initialization:
Generate $D_{n \times n}$ using expovariate(λ)
Generate Q using gauss(μ, σ) where $Q_0 = 0$
Set $\tau_{ij} = 1.0$ for all edges (i, j)
$S_{best} \leftarrow$ Null, $L^* \leftarrow \infty$, $T_{start} \leftarrow$ currentTime()
Main Optimization Loop:
for $t = 1$ **to** *Iterations* **do**
  for each ant $k = 1$ **to** *Ants* **do**
    $Visited \leftarrow \{0\}$, $Current_Solution \leftarrow \emptyset$, $L_k \leftarrow 0$
    while $|Visited| < N$ **do**
      $Route \leftarrow [0]$, $Current_Load \leftarrow 0, i \leftarrow 0$
      while exists $j \notin Visited$ **and** $Current_Load + Q_j \leq C_{max}$ **do**
        Calculate $P_{ij} = \frac{(\tau_{ij})^{\alpha} \cdot (1/d_{ij})^{\beta}}{\sum_{k \in \text{candidates}} (\tau_{ik})^{\alpha} \cdot (1/d_{ik})^{\beta}}$
        Select j via Roulette Wheel Selection
        $Current_Load \leftarrow Current_Load + Q_j$
        $L_k \leftarrow L_k + d_{ij}$
        $Visited \leftarrow Visited \cup \{j\}$
        Add j to $Route$, $i \leftarrow j$
      end while
      $L_k \leftarrow L_k + d_{i0}$ ▷ Add return to depot distance
      Add 0 to $Route$ and store $\{Route, Current_Load\}$
    end while
    if $L_k < L^*$ **then**
      $L^* \leftarrow L_k, S_{best} \leftarrow Current_Solution, Iter_Conv \leftarrow t$
    end if
  end for
  Pheromone Update:
  $\tau_{ij} \leftarrow \tau_{ij} \cdot (1 - \rho)$ ▷ Evaporation Step
  $\tau_{ij} \leftarrow \tau_{ij} + 1/L^*$ ▷ Global Best Reinforcement
end for
$T_{total} \leftarrow$ currentTime() $- T_{start}$
Output: S_{best}, Best Distance L^*, T_{total}, $Iter_Conv$, $Capacity_Usage$

The following are four blocks of results from the heuristic:

1. First block
   (a) Execution time: 0.0210 s.
   (b) Best overall time: 29.6851 days.
   (c) Convergence in iterations: 8.
   (d) Vehicle 1: Route [0, 7, 2, 5, 3, 13, 8, 12, 11, 4, 10, 0] | Load: 9885/10,000.
   (e) Vehicle 2: Route [0, 14, 6, 9, 1, 0] | Load: 4298/10,000.
2. Second block

(a) Execution time: 0.0230 s.
(b) Best overall time: 51.5345 days.
(c) Convergence in iteration: 9.
(d) Vehicle 1: Route [0, 14, 4, 9, 8, 6, 10, 1, 2, 11, 0] | Load: 9390/10,000.
(e) Vehicle 2: Route [0, 13, 12, 7, 5, 3, 0] | Load: 4304/10,000.

3. Third block
   (a) Execution time: 0.0220 s.
   (b) Best overall time: 73.7312 days.
   (c) Convergence in iteration: 6.
   (d) Vehicle 1: Route [0, 13, 10, 14, 2, 8, 3, 1, 7, 5, 0] | Load: 9987/10,000.
   (e) Vehicle 2: Route [0, 11, 6, 9, 4, 12, 0] | Load: 5179/10,000.
4. Fourth block
   (a) Execution time: 0.0230 s.
   (b) Best overall time: 60.1843 days.
   (c) Convergence in iteration: 17.
   (d) Vehicle 1: Route [0, 13, 10, 14, 2, 8, 3, 1, 7, 5, 0] | Load: 9987/10,000.
   (e) Vehicle 2: Route [0, 11, 6, 9, 4, 12, 0] | Load: 5179/10,000.

5.2. Model Optimization Using Tabu Search (TS)

The focus now will be on solving the described problem using Tabu Search. This method employs a single solution that attempts to improve at each step by moving customers between routes. It analyzes its environment step by step and forbids itself from retracing its steps to avoid getting stuck in circles, constantly striving to refine the current route. With Normal Demand ($\mu = 1000$), Tabu Search shines here because the demands are similar. This makes customer swapping between routes very smooth, as they almost always fit the new vehicle. With Exponential Time ($\lambda = 20$), because times are very small and variable, a small movement (moving a customer from Route A to Route B) can cause a drastic change in the total cost. Tabu Search is excellent for fine-tuning these route details. Algorithm 4 shows the pseudocode used to explore this alternative.

Algorithm 4 Tabu Search (TS) for VRP with Stochastic Data

Input: N (Total Points), Vehicle Capacity C_{max}
Time Distribution $\leftarrow$ Exponential($\lambda = 20$)
Demand Distribution $\leftarrow$ Normal($\mu = 1000, \sigma = 150$)
Max Iterations $\leftarrow$ 100, Tabu Tenure $\leftarrow$ 5
Initialization:
Generate $D_{n \times n}$ using expovariate(λ)
Generate Q using gauss(μ, σ) where $Q_0 = 0$
$S_{initial} \leftarrow$ Generate initial feasible solution via Greedy Shuffle
$S_{current} \leftarrow S_{initial}, S_{best} \leftarrow S_{initial}$
$TabuList \leftarrow \emptyset, T_{start} \leftarrow$ currentTime()
Main Optimization Loop:
for $t = 1$ **to** *Max Iterations* **do**
    $S_{candidate} \leftarrow$ Null, $L_{candidate} \leftarrow \infty$, $Best_Move \leftarrow$ Null
    $Neighborhood \leftarrow$ Generate feasible neighbors by Relocating clients
    for each $S_{neighbor}$ produced by $Move(client, route)$ **do**
        $L_{neighbor} \leftarrow$ Calculate Total Travel Time of $S_{neighbor}$

