Packing and Routing Approaches in Multi-package Delivery Problem

Bin Yao
binyao@umich.edu

December 21, 2020

Abstract

As delivery services provided by companies like Amazon become popular, models and algorithms for the multi-package delivery problem have increasing importance as customers can order several items that may even exceed the vehicle capacity. The problem formulation allows to split the orders from the same customer and dispatch these orders to different vehicles, with the expectation for a more compact packing solution. In this report, we propose two general approaches. The first one views packing and routing procedures as independent processes, and emphasizing the packing process to optimize the fleet size. The second one combines the packing and routing process for more comprehensive consideration. We use the column generation method incorporated with an exact algorithm for the *elementary shortest path problem with resource constraint* (ESPRRC) to iteratively generate routes. The experiments show that the column generation method produces the best results with the cost of time complexity.

keywords: Multi-package delivery; Column generation; ESPPRC

1 Introduction

The *vehicle routing problem* (VRP), proposed by Dantzig and Ramser in 1959 [7], is concerned with optimizing a set of routes all beginning and ending at a given node called the depot, to serve a given set of customers. Despite its elegant and concise formulation, variants and methodologies are evolved and complicated since its introduction, and this area still thriving with many new approaches in the last decade. All these exciting and informative results not only contribute to the development of combinatorial optimization but also have a strong connection with the industry such as logistics. It is thrilling to see how logistics become efficient with the help of models and algorithms for VRP. In this project, we consider a special variant of VRP where each customer can order multiple orders with different sizes. This scenario is common as delivery services offered by many companies (e.g., Amazon) become popular. We are interested in formulating different approaches to model this scenario and propose efficient algorithms to solve it.

To present many complicated real-world contexts, additional constraints are incorporated into the original VRP model and thus generate many variants. For example, *VRP with time windows*

(VRPTW) [3, 4] that associates a time window for each customer that vehicles can arrive, and *capaciated VRP* (CVRP) that restrict the capacity of vehicles, to name a few. For the multi-package delivery context we consider, there are several variants that we think might share some similarity or may be useful for references. The first one is called the *split delivery VRP* (SDVRP) introduced by Dror and Trudeau [12, 13], where each customer can order things that exceed the capacity of vehicles as each customer can be served by more than one vehicle. The motivation is that savings can be generated by allowing split deliveries, similar to that of multi-package deliveries. [1] provides a survey for this topic. The second one is called the *Vehicle Routing and Loading Problem* (VRLP) [2, 16, 18] that combines vehicle routing and three-dimensional loading. [2] explores different formulations of VRLP and considers under what conditions should we prefer combining the packing and routing together instead of solving them separately. The paper presents a pack-first route-second scheme for the problem. Other relevant topics include the *multi-depot VRP* (MDVRP) [19], the *VRP with a heterogeneous fleet of vehicles* (VRPHE) [21] and the *multi-trip VRP* (MTVRP) [23].

Another contributor to the richness of theories for VRP comes from the methodologies. The algorithms for solving VRP can be classified into exact, heuristic, and meta-heuristic algorithms. [20, 22] provides a comprehensive survey for the methodologies for solving VRP. In this project, we mainly focus on the exact algorithm, incorporated with some heuristics to speed up the implementation. Common exact algorithms include branch-and-bound, branch-and-cut, and set-covering-based algorithms, etc. [22] Column generation, a classic method for solving the large-scale optimization problem, has successfully applied to many variants of VRP [21, 5, 10, 17]. The algorithm formulates the master problem as a set covering problem, say use a set of feasible routes to cover all the customers and each column of the constraint matrix corresponds to a feasible route. The challenge arises from how to obtain an efficient algorithm for the subproblem, say how to find a profitable feasible route in each iteration. A practical scheme is to formulate it as the *shortest path problem with resource constraints* (SPPRC) which is an extension of the shortest path problem with additional restrictions on the available resources. [9] uses this method to solve VRPTW that the feasible routes extend to non-elementary paths. Though there exist efficient pseudo-polynomial algorithms for SPPRC, the *elementary shortest path problem with resource constraints* (ESPPRC) is proved to be strongly NP-Hard [11]. [15] proposes an exact algorithm for ESPPRC that adapts the definition of labels from [9] and uses dominance rules to improve efficiency. In this project, we will implement this method to examine its running time with different scales of the dataset. Column generation can also incorporate with branch-and-bound to form a method called branch-and-price. [14] provides a comprehensive tutorial for this method.

