CROWDSOURCING VRP

Marco Colocrese s301227
ICT for Smart Societies
Politecnico di Torino
Torino, TO 10129

Francesco Donato s304810
ICT for Smart Societies
Politecnico di Torino
Torino, TO 10129

April 7, 2023

Academic Year 2021/2022
Politecnico di Torino

OPERATIONAL RESEARCH: THEORY AND APPLICAITON
Professors: Edoardo Fadda and Emilio Leonardi

1 Introduction

Our work consists in finding an heuristic solution for a variant of the Probabilistic Travelling Salesman Problem, emerging when retailers crowdsource last-mile deliveries to their own customers, who can refuse or accept in exchange for a reward. Our task is to identify which deliveries to offer, knowing that all deliveries need fulfilment, either via crowdsourcing or using the retailer's own vehicle, and then finding a solution to minimize a specific cost function, related to the built route per each vehicle in charge of deliveries. Given the high complexity of the problem, it is necessary to use heuristic methods in order to reduce the number of evaluations for simulation time purposes.

In this paper, we propose an heuristic method based on clustering to take decisions about which delivery to assign to crowdsource and two heurisitcs methods to build the deliveries route. We propose two methods, as one of them (which uses Simulated Annealing and Genetic Algorithm) is giving good results in therms of objective function cost, however we believe that it would not scale to much higher number of deliveries in the sense of time needed to for computation, while the other method (based on insertion heuristic) is much faster and more scalable.

2 Problem Statement and Definition

2.1 Vehicle Routing Problem with Time Windows

The vehicle routing problem with time windows (VRPTW) is an important issue in logistics systems which has been researched widely in recent years. The problem can be described as choosing a set of routes, starting and ending at a depot, for a limited number of vehicles to serve a group of customers which have a given demand, and no vehicle can service more customers than its capacity permits. Each customer should be served exactly once. The objective of the VRPTW is to minimize the total transport costs. We consider the vehicle routing problem with time windows (VRPTW), which is a generalization of the VRP where the service at any customer starts within a given time interval, called a time window. If a vehicle arrives too early at a customer, it must wait until the time window opens, while if it arrives to late it is no more allowed to serve that customer, as it does not respect the related constraint.

2.2 Crowdsourcing

In the problem we are considering, retailers crowdsource last-mile deliveries to their own customers, who can refuse or accept in exchange for a reward. One of the objectives is to identify which deliveries to offer to external people, knowing that all deliveries need fulfillment, either via crowdsourcing or using the retailer's own vehicle, being aware that this strategy can bring an important improvement to the VRP solution, where it is important to balance the performance of both the VRPTW and the crowdsourcing strategy in order to obtain a feasible solution as optimal as possible.

As the optimization of the crowdsourcing fees has already been studied (Fadda et al.[1]), in this problem we assume that they are fixed and we take them as an input to our problem.

3 Mathematical Model

In this section, the detailed description of the problem is explained.

3.1 Overall Model

Let $G = (V', E)$ be a complete undirected graph with a set of vertexes $V' = \{0, 1, ..., n\}$, where vertex 0 represents the depot. The notation $c_{ij} \in \mathbb{R}^+$ represents the cost of traversing an edge $\{i, j\} \in E$. When a delivery $i \in V$ is offered for crowdsourcing, there is a probability $p_i \in [0, 1]$ that some provider will accept the offer. When this happens, the decision-maker pays a fee $m_i \in \mathbb{R}^+$ and removes i from the list of customers to visit. If the offer is not accepted, the planner has to visit i with the retailer's own vehicle.

The subset of deliveries offered for crowdsourcing is denoted with $O \subseteq V$ and the set of accepted offers with $A \subseteq O$, which is revealed at the end of the day. Figure 1 shows the scheme representing the sets described.