Algorithm 4 *Cont.*

```
        is_Tabu ← (Move ∈ TabuList)
        if not is_Tabu or L_neighbor < L* then                ▷ Aspiration Criterion
            if L_neighbor < L_candidate then
                S_candidate ← S_neighbor
                L_candidate ← L_neighbor
                Best_Move ← Move
            end if
        end if
    end for
    if S_candidate is not Null then
        S_current ← S_candidate
        Update TabuList adding Best_Move for Tabu Tenure iterations
        if L_candidate < L* then
            L* ← L_candidate, S_best ← S_candidate
        end if
    end if
end for
T_total ← currentTime() − T_start
Output: S_best, Best Time L*, T_total, Capacity_Usage
```

Below are four blocks obtained using this method:

1. First block
   (a) Execution time: 0.0320 s.
   (b) Best overall time: 119.5517 days.
   (c) Convergence in iterations: 44.
   (d) Vehicle 1: Route [0, 14, 4, 12, 11, 6, 3, 1, 7, 9, 0] | Load Usage: 8729/10,000.
   (e) Vehicle 2: Route [0, 2, 10, 5, 8, 13, 0] | Load Usage: 4958/10,000.
2. Second block
   (a) Execution time: 0.0320 s.
   (b) Best overall time: 140.9756 days.
   (c) Convergence in iterations: 8.
   (d) Vehicle 1: Route [0, 13, 6, 11, 5, 3, 4, 0] | Load Usage: 5861/10,000.
   (e) Vehicle 2: Route [0, 8, 10, 9, 7, 2, 1, 14, 12, 0] | Load Usage: 8453/10,000.
3. Third block
   (a) Execution time: 0.0330 s.
   (b) Best overall time: 89.5850 days.
   (c) Convergence in iterations: 8.
   (d) Vehicle 1: Route [0, 1, 7, 6, 4, 10, 8, 2, 0] | Load Usage: 7320/10,000.
   (e) Vehicle 2: Route [0, 11, 13, 5, 3, 12, 14, 9, 0] | Load Usage: 7886/10,000.
4. Fourth block
   (a) Execution time: 0.0330 s.
   (b) Best overall time: 150.9715 days.
   (c) Convergence in iterations: 13.
   (d) Vehicle 1: Route [0, 7, 6, 3, 1, 9, 0] | Load Usage: 4863/10,000.
   (e) Vehicle 2: Route [0, 4, 14, 5, 11, 12, 13, 8, 2, 10, 0] | Load Usage: 8846/10,000.

5.3. Model Optimization Using Simulated Annealing (SA)

Another alternative we will explain in this model is the Simulated Annealing technique. Unlike deterministic algorithms, Simulated Annealing is particularly well-suited to handling the probability distributions embedded in this model. The extreme variability

of travel times (modeled with an exponential distribution $\lambda = 20$) and the concentration of loads (modeled with a normal distribution $\mu = 1000$) generate a solution space filled with low-cost "valleys" and inefficiency "peaks." Simulated Annealing, inspired by thermodynamics, allows the algorithm to navigate this irregular surface with great agility.

To apply the model to the given instance, we use the pseudocode shown in Algorithm 5.

Algorithm 5 Simulated Annealing (SA) for VRP with Stochastic Data

Input: N (Total Points), Vehicle Capacity C_{max}
Time Distribution $\leftarrow$ Exponential$(\lambda = 20)$
Demand Distribution $\leftarrow$ Normal$(\mu = 1000, \sigma = 150)$
Initial Temperature $T \leftarrow 100.0$, Cooling Rate $\alpha \leftarrow 0.99$
Minimum Temp $T_{min} \leftarrow 0.01$, Iterations per Step $\leftarrow 50$
Initialization:
Generate $D_{n \times n}$ using expovariate(λ)
Generate Q using gauss(μ, σ) where $Q_0 = 0$
$S_{current} \leftarrow$ Generate initial feasible solution via Greedy Shuffle
$S_{best} \leftarrow S_{current}, L^* \leftarrow Cost(S_{current}), T_{start} \leftarrow$ currentTime()
$Total_Iter \leftarrow Total_Iter + 1$
Main Cooling Loop:
while $T > T_{min}$ **do**
    for $step = 1$ **to** $Iterations_per_Step$ **do**
        $Total_Iter \leftarrow Total_Iter + 1$
        $S_{neighbor} \leftarrow$ Generate neighbor by Relocating one client
        if $Capacity_Check(S_{neighbor}) \leq C_{max}$ **then**
            $\Delta \leftarrow Cost(S_{neighbor}) - Cost(S_{current})$
            if $\Delta < 0$ **or** $random(0,1) < e^{-\Delta/T}$ **then** ▷ Metropolis Criterion
                $S_{current} \leftarrow S_{neighbor}$
                if $Cost(S_{neighbor}) < L^*$ **then**
                $L^* \leftarrow Cost(S_{neighbor}), S_{best} \leftarrow S_{neighbor}$
                $Iter_Conv \leftarrow Total_Iter$
                end if
            end if
        end if
    end for ▷ This closes the temperature-based step loop.
    $T \leftarrow T \times \alpha$ ▷ Cooling Schedule
end while
$T_{total} \leftarrow$ currentTime() $- T_{start}$
Output: S_{best}, Best Time L^*, T_{total}, $Iter_Conv$, $Capacity_Usage$

Below are four blocks of the results obtained with this heuristic.