The rest of the paper is organized as follows. Section 2 provides the assumptions we made for the VRP. Section 3 presents a naive implementation for this problem that separates the packing and routing procedures. Section 4 develops a column generation method together with the algorithm for ESPPRC proposed by [15]. Section 5 presents the experimental results. Finally, we conclude our work and outlook future directions in section 6.

2 Problem statement

We consider the multi-package delivery problem where the main feature is that each customer can order multiple orders while all these orders can be dispatched to different vehicles. This may help to deliver a more compact packing solution and thus reduce the fleet size. We also make the following assumptions:

- All vehicles are identical with the capacity of Q.
- All vehicles and customers have the same time window.
- We consider the single-depot scenario, as the multi-depot problem can be decomposed into the single-depot problem by assigning each customer to the nearest depot.
- Reload is allowed, say the vehicles can travel several trips in a single day.
- Depot has unlimited inventory for all products.
- We consider one-dimensional case for the packing problem. Since only the size of products matters, we do not differentiate different products of the same size. Therefore, we assume the data only contains the size of each order without additional information.
- The objective consists of two parts: optimize the fleet size and the total routing distance. Different methods may emphasize different objectives.

3 Method I: Pack-first, route-second

In this section, we present a fast and naive algorithm that separates the packing and routing procedure apart and we call it *pack-first, route-second*. The logic of this algorithm is to first obtain an optimal packing solution to limit the fleet size while considering the later routing process. The optimal routing solution is computed based on the given packing solution.

3.1 Packing

Without the routing part, the problem naturally becomes the *bin-packing problem*. There exist many efficient heuristic and exact algorithms for this problem. For simplicity, we use *Best-Fit Decreasing* (BFD) to obtain a feasible solution and upper bound on the fleet size. BFD first sorts the items in decreasing order and each time place an item to the most full bin it fits into. Given the proven bound of this heuristic and the scale of the problem we consider, the upper bound provided by this heuristic is tight enough to process the next step.

Traditional bin-packing problem is formulated as follows:

$$\min \quad \sum_{j \in V} y_j \tag{3a}$$

$$s.t. \quad \sum_{i \in O} s_i x_{ij} \leq Q y_j \qquad j \in V \tag{3b}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in O \tag{3c}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i \in O, j \in V \tag{3d}$$

Here O is the set of orders, and V is the set of vehicles given the upper bound provided by BFD. s_i is the size of order i. x_{ij} is binary variables that indicates whether order i is delivered by vehicle j. y_j is binary variables that indicates whether vehicle j is used. (3b) guarantees the capacity constraint. (3c) requires each order is delivered by exactly one vehicle. The objective (3a) minimize the fleet size.

Since the trivial packing algorithm does not consider the location of each order, this may lead to a bad routing solution. Therefore, we take one step further by limiting the number of vehicles served for each customer. The intuition is that by assigning the orders from the same customers to the same vehicle instead of a different vehicle, the total mileage can be reduced. The premise is that the fleet size remains the same. Therefore, we have the following formulation:

$$\min \quad \sum_{c \in C} \sum_{j \in \overline{V}} z_{cj} \tag{3e}$$

$$s.t. \quad \sum_{i \in O_c} x_{ij} \leq |O_c| z_{cj} \qquad \forall c \in C, j \in \overline{V} \tag{3f}$$

$$\sum_{i \in I} s_i x_{ij} \leq Q y_j \qquad j \in \overline{V} \tag{3g}$$

$$\sum_{j \in \overline{V}} x_{ij} = 1 \qquad \forall i \in O \tag{3h}$$

$$x_{ij}, z_{cj} \in \{0, 1\} \qquad \forall i \in O, j \in \overline{V}, c \in C \tag{3i}$$

Here C is the set of customers, and O_c is the set of orders from customer c. $\overline{V}$ is the set of vehicles given the upper bound provided by solving (3a)$\sim$(3d). z_{cj} is binary variables that indicate whether customer c is serviced by vehicle j. (3f) ensures at most I_c number of items is served by each vehicle. The objective (3e) minimize the total number of customer-vehicle service pairs given the pre-determined fleet size.