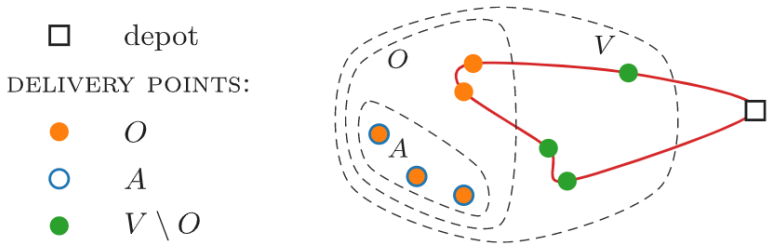


Figure 1: Relation between sets V, O and A.

The optimization model has as output the expected value of the total cost:

$$E_A[C(O)] = \sum_{(A \subseteq O)} \left[\left(\prod_{i \in A} p_i \prod_{i \in O/A} (1 - p_i) \right) * \left(\sum_{i \in A} m_i + c_{V'/A} \right) \right] \tag{1}$$

The objective of the problem is to find the set O^{opt} which gives the lowest expected cost:

$$O^{\circ pt} = \arg \min_{O \subseteq V} E_A[C(O)] \tag{2}$$

3.2 VRPTW Model

Let $G = (V, A)$ be a graph consisting of a set of $N + 1$ nodes, in which $O \subseteq V$ is the subset of deliveries offered for crowdsourcing and $A \subseteq O$ is considered as the set of accepted offers. The remaining nodes are considered as input data to the VRPTW (Vehicle Routing Problem with Time Windows), where every remaining node can be services only within a specified time interval called time window. There are $N + 1$ customers and K vehicles, where the depot node is denoted as customer 0. For each route a different vehicle is used and each arc in the network represents a connection between two nodes associated with a cost c_{ij} and travel time t_{ij}. The distances were provided through a file, while the speed of the vehicles is assumed to be unity. Regarding the capacity of the vehicles and the customers demand, the first one is represented by the constant q_k, while the second one can vary according to the customer and is represented by the variable m_i. The capacity of the single vehicle must be greater or equal to the total customers demand on the route traveled by the k^{th} vehicle, meaning that a vehicle cannot be overloaded. The earliest and latest arrival time define the time-window constraint, which must be respected by the vehicles that must arrive at the customers no later than the latest arrival time, while if the vehicle arrives earlier than the earliest arrival time, it must wait, resulting in a waste of resources. There is also a total route time that the vehicles are supposed to respect when completing their routes, represented by the time window of the depot. In addition, there is a time for serving the customer defined as service time f_i at node i and is assumed to be unique independently from the quantity of the load handled.

3.2.1 Mathematical Model

$$min \quad \sum_{i=0}^{N} \sum_{j=0, j\neq i}^{N} \sum_{k=1}^{K} c_{ij} x_{ijk} \tag{3}$$

$s.t.:$

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{ijk} \leq K \quad for \quad i = 0 \tag{4}$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{jik} \leq 1 \quad for \quad i = 0 \quad and \quad k \in \{1, \ldots, K\} \tag{5}$$

$$\sum_{k=1}^{K} \sum_{j=0 j\neq i}^{N} x_{ijk} = 1 \quad for \quad i \in \{1, \ldots, N\} \tag{6}$$

$$\sum_{k=1}^{K} \sum_{i=0 j\neq i}^{N} x_{ijk} = 1 \quad for \quad j \in \{1, \ldots, N\} \tag{7}$$

$$\sum_{i=1}^{N} m_i \sum_{j=0, j\neq i}^{N} x_{ijk} \leq q_k \quad for \quad k \in \{1, \ldots, K\} \tag{8}$$

$$\sum_{i=0}^{N} \sum_{j=0 j\neq i}^{N} x_{ijk} \left(t_{ij} + f_i + w_i\right) \leq r_k \quad for \quad k \in \{1, \ldots, K\} \tag{9}$$

$$t_0 = w_0 = f_0 = 0 \tag{10}$$

$$\sum_{k=1}^{K} \sum_{j=0, j\neq i}^{N} x_{ijk} \left(t_i + t_{ij} + f_i + w_i\right) \leq t_j \quad for \quad j \in \{1, \ldots, N\} \tag{11}$$

$$e_i \leq \left(t_i + w_i\right) \leq l_i \quad for \quad j \in \{1, \ldots, N\} \tag{12}$$

$$\begin{cases} x_{ijk} = 0 & \text{if there is no arc from node i to node j} \\ x_{i,k} = 1 & \text{otherwise} \end{cases} \qquad \text{i} \neq \text{j; i, j} \in \{0, 1, 2, \ldots, N\}$$