1. First block
    (a) Execution time: 0.4730 s.
    (b) Best overall time: 71.2908 days.
    (c) Convergence in iterations: 8219.
    (d) Vehicle 1: Route [0, 10, 4, 6, 13, 5, 7, 1, 12, 8, 0] | Load: 9034/10,000.
    (e) Vehicle 2: Route [0, 14, 9, 11, 3, 2, 0] | Load: 5035/10,000.
2. Second block
    (a) Execution time: 0.4710 s.
    (b) Best overall time: 88.4653 days.
    (c) Convergence in iterations: 9839.
    (d) Vehicle 1: Route [0, 6, 5, 8, 13, 7, 14, 11, 10, 12, 4, 0] | Load: 9908/10,000.
    (e) Vehicle 2: Route [0, 2, 3, 9, 1, 0] | Load: 3707/10,000.

3. Third block
   (a) Execution time: 0.4750 s.
   (b) Best overall time: 96.8679 days.
   (c) Convergence in iterations: 6523.
   (d) Vehicle 1: Route [0, 4, 13, 14, 12, 11, 7, 2, 1, 0] | Load: 8423/10,000.
   (e) Vehicle 2: Route [0, 3, 10, 8, 5, 9, 6, 0] | Load: 5456/10,000.
4. Fourth block
   (a) Execution time: 0.4850 s.
   (b) Best overall time: 67.1678 days.
   (c) Convergence in iterations: 6248.
   (d) Vehicle 1: Route [0, 6, 10, 11, 8, 7, 1, 3, 0] | Load: 7142/10,000.
   (e) Vehicle 2: Route [0, 13, 5, 12, 9, 2, 14, 4, 0] | Load: 6934/10,000.

Optimizing Peripheral Route Two

Table 9 shows the average comparative results of our heuristic and those of ant colony, taboo search and Simulated Annealing.

Table 9. Average comparative values.

Heuristics Used	Convergence Time (s)	Iterations Required	Optimal Value
Ant Colony	0.00025	10.00	53.78
Taboo	0.03250	18.25	125.27
Simulated Annealing	0.47760	7707.25	80.94
Our proposal	0.05400	250.00	97.16

The results suggest that the ACO is the most efficient in this process. Again, using this algorithm, the final solution to the problem is obtained by calculating the secondary path of the model. As before, four sample blocks are generated to obtain a comparative average sample.

1. First secondary route (12 cities)
   (a) First block.
      i. Execution time: 0.0160 s.
      ii. Best overall time: 62.1039 days.
      iii. Convergence in iterations: 10.
      iv. Vehicle 1: Route [0, 6, 3, 5, 11, 4, 8, 10, 7, 1, 9, 0] | Load: 9674/10,000.
      v. Vehicle 2: Route [0, 2, 0] | Load: 811/10,000.
   (b) Second block
      i. Execution time: 0.0150 s.
      ii. Best overall time: 77.8051 days.
      iii. Convergence in iterations: 20.
      iv. Vehicle 1: Route [0, 5, 4, 7, 6, 8, 10, 3, 1, 2, 0] | Load: 9226/10,000.
      v. Vehicle 2: Route [0, 9, 11, 0] | Load: 2018/10,000.
   (c) Third block
      i. Execution time: 0.0150 s.
      ii. Best overall time: 22.5694 days.
      iii. Convergence in iterations: 8.
      iv. Vehicle 1: Route [0, 10, 9, 4, 11, 3, 7, 1, 8, 6, 5, 0] | Load: 9854/10,000.
      v. Vehicle 2: Route [0, 2, 0] | Load: 977/10,000.

(d) Fourth block

i. Execution time: 0.0150 s.
ii. Best time total: 29.5032 days.
iii. Convergence in iterations: 16.
iv. Vehicle 1: Route [0, 2, 1, 6, 8, 5, 7, 11, 4, 10, 3, 0] | Load: 9538/10,000.
v. Vehicle 2: Route [0, 9, 0] | Load: 751/10,000.

2. Second secondary route (7 Cities)

(a) First block

i. Execution time: 0.0060 s.
ii. Best overall time: 57.3239 days.
iii. Convergence in iterations: 1.
iv. Vehicle 1: Route [0, 5, 2, 1, 3, 6, 4, 0] | Load: 6390/10,000.

(b) Second block

i. Execution time: 0.0060 s.
ii. Best overall time: 69.6927 days.
iii. Convergence in iterations: 1.
iv. Vehicle 1: Route [0, 5, 3, 6, 2, 4, 1, 0] | Load: 5482/10,000.

(c) Execution time: 0.0060 s.
(d) Best overall time: 76.8019.
(e) Convergence in iterations: 3.
(f) Vehicle 1: Route [0, 6, 1, 2, 4, 3, 5, 0] | Load: 6169/10,000.
(g) Third block

i. Execution time: 0.0060 s.
ii. Best overall time: 22.5694 days.
iii. Convergence in iterations: 3.
iv. Vehicle 1: Route [0, 6, 1, 2, 4, 3, 5, 0] | Load: 6169/10,000.

(h) Fourth block

i. Execution time: 0.0060 s.
ii. Best overall time: 43.1665 days.
iii. Convergence in iterations: 3. Vehicle 1: Route [0, 1, 5, 6, 2, 4, 3, 0] | Load: 6169/10,000.

Thus, assuming a minimum travel time for the primary route of 57.78 days and including the minimum time on the secondary route of 22.56 days, the lower threshold for the time window is 80.34 days. Similarly, the high threshold is given by the worst times on both routes: 125.27 plus 69.6927 days from the second route provide a maximum total of approximately 195 days.

6. On the Convergence of the Algorithm

Algorithms that operate based on random search converge globally with probability 1 almost always towards the global optimum. The fundamental requirement is that the search space be bounded and the number of iterations (samples) tend towards infinity (almost certain convergence). Unfortunately, their convergence rate is almost always slow compared to other optimization methods. The advantage of using this type of algorithm is that, unlike gradient techniques, over time, the entire search region will be explored, as it explores random points across the entire solution space without the risk of getting stuck at a local minimum. However, the technique is limited by requiring a prohibitively large number of iterations to achieve good accuracy.