3.2 Multi-trip routing

After obtaining the packing solution, we decompose the VRP into several TSP for each vehicle. We omit the details of the TSP model here as they are already standard. The final step is to combine these trips to further optimize the fleet size. Since reload is allowed, we can combine several single-trip into one multi-trip that meet the time constraint. Here is the formulation for

this step:

$$\min \quad \sum_{j \in \overline{V}} y_j \tag{3j}$$

$$s.t. \quad \sum_{t \in \Omega} T_t x_{tj} \leq T y_j \qquad j \in \overline{V} \tag{3k}$$

$$\sum_{j \in \overline{V}} x_{tj} = 1 \qquad \forall t \in \Omega \tag{3l}$$

$$x_{tj}, y_j \in \{0, 1\} \qquad \forall t \in \Omega, j \in \overline{V} \tag{3m}$$

Ω is the set of all single-trip. T_t is the time taken by route t and T is the total driving time in a single day. x_{tj} is binary variable indicates whether vehicle j runs route t and y_j is binary variable indicates whether vehicle j is used. (3k) requires to satisfy the time constraint. (3l) ensures each trip is assigned to exactly one vehicle. The objective (3j) minimizes the final fleet size.

This formulation is efficient but naive in the sense that it does not fully explore the location of each order and thus makes the routing solution suboptimal. In the next section, we try to combine the packing and routing together with more emphasis on the routing aspect.

4 Method II: Column generation and ESPPRC

In this section, we present a column-generation-based method. We first transform the original problem into a traditional vehicle routing problem by splitting all customers with multiple orders into several dummy customers with the same address and associates each dummy customer with each order from the customer. For instance, if customer A orders two products with sizes 5 and 7, then we will generate two dummy customers A_1 and A_2 with an order of size 5 and 7 respectively, while sharing the same address as A.

We formulate the master problem as set covering problem:

$$\min \quad \sum_{t \in \Omega} c_t x_t \tag{4a}$$

$$s.t. \quad \sum_{t \in \Omega} a_{it} x_t \geq 1 \qquad i \in O \tag{4b}$$

$$\sum_{j \in \overline{V}} x_t \leq U \tag{4c}$$

$$x_t \in \{0, 1\} \qquad \forall t \in \Omega \tag{4d}$$

where Ω is the set of all routes. For each route $t \in \Omega$, c_t is the total distance in this route. x_t is binary variable indicates whether route t is being used or not. $\boldsymbol{a}_t$ is the column associates with variable x_t and for each order $i \in O$, $x_{it} = 1$ if route t serves order i and equals to 0 otherwise. (4b) requires each order is served by at least one route. (4c) restricts the number of vehicles used where U can be any upper bound obtained by heuristics like the *Clarke–Wright savings heuristic* [6]. The objective (4a) minimizes the total distance.

For every $i \in O$, let p_i be the dual variable associates with constraint in (4b) and q be the dual variable associates with constraint (4c). The subproblem is to find a feasible route that the corresponding columns $\boldsymbol{x}_t$ has negative reduced cost, say

$$c_t - \boldsymbol{p}^T \boldsymbol{a}_t - q < 0 \tag{4e}$$

Our previous experience indicates that directly solving the subproblem by integer programming might not be efficient to deal with instances with even just 15 to 30 customers as the time for solving the subproblem increases dramatically. Therefore, this time we turn to another algorithm for solving the subproblem. One prominent idea is to formulate it as the variant of the shortest path problem. This is intuitive from the observation of (4e) as (1) the first two terms can be viewed as the distance of a route from depot to depot and subtract the profits of all nodes in the route and (2) the goal is to find a route to satisfy (4e). *Elementary shortest path problem with resource constraints* (ESPPRC) is the model specifically solving this type of problem. We first create two dummy depots that act as source s and sink node t. The distance between any two nodes is modified as follows:

$$d'_{ij} = d_{ij} - \frac{p_i}{2} - \frac{p_j}{2} \tag{4f}$$

where the profits associate with depots are 0. Now solving (4e) is equivalent to find the shortest path with modified distance from the source to sink and satisfy additional resource constraints. Suppose there are L resources. For each resource l, it has interval $[a_i^l, b_i^l]$ associates with each node i. When solving the problem with dynamic programming, we will check whether the resources being used falls within the interval. In the VRP context, resources typically refer to capacity and time.

[15] proposed the first exact algorithm for solving ESPPRC and we will modify this method for our problem. This is a label-correcting algorithm that speeds up implementation by improving the definition of labels and use dominance rules to tighten the search space. The definitions of label and dominance rule are presented in definition 1 and 2.