The formulas above represent the constraints defining the conditions to respect in order to have feasible solutions. The objective function of the problem is defined in 3, while eq. 4 specifies the maximum number of routes exiting from the depot. As mentioned previously, we need a constraint that makes sure that every route starts and ends at the central depot (eq. 5). Equations 6 and 7 ensure that a vehicle can visit every customer node only once, while eq. 8 is a capacity constraint. The time windows are specified in equations 10 and 12, while constraint 9 defines the maximum travel time.

Parameter	Meaning
K	Total number of vehicles
N	Total number of customers
y_i	Arbitrary real number
d_{ij}	Euclidean distance between node i and node j
c_{ij}	Cost incurred on arc from node i to j
t_{ij}	Travel time between node i and j
m_i	Demand at node i
q_k	Capacity of vehicle k
e_i	Earliest arrival time at node i
l_i	Latest arrival time at node i
f_i	Service time at node i
r_k	Maximum route time allowed for vehicle k
t_i	Arrival time at node i
w_i	Wait time at node i

4 Literature review

The relevant increasing in delivery experienced during the last decades has caused an increasing interest in founding optimized solutions for the Vehicle Routing Problem. Since the first definition of the VRP problem, introduced in 1959 by Dantzig and Ramser[2] (they proposed a procedure based on linear programming to obtain a near optimal solution to assign stations to trucks such that station demands are satisfied and total mileage covered by the fleet is minimized), many possible solutions have been proposed.

Currently, no optimal solutions can be used for enough big instances, like the ones we are targeting which are close to everyday delivery scenarios (in the order of hundreds).

4.1 Classical solutions

In literature, the first solutions were only heuristics, some of which are currently used, such as The Clarke and Wright [3] one based on iterative merging solution between different routes (it basically starts with individual route for each customer and merges routes iteratively as long as the combination can reduce cost and meet the constraints until no further merge is possible). Some improvements of this type of solution have been proposed throughout the years, such as better sorting algorithms ore more complex data structures or optimized merging algorithms. Another interesting 'classical solution' is the Fisher and Jaikumar algorithm[4], which is based on two sub-processes: the creation of clusters and the implementation of a TSP solution for each of them. We took inspiration from this solution to apply a cluster-optimized cost estimation while computing the initial solution to assign the points to be crowdshipped (see section 5.1).

More recent solutions have moved to a metaheuristic approach, expanding the search space overtaking and exploiting both local searches and constructive heuristic approaches. Two examples are Simulated Annealing (SA) and Tabu search (TS).

Simulated Annealing takes its origin from metallurgy[5] and was adapted to the Operational Research field during 1970s by Martin Pincus [6]. It is an optimization technique inspired by the annealing process of solids. The algorithm attempts

to find the global minimum/maximum of a given function by treating it as a physical system with a temperature-like parameter that controls the probability of accepting worse solutions as the temperature cools down. It begins with a random solution and repeatedly makes small changes to the solution, accepting the change if it results in a lower energy (i.e. better solution) and sometimes accepting it if it results in a higher energy (i.e. worse solution) with a probability that decreases as the temperature decreases. The hope is that by allowing the algorithm to escape local minima (i.e. solutions that are not as good as the global minimum) early on when the temperature is high, it will eventually converge to the global minimum as the temperature cools down.