Recent studies demonstrate that standard random search methods that do not rely on second-order information (i.e., using the second derivative of the objective function to guide the search for a minimum or maximum while simultaneously obtaining information about the function's curvature) converge to a second-order stationary point. The technique is based on showing that convergence is faster because the number of evaluations of the objective function is always linear [37–41].

Unfortunately, studying convergence and accuracy in stochastic models is not as straightforward as in deterministic models. Given the random nature of the models (where algorithms generate sequences of random variables), there are different and subtle types of convergence that affect their measurement. These types evaluate, in different ways, how closely different sequences of solutions approach the desired optimum as the number of iterations increases. Although there is considerable variability in measuring the closeness between two successive solutions, the most common types of convergence are as follows.

1. Near-certain convergence (*n.c.*): This is the strongest type of convergence. It means that the probability of the sequence of random variables converging to the limit is equal to one. It is analogous to point-wise convergence in deterministic analysis.
2. Probability convergence: This mode is weaker than near-certain convergence. It means that, as the number of iterations increases, the probability that the current solution differs from the limit by a non-trivial amount becomes arbitrarily small.
3. Mean convergence (or Lp-convergence): This implies that the expected value of the norm of the difference between the current solution and the limit converges to zero. A common case is mean-square convergence (or L^2 convergence), where the expected value of the squared error is considered. Mean-square convergence implies probability convergence.
4. Distributional convergence (or law convergence): This is the weakest type of convergence. It does not require that the values of the random variables approach each other, but rather requires that the cumulative distribution functions of the sequence converge to the limit distribution function. In optimization, this is often used to characterize the asymptotic behavior of algorithms, frequently related to the Central Limit Theorem.

In our proposal, the focus was on Mean Convergence.

7. Conclusions

This document presents one of the most common variants found in real-world problems related to load planning and vehicle routing in logistics models. We developed the model based on the unique topology present in structures with two or more linked chains. We obtained the mathematical model of the proposed solution as a two-stage optimization alternative. In the first stage, we solved the routing problem for the primary route and, subsequently, implemented this solution for the secondary route problem.

To strengthen our proposal, we compared it with three of the most important existing frameworks for solving NP-hard problems. To simplify the construction of the distance and demand matrices, we included them in the pseudocode before programming them in Python.

The results obtained suggest that the ACO algorithm is highly robust and effective for solving the vehicle routing problem (VRP). Unlike exhaustive search methods, collective memory (pheromone matrix) allowed for finding high-quality solutions in a reduced computational time, even when integrating rigid constraints such as maximum load capacity.

The impact of statistical distributions on logistical reality transformed a theoretical model into a realistic simulation: the exponential distribution ($\lambda = 20$) allowed for modeling travel times and simulating an environment where most points are close to each other.

However, long-duration "jumps" occasionally occur, which is typical in urban areas with heavy traffic and requires further analysis as a topic for prospective research.

Likewise, when using the normal distribution $(\mu = 1000, \sigma = 150)$ to model demand, it was observed that the vehicle fleet tends to standardize its routes, since load variability is controlled. This facilitates fleet planning but requires that vehicle capacity be very well calibrated with respect to the average. Regarding the efficiency and convergence of the methods studied, it is suggested that they cease to show significant improvements after a certain number of cycles (generally between 50 % and 70% of the total iterations). This implies that, to optimize the use of computational resources in the future, an early stopping criterion could be implemented if the route cost does not change after several iterations. With respect to the relationship between capacity and distance, it is concluded that vehicle capacity (Cmax) is the most sensitive factor in the model. A small adjustment to this value can drastically reduce the number of vehicles required (fleet) and, therefore, the total distance traveled. The "Load Used" report (e.g., 2850/3000) allows for identifying how tightly the logistics operation is configured and the margin of error in the face of unexpected orders.

The following epilogue presents the answers to the questions raised as the central focuses of this research (in the same order in which they were presented).

1. It is possible to approximate a solution to the model presented in Figure 1, consisting of two main chains with several sub-links in each chain, and scaling it to a larger number of variables is entirely feasible.
2. The influence of probability distributions is fundamental for obtaining optimal values for the model. The main function of probability distributions is to quantify and model the uncertainty inherent in the problem's input variables (demand, travel times, shelf life of transported products, time windows, and vehicle emptying time). These distributions define the range of all possible outcomes and the probability associated with each, allowing for the exploration of a much richer solution space than a deterministic model. The philosophy of stochastic methods lies in the fact that, instead of optimizing for a single average scenario, the stochastic model considers multiple scenarios, weighted by their probability of occurrence. Regarding the impact on the objective function and the model's constraints, it is the way in which they are incorporated that affects the model, for example, in the form of a conditional expected value, as was our case. Furthermore, distributions allow us to include risk measures such as variance, which is crucial for decision-makers who often seek not only to optimize a utility function but also to assess its volatility and avoid catastrophic outcomes. Therefore, the type of distribution used can influence the choice of optimization algorithm. For example, if normal (Gaussian) distributions are assumed, certain analytical methods can be applied. However, for more complex or nonparametric distributions, Monte Carlo simulation methods or multistage stochastic programming are likely required to approximate the optimal solution.
3. Travel times in the network definitely affect the value of the optimal solution in the model, as they transform the problem by including risk and uncertainty management, producing routes that are more realistic and resilient to changing traffic conditions. In stochastic routing models, the choice of probability distribution is crucial because travel time is not symmetrical (a traffic delay is much more likely than an extreme advance) and must always be positive, as we consider it to be. In general, some authors prefer to use the lognormal distribution to model travel times due to its positive skewness; that is, it reflects the reality that most trips occur near an average, but there are long tails to the right that represent traffic jams or accidents [42,43]. Unlike the normal distribution, the lognormal distribution restricts values to positive

numbers, which is physically necessary for time. Similarly, the gamma distribution is frequently used to model the sum of times in different segments of a route, with almost the same properties as the lognormal distribution.

4. Considering stochastic demand introduced uncertainty, meaning that the initially optimal routes may be suboptimal when the secondary route model is solved. A viable alternative in this context is to balance initial routing costs with potential penalty costs for failing to meet actual demand. This would require different algorithms and metaheuristics to manage customer demand variability through wait-and-see approaches or the creation of flexible sub-routes to adapt to actual demand, impacting vehicle capacity and travel times.
5. It is indeed possible to draw valid conclusions about the relationships between travel times, required demands, and imposed time windows in a stochastic routing model. In fact, the analysis of these interactions was of great importance in evaluating system performance as it allowed for the incorporation of the inherent uncertainty of real-world variables, such as variability in travel times (due to traffic, weather, etc.) and fluctuations in demand. The use of the Monte Carlo method allowed us to obtain the following conclusions.
    (a) Travel times and time windows: The model revealed how variability in travel times affects service punctuality. In our case, it is possible to assess the probability of a vehicle arriving at a delivery point within a specific time window. Increased travel time variability may require longer buffer times to meet time windows, potentially increasing operating costs or extending routes.
    (b) The model demonstrates that changes in demand (in volume and/or location) impact the ability to meet time constraints. For example, unexpected demand spikes in a congested area can make it impossible to meet tight time windows for some customers, leading to the need for capacity management strategies or dynamic rescheduling.
    (c) Interaction of the three variables: Further analysis explores how they influence each other. It can be concluded, for example, that high demand combined with significant uncertainty in travel times makes tight time windows a significant risk factor for routing efficiency and reliability.

Finally, regarding scalability and execution time, the model could be scaled vertically as long as this time is monitored. The measurement implemented with 'time.time()' is vital for determining the program's operational limit before requiring adjustments, for example, to the number of ants or the evaporation rate when using ACO. In summary, the developed system not only finds a short route but also offers a decision-making framework based on statistical data, enabling smarter logistics planning adapted to real-world uncertainty.

Author Contributions: Conceptualization, G.P.-L.; Methodology, F.V.-M.; Software, G.P.-L.; Validation, F.V.-M.; Formal analysis, G.P.-L. and F.V.-M.; Investigation, G.P.-L.; Writing—review & editing, G.P.-L.; Project administration, G.P.-L. and F.V.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Computer Programs in Python

Listing A1. ACO VRP Simulation.

```
#Copyright  Corporate Engineering, Mexico
# Author: Gilberto Perez Lechuga, PhD in Engineering
# Author: Francisco Venegas Martinez, PhD in Mathematics
import random
import time

def solve_aco_vrp_final_complete():
    # --- 1. DATA CONFIGURATION ---
    n_total = 15
    mean_demand = 1000
    std_dev_demand = 150
    max_capacity = 10000
    lambda_param = 1/20

    # A. Travel Time Matrix (Exponential)
    distances = [[0.0 for _ in range(n_total)] for _ in range(n_total)]
    for i in range(n_total):
        for j in range(i + 1, n_total):
            travel_time = random.expovariate(lambda_param)
            value = round(travel_time, 4)
            distances[i][j] = value
            distances[j][i] = value

    # B. Demands (Normal)
    demands = [0] # Depot
    for _ in range(n_total - 1):
        d = random.gauss(mean_demand, std_dev_demand)
        demands.append(int(max(1, d)))

    # ACO Parameters
    N_ANTS = 10
    ITERATIONS = 30
    ALPHA, BETA, EVAPORATION = 1.0, 2.0, 0.5
    pheromones = [[1.0 for _ in range(n_total)] for _ in range(n_total)]

    global_best_distance = float('inf')
    global_best_solution = []
    convergence_iteration = 0

    # --- 2. OPTIMIZATION ---
    start_time = time.time()

    for iter_idx in range(ITERATIONS):
        iteration_solutions = []
        for _ in range(N_ANTS):
            visited = {0}
            ant_solution = []
            ant_total_distance = 0

            while len(visited) < n_total:
                vehicle_route = [0]
                current_load = 0
                current_node = 0
                while True:
                    probabilities, candidates = [], []
                    for next_node in range(n_total):
                        if next_node not in visited and current_load +
    demands[next_node] <= max_capacity:
                            tau = pheromones[current_node][next_node] **
    ALPHA
```

https://doi.org/10.3390/logistics10010013

```
                            d = distances[current_node][next_node] if distances[current_node][next_node] > 0.0001 else 0.0001
                            eta = (1.0 / d) ** BETA
                            probabilities.append(tau * eta)
                            candidates.append(next_node)
                    if not candidates: break
                    selected = random.choices(candidates, weights=probabilities)[0]
                    vehicle_route.append(selected)
                    current_load += demands[selected]
                    ant_total_distance += distances[current_node][selected]
                    visited.add(selected)
                    current_node = selected

                ant_total_distance += distances[current_node][0]
                vehicle_route.append(0)
                ant_solution.append({"route": vehicle_route, "load": current_load})
            iteration_solutions.append((ant_solution, ant_total_distance))

            if ant_total_distance < global_best_distance:
                global_best_distance = ant_total_distance
                global_best_solution = ant_solution
                convergence_iteration = iter_idx + 1

        # Pheromone Update
        for i in range(n_total):
            for j in range(n_total):
                pheromones[i][j] *= (1 - EVAPORATION)
        for sol, dist in iteration_solutions:
            contribution = 1.0 / dist
            for r in sol:
                route = r['route']
                for k in range(len(route) - 1):
                    pheromones[route[k]][route[k+1]] += contribution

    end_time = time.time()
    print(f"Execution Time: {end_time - start_time:.4f} s")
```