The Tabu Search optimization technique was firstly proposed by Fred Glover[7]. It explores the solution space by constantly replacing the current solution with the best non visited neighboring solution. Throughout the years, the performance of TS has been improved though new local search techniques which improve the mechanism of preventing to be trapped into a local optima, forbidding the selection of already visited solutions. Also with TS, the algorithm is able to escape from a local minimum as worsening solutions can be accepted. Moreover, cycling visiting is forbidden through visited solutions storage. This aspect has been object of different papers that aim at optimizing the memory usage, such as the idea to store last movements rather than the entire last visited solutions or the the two-level tree data structure proposed by Dorabela Gamboa, Cesar Rego and Fred Glover in 2005 [8] that aims at improving the efficiency reached through classical data structures.

4.2 More recent solutions

In this part, we discuss some of the studies which finally brought to the optimization problem we are facing, previously defined in section 2.

Moving to a better reality-fitting, the problem in last years has moved to the Vechile Routing Problem with Occasional Drivers (OD), defined by Walmart and Amazon with the introduction of 'crowdshipping': the possibility to have some deliveries taken by ordinary customers or other people not employed by the companies. An interesting wide field of research has been developed around the OD figure as high importance has to be given to these elements behavior in optimizing the overall deliveries cost: in particular many studies have been developed to understand and predict the acceptance probability of a particular purchase. The first papers introducing this possibility considered ODs always accepting the delivery task. However, this does not actually fit the real scenario: they consider to statically assign all deliveries to ODs and only the remaining ones to professional fleet; a solution example is the multi-start heuristic approach proposed by Archetti et al[9]. Dynamicity among the day was then introduced by Dayarian and Savelsbergh[10] developing two rolling horizon dispatching approaches: one that considers only the state of the system when making decisions, and one that also incorporates probabilistic information about future online order and in-store customer arrivals. Probability of acceptance was also discussed by Katarzyna Gdowska, Ana Viana and João Pedro Pedroso[11] based on the analysis of the OD's willingness to accept or reject delivery tasks and the influence of their decision on the total delivery cost. Moreover, while Archetti et al[9] considered fixed fees for ODs, Katarzyna Gdowska et al[11] defined elements characterizing the acceptance probability (that still continues to have some randomness): 1- there is no guarantee that an OD will accept the task proposed; 2- it is likely that the probability of accepting a task increases with the amount offered; 3- the optimum amount that should be offered for a given task depends on the costs incurred by the professional feeds, and hence depends also on its current solution. These aspects lead to specific pricing and waging strategies in crowdsource-based business models.

Given that these elements have been widely studied and forecasting methods based on seasonality (i.e. using ARIMA models or similar) are commonly commercially used, we assumed the acceptance probability as given in our case study.

It is common, while making an online order, to be allowed to choose among different time slots for receiving availability: increasing the closeness to real problems, a Time Window (TW) constraint was added to the studied scenario in 2000[12], since when different solutions have been proposed. Time windows can be considered as a constraint or as a cost (if violated, through a Lagrangian relaxation). These different considerations have been analyzed by Alejandro

Fernandez Gill, Maria Gomez Sanchez, Eduardo Lalla-Ruiz and Carlos Castrol[13]: in their case study, the cost computed through a matheuristic method derived by GRASP led to a percentage gap of 4.525 for the biggest analyzed instance. While considering 'soft' TW (that can be violated on the upper bound), the main difficulties are the choice of the parameters that determine the cost increase due to the related violation, as it strictly depends on the customer and on the specific type of service.

The introduction of TW constraint defines our study case: VRPTW (Vehicle Routing Problem with Time Windows) with crowdsourcing.