Listing A2. VRP Solution using Tabu Search Metaheuristic.

```
#Copyright © Corporate Engineering, Mexico
# Author: Gilberto Perez Lechuga, PhD in Engineering
# Author: Francisco Venegas Martinez, PhD in Mathematics

import random
import time
import copy

def solve_vrp_tabu_integrated():
    # --- 1. DATA CONFIGURATION (2025 Scenario) ---
    n_total = 15  # Depot + 14 Customers
    vehicle_capacity = 3500
    demand_mean, demand_std = 1000, 150
    lambda_param = 1/20 # For Exponential Distribution (Mean = 20)

    # Time Matrix (Exponential Distribution)
    distances = [[0.0 for _ in range(n_total)] for _ in range(n_total)]
    for i in range(n_total):
        for j in range(i + 1, n_total):
            val = round(random.expovariate(lambda_param), 4)
            distances[i][j] = distances[j][i] = val

    # Customer Demands (Normal Distribution)
```

```
    demands = [0] # Index 0 is the Depot
    for _ in range(n_total - 1):
        d = random.gauss(demand_mean, demand_std)
        demands.append(int(max(1, d)))

    # --- 2. SUPPORT FUNCTIONS ---
    def calculate_total_cost(solution):
        total_cost = 0
        for route in solution:
            for i in range(len(route) - 1):
                total_cost += distances[route[i]][route[i+1]]
        return round(total_cost, 4)

    def generate_initial_solution():
        clients = list(range(1, n_total))
        random.shuffle(clients)
        solution = []
        while clients:
            route = [0]
            current_load = 0
            while clients and current_load + demands[clients[0]] <=
    vehicle_capacity:
                client = clients.pop(0)
                route.append(client)
                current_load += demands[client]
            route.append(0)
            solution.append(route)
        return solution

    # --- 3. Tabu SearcH ALGORITHM ---
    MAX_ITERATIONS = 100
    TABU_TENURE = 10

    current_solution = generate_initial_solution()
    best_solution = copy.deepcopy(current_solution)
    best_cost = calculate_total_cost(best_solution)

    # Tabu List Structure: {(client, target_route): expiration_iteration}
    tabu_list = {}

    print(f"Initial Cost: {best_cost}")

    for iteration in range(MAX_ITERATIONS):
        neighborhood = []

        # Generate Neighborhood (Relocate moves)
        for r_idx, route in enumerate(current_solution):
            if len(route) <= 3: continue

            for i in range(1, len(route) - 1):
                client = route[i]

                for r_dest_idx, route_dest in enumerate(current_solution):
                    # Capacity Constraint Check
                    dest_load = sum(demands[c] for c in route_dest)
                    if r_idx != r_dest_idx and dest_load + demands[client]
    > vehicle_capacity:
                        continue

                    # Try insertions in the destination route
                    for j in range(1, len(route_dest)):
                        new_sol = copy.deepcopy(current_solution)
                        moving_client = new_sol[r_idx].pop(i)
```

```
                        new_sol[r_dest_idx].insert(j, moving_client)

                        # Remove empty routes
                        new_sol = [r for r in new_sol if len(r) > 2]
                        cost = calculate_total_cost(new_sol)

                        # Move definition for Tabu List
                        move = (client, r_dest_idx)
                        neighborhood.append((new_sol, cost, move))

        if not neighborhood: break
        neighborhood.sort(key=lambda x: x[1])

        # Select the best non-tabu move (or apply Aspiration Criterion)
        found_move = False
        for next_sol, next_cost, move in neighborhood:
            if move not in tabu_list or tabu_list[move] <= iteration or next_cost < best_cost:
                current_solution = next_sol
                tabu_list[move] = iteration + TABU_TENURE

                if next_cost < best_cost:
                    best_solution = copy.deepcopy(next_sol)
                    best_cost = next_cost
                found_move = True
                break

        if not found_move: break

    return best_solution, best_cost

# Execute
best_sol, final_cost = solve_vrp_tabu_integrated()
print(f"\n--- Final Results 2025 ---")
print(f"Best Cost: {final_cost}")
for idx, route in enumerate(best_sol):
    print(f"Route {idx+1}: {route}")
```

Listing A3. VRP Solution using Simulated Annealing Metaheuristic.

```
#Copyright © Corporate Engineering, Mexico
# Author: Gilberto Perez Lechuga, PhD in Engineering
# Author: Francisco Venegas Martinez, PhD in Mathematics

import random
import time
import math
import copy

def solve_vrp_simulated_annealing():
    # --- 1. DATA CONFIGURATION ---
    n_total = 15
    vehicle_capacity = 10000
    demand_mean, demand_std = 1000, 150
    lambda_param = 1/20 # Average times of 1/20 = 0.05

    # Distance Matrix (Exponential Distribution)
    distances = [[0.0 for _ in range(n_total)] for _ in range(n_total)]
    for i in range(n_total):
        for j in range(i + 1, n_total):
            val = round(random.expovariate(lambda_param), 4)
            distances[i][j] = distances[j][i] = val

    # Customer Demands (Normal Distribution)
```

```
    demands = # Depot at index 0
    for _ in range(n_total - 1):
        d = random.gauss(demand_mean, demand_std)
        demands.append(int(max(1, d)))

    # --- 2. SUPPORT FUNCTIONS ---
    def calculate_total_cost(solution):
        total_cost = 0
        for route in solution:
            for i in range(len(route) - 1):
                total_cost += distances[route[i]][route[i+1]]
        return round(total_cost, 4)

    def generate_initial_solution():
        clients = list(range(1, n_total))
        random.shuffle(clients)
        solution = []
        while clients:
            route =
            current_load = 0
            while clients and current_load + demands[clients] <=
    vehicle_capacity:
                client = clients.pop(0)
                route.append(client)
                current_load += demands[client]
            route.append(0)
            solution.append(route)
        return solution

    def get_neighbor(solution):
        # Implementation of a simple swap move (intra-route)
        new_solution = copy.deepcopy(solution)
        r_idx = random.randint(0, len(new_solution) - 1)
        route = new_solution[r_idx]
        if len(route) > 3:
            i, j = random.sample(range(1, len(route) - 1), 2)
            route[i], route[j] = route[j], route[i]
        return new_solution

    # --- 3. Simulated AnnealING ALGORITHM ---
    current_solution = generate_initial_solution()
    current_cost = calculate_total_cost(current_solution)
    best_solution = copy.deepcopy(current_solution)
    best_cost = current_cost

    T_start = 100.0
    T_end = 0.1
    alpha = 0.99
    T = T_start

    while T > T_end:
        neighbor_solution = get_neighbor(current_solution)
        neighbor_cost = calculate_total_cost(neighbor_solution)

        cost_diff = neighbor_cost - current_cost

        if cost_diff < 0 or random.random() < math.exp(-cost_diff / T):
            current_solution = neighbor_solution
            current_cost = neighbor_cost

            if current_cost < best_cost:
                best_solution = copy.deepcopy(current_solution)
                best_cost = current_cost
```

```
        T *= alpha

    return best_solution, best_cost

# Execution
# sol, cost = solve_vrp_simulated_annealing()
# print(f"Best Cost: {cost}")
```

References

1. Dantzig, G.; Fulkerson, R.; Johnson, S. Solution of a Large-Scale Traveling-Salesman Problem. *J. Oper. Res. Soc. Am.* **1954**, *2*, 393–410. [CrossRef]
2. Pérez-Lechuga, G.; Martínez-Sánchez, J.F.; Venegas-Martínez, F.; Madrid-Fernández, K.N. A Routing Model for the Distribution of Perishable Food in a Green Cold Chain. *Mathematics* **2024**, *12*, 332. [CrossRef]
3. Soto-Mendoza, V.; Ruiz-y-Ruiz, E.; García-Calvillo, I.; Nucamendi-Guillén, S.; Cardona-Valdés, Y. A location-routing problem for local supply chains. *Comput. Ind. Eng.* **2023**, *183*, 109528. [CrossRef]
4. Agárdi, A.; Kovács, L.; Bányai, T. Mathematical Model for the Generalized VRP Model. *Sustainability* **2022**, *14*, 11639. [CrossRef]
5. Dornemann, J. Solving the capacitated vehicle routing problem with time windows via graph convolutional network assisted tree search and quantum-inspired computing. *Front. Appl. Math. Stat.* **2023**, *9*, 1155356. [CrossRef]
6. Soares, R.; Marques, A.; Amorim, P.; Parragh, S.N. Synchronisation in vehicle routing: Classification schema, modelling framework and literature review. *Eur. J. Oper. Res.* **2024**, *313*, 817–840. [CrossRef]
7. Archetti, C.; Fernandez, E.; Huerta, D. The Flexible Periodic Vehicle Routing Problem. *Comput. Oper. Res.* **2017**, *85*. [CrossRef]
8. Contardo, C.; Martinelli, R. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discret. Optim.* **2014**, *12*, 129–146. [CrossRef]
9. What Is a Warehouse Management System (WMS)? Available online: https://www.oracle.com/scm/logistics/warehouse-management/what-is-warehouse-management/#:~:text=de%20Oracle%20(WMS)-,%C2%BFQu%C3%A9%20es%20un%20sistema%20de%20gesti%C3%B3n%20de%20almacenes%20(WMS)%3F,el%20estante%20de%20la%20tienda (accessed on 15 May 2025).
10. Silvertouch SAP. SAP for Warehouse Management—Its Types, Modules, Benefits, Features, and Price Details. Available online: https://sap.silvertouch.com/blog/sap-erp-for-warehouse-management-system-modules-benefits-features-andprice#:~:text=Simply%20put%2C%20the%20SAP%20warehouse,all%20warehouse%2Drelated%20operations%20efficiently (accessed on 17 May 2025).
11. Boumpa, E.; Tsoukas, V.; Chioktour, V.; Kalafati, M.; Spathoulas, G.; Kakarountas, A.; Trivellas, P.; Reklitis, P.; Malindretos, G. A Review of the Vehicle Routing Problem and the Current Routing Services in Smart Cities. *Analytics* **2023**, *2*, 1. [CrossRef]
12. Liu, X.; Chen, Y.-L.; Por, L.Y.; Ku, C.S. A Systematic Literature Review of Vehicle Routing Problems with Time Windows. *Sustainability* **2023**, *15*, 12004. [CrossRef]
13. Chabot, T.; Coelho, L.C.; Renaud, J.; Roodbergen, K.J. Mathematical Models for the Warehouse Reassignment Problem. In *Warehousing and Material Handling Systems for the Digital Industry*; Manzini, R., Accorsi, R., Eds.; Springer: Cham, Switzerland, 2024. [CrossRef]
14. Li, X.; Tian, P.; Leung, S.C.H. Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *Int. J. Prod. Econ.* **2010**, *125*, 137–145. [CrossRef]
15. Pérez Lechuga, G. Optimal logistics strategy to distribute medicines in clinics and hospitals. *J. Math. Ind.* **2018**, *8*, 2. [CrossRef]
16. Pérez-Lechuga, G.; Aguilar-Velázquez, S.L.; Cisneros-López, M.A.; Martínez, F.V. A model for the location and scheduling of the operation of second-generation ethanol biorefineries. *J. Math. Ind.* **2019**, *9*, 3. [CrossRef]
17. Widuch, J. Chapter 1—Current and emerging formulations and models of real-life rich vehicle routing problems. In *Intelligent Data-Centric Systems, Smart Delivery Systems*; Nalepa, J., Ed.; Elsevier: Amsterdam, The Netherlands, 2020; pp. 1–35. [CrossRef]
18. Stewart, W.R., Jr.; Golden, B.L. Stochastic vehicle routing: A comprehensive approach. *Eur. J. Oper. Res.* **1983**, *14*, 371–385. [CrossRef]
19. Ulmer, M.W.; Goodson, J.C.; Mattfeld, D.C.; Thomas, B.W. On modeling stochastic dynamic vehicle routing problems. *EURO J. Transp. Logist.* **2020**, *9*, 100008. [CrossRef]
20. Adulyasak, Y.; Jaillet, P. Models and algorithms for stochastic and robust vehicle routing with deadlines. *Transp. Sci.* **2015**, *50*, 608–626. [CrossRef]
21. Gómez, A.; Mariño, R.; Akhavan-Tabatabaei, R.; Medaglia, A.L.; Mendoza, J.E. On Modeling Stochastic Travel and Service Times in Vehicle Routing. *Transp. Sci.* **2015**, *50*, 627–641. [CrossRef]