A relevant optimization technique that was considered to solve the VRPTW problem is the one known as Genetic Algorithm (GA). A genetic algorithm is a type of optimization technique that is inspired by the process of natural selection. The basic idea is to use a population of potential solutions to a problem, and apply evolutionary operators (such as selection, crossover, and mutation) to generate new solutions. The goal is to find the best solution to the problem by iteratively improving the population over time. The process starts by initializing a population of solutions, which can be randomly generated or based on some heuristics. Each solution is represented as a string of bits or genes, which correspond to the parameters of the problem. These genes can be thought of as the "DNA" of the solution. The selection operator is used to select the best solutions from the current population, based on some fitness function that measures the quality of the solution. These solutions are then used to generate new solutions through the crossover operator, which creates new solutions by combining the genes of two parent solutions. The mutation operator is then applied to introduce small random changes to the genes of the new solutions. The new solutions are then added to the population, and the process is repeated for a fixed number of generations or until a satisfactory solution is found. The best solution found so far is typically the one that has the highest fitness value. This approach as well as the Simulated Annealing optimization technique mentioned previously, are used to attempt the improvement of our problem solution.

5 Our Solution

In order to obtain a feasible solution for the VRPTW with crowdshipping optimization problem, it is necessary to firstly compute and evaluate the amount of deliveries that will be assigned to the crowdsource, hence the so-called occasional drivers. These deliveries are then fed into the algorithm which evaluates the actual routes performed by the vehicles going out from the depot. During this step, an optimization algorithm is implemented in order to improve the solution evaluated for the VRPTW problem.

5.1 Crowdshipping

To decide which deliveries to assign to crowdship, we exploit the Python library `DBSCAN`[16], computing a clusterization of all the deliveries. Before the definition of the used clusterization, we exploit a preliminary method to be run before the execution of the algorithm on the actual data: we start with a small cluster radius (`eps`), doubling it at each iteration until reaching a value that lead to a sufficient number of clustered points, to avoid both the extreme situations: almost all nodes are outliers or almost all nodes are in the same cluster. After having thus computed the right `eps` value, we perform the final clusterization. We use it to assign to each point a distance not in absolute terms, but related to the presence of other close points. Before the computation of the 'cluster distances' (the metric we use to decide whether to assign or not its points to crowdship), clusters are re-defined according to time windows: if points belonging to the same cluster have completely separated time windows, we create sub-clusters with them. Our decision metric shown in formula 13: we consider the distance of the more distant point in the cluster (that resulted more efficient than the cluster mass center) and the average failure probability of the points present in the cluster. Through these formulas we obtain a value to be compared to a threshold in order to decide if to assign or not the specific point to the crowdship (if clsuter distance $>$ threshold).

$$d_{cluster} = \frac{dist_{max}}{cluster_{card}} \cdot (1 - p_{failure}) \tag{13}$$

$$d_{singlepoint} = dist_{point} \cdot (1 - p_{failure}) \tag{14}$$

5.2 Routing Algorithm

In this section a broad explanation of the algorithms implemented for optimizing the VRPTW problem is provided.

5.2.1 Insertion Heuristic

The first algorithm we implemented is an insertion type heuristic, in which deliveries are iteratively inserted in a vehicle route without violating capacity and time window constraints. This approach is more simplistic than the second one and, for this reason, a significant improvement in terms of optimization is not expected.

In order to describe this simple algorithm, a pseudo-code is provided below.

Algorithm 1 Route Algorithm

```
sort deliveries according to their time windows
assign the first delivery to the vehicle with the highest capacity
for each vehicle which already contains a point in the route do[2]
    among the non-assigned nodes with a time window compatible with the one in each vehicle[3], choose the delivery with the highest relative distance value15
    update the vehicle time window
    if already built routes are no more expandable[4] then
        the delivery is assigned to a new vehicle which starts its route from the depot
    end if
    if all deliveries have been assigned then
        return the solution
    end if
end for
eventually reassign entire routes to smaller vehicles if capacity is sufficient
```

[2] *At the first iteration we already have the first delivery assigned to a vehicle.*

[3] *The vehicle's time window considers the last inserted vehicle's time window. Compatibility is verified taking the minimum time window of the last point (very close to the moment at which the vehicle arrives at that point) and the minimum time window of the last point +2, to have a relaxed chronological order.*

[4] *The last time slot is reached or vehicle capacity is finished.*

In order to evaluate the relative cost between the start of a new route from the depot and the insertion of the point in the current route, the following formula is used ($relative\ distance\ value$):

$$rdw(B) = distance(depot, B) - distance(A, B) \tag{15}$$

5.2.2 Simulated Annealing and Genetic Algorithm based heuristics

The main idea behind the second algorithm we developed is to create subsets according to the time window constraints related to the points to be reached with the vehicles. Of course, these subsets are filled with the points themselves. The proper vehicle (or more than one vehicle) is chosen for each subset and then are ordered according to the volume, knowing that a smaller volume means a smaller cost. If a vehicles with a smaller volume is not appropriate for a specific subset, a bigger one is selected. There is the possibility that the bigger vehicle is not enough. In this case, the vehicle

is assigned to the route and a second mean is provided to reach the volume capacity needed for the given subset. For each chosen vehicle the time window is saved. Once we provide vehicles to a subset we repeat the procedure for the remaining subsets choosing from the "free" vehicles left, starting from the smaller in volume as before. It must be highlighted that the subsets are ordered according to their time windows as if a vehicle is shared between two subsets it must firstly provide service to the one that has the earliest time span. Finally, for each of the subsets created we compute the best route using either a simulated annealing or genetic algorithm through the aid of the `mlrose` library [15]. The final step consists in assigning the subsets routes to the same vehicle, in a chronological order (considering the volume for each vehicle).

Algorithm 2 Route Algorithm

for n in $nodes$ **do**
  if $TW1 \approx TW2$ **then** ▷ $TW1$ and $TW2$ represent the time windows for two generic nodes
    add n to the proper $subset$
  end if
end for
for each $subset$ **do**
  order $vehicles$ from $lower$ to $higher$ volume
end for
for each $vehicle$ **do**
  save TW
end for
order $subsets$ according to TW
for each $subset$ **do**
  compute best route through $simulated\ annealing$
end for

6 Results discussion

In this section, we show the results obtained from the implementation of the two algorithms previously introduced on the provided dataset composed of 100 points.

The time required to compute the solution is referred to the algorithm execution on a Dell Inspiron 5570 (Intel® Core™ i5-8250U Processor, 16 GB of RAM and no GPU).

6.1 Insertion Heuristic

For the insertion heuristic algorithm we implemented, the result obtained was the following:

- Objective Function = 502.28;
- Time to solution = 0.048 s.

6.2 Simulated Annealing and Genetic Algorithm

In tab. 1 the results obtained from the SA and GA algorithms are shown. As it can be observed, independently from the hyperparameters setting, the results are more or less the same, without changing significantly or not changing at all. We suppose that the cause of this behavior is related to the number of samples each subset is composed of. These algorithms are applied to the single subsets and due to the fact that the number of points for each of them is quite small, these algorithm do not allow high results variability with respects to the used parameters. Our idea is that by working on a larger population of points, both the SA and the GA would perform better.

Algorithm	Obj. Value	Time (s)	Mutation Probability	Iterations	Attempts
GA	392.34	1.94	0.2	/	1
GA	392.34	1.94	0.2	/	10
GA	391.87	11.13	0.6	/	10
GA	391.87	1.77	0.6	/	1
SA	388.78	290.09	/	150000	1
SA	388.78	0.52	/	150	1

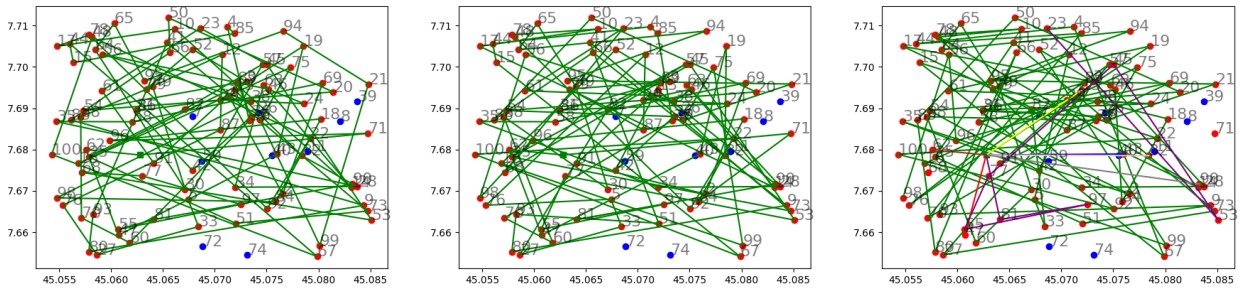
Table 1: SA and GA results.