22. Elgharably, N.; Easa, S.; Nassef, A.; El Damatty, A. Stochastic Multi-Objective Vehicle Routing Model in Green Environment with Customer Satisfaction. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 1337–1355. [CrossRef]
23. Mardešić, N.; Erdelić, T.; Carić, T.; Đurasević, M. Review of Stochastic Dynamic Vehicle Routing in the Evolving Urban Logistics Environment. *Mathematics* **2024**, *12*, 28. [CrossRef]
24. Gendreau, M.; Jabali, O.; Rei, W. 50th Anniversary Invited Article—Future Research Directions in Stochastic Vehicle Routing. *Transp. Sci.* **2016**, *50*, 1163–1173. [CrossRef]
25. Marinakis, Y.; Iordanidou, G.-R.; Marinaki, M. Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands. *Appl. Soft Comput.* **2013**, *13*, 1693–1704. [CrossRef]
26. Herazo-Padilla, N.; Montoya-Torres, J.R.; Nieto Isaza, S.; Alvarado-Valencia, J. Simulation-optimization approach for the stochastic location-routing problem. *J. Simul.* **2015**, *9*, 296–311. [CrossRef]
27. Akbarpour, N.; Salehi-Amiri, A.; Hajiaghaei-Keshteli, M.; Oliva, D. An innovative waste management system in a smart city under stochastic optimization using vehicle routing problem. *Soft Comput.* **2021**, *25*, 6707–6727. [CrossRef]
28. Iklassov, Z.; Sobirov, I.; Solozabal, R.; Takáč, M. Reinforcement Learning Approach to Stochastic Vehicle Routing Problem with Correlated Demands. *IEEE Access* **2023**, *11*, 87958–87969. [CrossRef]
29. Iklassov, Z.; Sobirov, I.; Solozabal, R.; Takáč, M. Reinforcement Learning for Solving Stochastic Vehicle Routing Problem with Time Windows. 2024. Available online: https://arxiv.org/pdf/2402.09765 (accessed on 22 November 2025).
30. Li, Z.; Jiao, P. Two-stage stochastic programming for the inventory routing problem with stochastic demands in fuel delivery. *Int. J. Ind. Eng. Comput.* **2022**, *13*, 507–522. [CrossRef]
31. Bernardo, M.; Pannek, J. Robust Solution Approach for the Dynamic and Stochastic Vehicle Routing Problem. *J. Adv. Transp.* **2018**, *2018*, 9848104. [CrossRef]
32. Ruszczyński, A.; Shapiro, A. Stochastic Programming Models. In *Handbooks in Operations Research and Management Science*; Elsevier: Amsterdam, The Netherlands, 2003; Volume 10, pp. 1–64. [CrossRef]
33. García Calvillo Irma, D. Un Enfoque Metaheurístico Para un Problema de Ruteo con Flexibilidad en las Fechas de Entrega. Ph.D. Thesis, Engineering, Facultad de Ingeniería Mecánica y Eléctrica, Universidad de Nuevo León, San Nicolás de los Garz, México, 2010.
34. Chao, I.M.; Golden, B.L.; Wasil, E.A. A New Algorithm for the Site-dependent Vehicle Routing Problem. In *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*; Woodruff, D.L., Ed.; Kluwer Academic Publishers: Boston, MA, USA; Dordrecht, The Netherlands; London, UK, 1997; pp. 301–312.
35. Vajda, S. *Probabilistic Programming; Probabilistic and Mathematical Statistics*; Birnbaum, Z.W., Lukacs, E., Eds.; Academic Press: New York, NY, USA; London, UK, 1972; pp. 27–35.
36. Free Software, Open Standards, and Web Services for Interactive Computing Across All Programming Languages (Introduction to the JupyterLab and Jupyter Notebooks). Available online: https://jupyter.org/try-jupyter/lab/ (accessed on 15 May 2025).
37. Pérez-Lechuga, G. A Perspective on Stochastic Search Efficiency via Quasigradient Techniques in Constrained Models. *Am. J. Oper. Res.* **2025**, *15*, 195–221. [CrossRef]
38. Ombach, J. A proof of convergence of general stochastic search for global minimum. *J. Differ. Equ. Appl.* **2007**, *13*, 795–802. [CrossRef]
39. Regis, R. On the Convergence of Adaptive Stochastic Search Methods for Constrained and Multi-objective Black-Box Optimization. *J. Optim. Theory Appl.* **2016**, *170*, 932–959. [CrossRef]
40. Li, X.; Milzarek, A. A Unified Convergence Theorem for Stochastic Optimization Methods. Available online: https://proceedings.neurips.cc/paper_files/paper/2022/file/d630537fc4402cfa3ebbc7450a0cac91-Paper-Conference.pdf (accessed on 20 November 2025).
41. Pérez-Lechuga, G.; Tuoh-Mora, J.; Morales-Sánchez, E.; Suárez-Álvarez, M. On the Efficiency of a Random Search Method. 2005. Available online: https://www.researchgate.net/publication/241769436_726 (accessed on 27 November 2025).
42. Li, W.; Duan, Z.; Guo, G. Research on Travel Time Distribution Characteristics of Expressways in Shanghai. *Procedia—Soc. Behav. Sci.* **2013**, *96*, 339–350. [CrossRef]
43. Khavarian, K.; Samimi, A.; Rashidi, T.H. Travel time variation and its impact on the elasticity of route choice model parameters. *Travel Behav. Soc.* **2023**, *33*, 100648. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.