6.3 Results comparison

As it as been shown, we got quite different results from the two implementations. In the insertion heuristic algorithm the value of the objective function is quite significant compared to the results obtained through the SA and GA approaches. This difference could be explained by the fact that the insertion heuristic is a less elaborated algorithm and it is not capable of exploring more efficient solutions that algorithms such as the SA can offer. Even if the SA approach is not very efficient given to the smallness of the subsets on which it is applied, it shows how an exploration of potential better solutions benefits the value of the objective function which is significantly lower than the one obtained with the insertion heuristic. The best result obtained from the SA/GA approach is through the simulated annealing algorithm with 1 attempt and 150 iterations, giving an objective function equal to 388.78 in only 0.52 seconds, thus we believe it could be used in case of larger datasets and small computation capabilities.

Moreover, as we can deduce from the representation of the routes in fig. 2, the insertion heuristic fails in optimizing the number of used vehicles, increasing the overall cost, while the other method keeps this number at its minimum.

On the other hand, despite the insertion heuristic gives poor results it is quite fast taking around 0.048 s to reach the solution.



(a) Genetic algorithm (b) Simulated Annealing (c) Insertion heuristic

Figure 2: Generated routes. *Red points: deliveries, blue points: crowdship delivereis, lines: vehicles routes.*

References

[1] E. Fadda, G. Perboli, R. Tadei, "A progressive hedging method for the optimization of social engagement and opportunistic IoT problems", 2019

[2] Dantzig GB and Ramser JH; "The truck dispatching problem", 1959

[3] https://www2.isye.gatech.edu/~mgoetsch/cali/VEHICLE/TSP/TSP007__.HTM

[4] M. Fisher and R. Jaikumar; “A Generalized Assignment Heuristic for Vehicle Routing”, 1981

[5] Metropolis, Nicholas; Rosenbluth, Arianna W.; Rosenbluth, Marshall N.; Teller, Augusta H.; Teller, Edward; "Equation of State Calculations by Fast Computing Machines". The Journal of Chemical Physics, 1953

[6] M Pincus; "A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems", 1970

[7] F. Glover; "Future paths for integer programming and links to artificial intelligence", 1986

[8] Dorabela Gamboa, Cesar Rego, Fred Glover; "Data structures and ejection chains for solving", 2004 large-scale traveling salesman problems

[9] Claudia Archetti, Martin Savelsbergh, M. Grazia Speranza; "The Vehicle Routing Problem with Occasional Drivers", 2016

[10] Iman Dayarian, Martin Savelsbergh; "Crowdshipping and Same-day Delivery: Employing In-store Customers to Deliver Online Orders", 2020

[11] Katarzyna Gdowska, Ana Viana, João Pedro Pedroso; "Stochastic last-mile delivery with crowdshipping", 2018

[12] J.-F. Cordeau, Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon, Francois Soumis; "The VRP with Time Windows", 2000

[13] Alejandro Fernandez Gill, Mariam Gomez Sanchez, Eduardo Lalla-Ruiz and Carlos Castrol; "Cumulative VRP with Time Windows: A Trade-Off Analysis", 2020

[14] Christian Prins; "Efficient Heuristics for the Heterogeneous Fleet Multitrip VRP with Application to a Large-Scale Real Case", 2001

[15] https://pypi.org/project/mlrose/

[16] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html