Definition 1. For any node i and a path from s to i, the label associates with node i consists of three parts: (1) $\boldsymbol{R}_i = (R_i^1, \ldots, R_i^L)$ indicates the resources being used so far; (2) C_i is the total travel distance; and (3) $\boldsymbol{s}_i = [s_i^1, s_i^2, \ldots, s_i^n]$ is a list of n binary variables corresponding to the states of all nodes excluding the source node. For each node j, $s_j = 1$ if it is unreachable, say either it is already visited, or due to limitation of resources which means there exists resource l such that $R_i^l + r_{ij}^l > b_j^l$ where r_{ij}^l is the consumption of resource l when travels from i to j.

Definition 2. For any two labels associate with node i $\lambda_1 = (\boldsymbol{R}_1, C_1, \boldsymbol{s}_1)$ and $\lambda_2 = (\boldsymbol{R}_2, C_2, \boldsymbol{s}_2)$. If $R_1^l \leq R_2^l$ for every resource l, $C_1 \leq C_2$ and $s_1^i \leq s_2^i$ for every node i, plus $\lambda_1 \neq \lambda_2$, then λ_1 dominates λ_2.

The intuition of the dominance rule is that if λ_1 uses fewer resources to achieve a smaller objective than λ_2, we can safely discard the latter. With the dominance rule, we can only maintain non-dominated labels in each iteration. We present additional definitions:

- Λ_i: List of all non-dominated labels on node i.
- succ(i): Set of successors of node i.

- Queue: List of nodes waiting to be treated.
- EXTEND(i, j): Function that returns labels resulting from the extension of all label $\lambda \in \Lambda_i$ towards node j. The function examines the feasibility of each extension.
- N_{ij}: Set of newly generated labels from i to j.
- SIFT(Λ_1, Λ_2): Function that examines every pair (λ_1, λ_2) where $\lambda_1 \in \Lambda_1$ and $\lambda_2 \in \Lambda_2$, and return a list of all the non-dominated labels. We assume all labels in Λ_1 and Λ_2 are non-dominated.

The pseudo code is presented in algorithm 1. After obtaining all the columns, we resolve the master problem as integer programming and then obtain the set of all feasible routes.

Algorithm 1 Exact algorithm for ESPPRC

```
INITIALIZATION
Λ_s = {((0, ..., 0), 0, (0, ..., 0))}
for each i ∈ V\{s} do
    Λ_i ← ∅
end for
Queue ← {s}

while Queue ≠ ∅ do
    i ← pop the first node out of Queue
    for each j ∈ succ(i) do
        N_ij ← EXTEND(i, j)
        Λ_j ← SIFT(Λ_j, N_ij)
        if Λ_j has changed then
            if new labels are added to Λ_j then
                Update the state information in R_j of each new label given succ(j)
            end if
            if j ∉ Queue then
                Queue ← Queue ∪ {j}
            end if
        end if
    end for
end while
```

5 Experiments

Since we cannot assess the dataset with the multi-package attribute. The computational study is based on the dataset we randomly generated. We set the vehicle capacity as $Q = 60$ and the size of orders are integers randomly sampled between 5 and 15. The time window for all vehicles and customer is from 7 a.m. to 6 p.m., which means there are 11 hours in a single day. The speed of all vehicles is 40 miles/hour. We generate instances in a 2D plain where depot locates

Table 1: Solutions of the MDP instances by different methods

	P1R2			CWS			CG + ESPPRC		
Instance	V	D	T	V	D	T	V	D	T
MPD1-25	2	818.06	0.05	2	575.42	<0.01	2	474.26	66
MDP2-25	2	795.48	0.05	2	676.33	<0.01	2	558.48	194
MDP3-25	2	580.90	0.07	2	520.27	<0.01	1	253.54	45
MDP1-50	4	1391.64	0.15	3	914.54	<0.01	2	631.96	ABORT
MDP2-50	4	1386.70	0.34	3	1032.04	0.01	2	619.48	ABORT
MDP3-50	3	1082.92	1.75	2	870.15	0.02	2	729.88	ABORT
MDP1-100	6	2468.21	0.47	4	1741.96	0.06	4	1692.30	ABORT
MDP2-100	7	2681.21	0.47	5	1782.80	0.05	5	1782.80	ABORT
MDP3-100	6	2491.20	2.48	5	1824.55	0.06	5	1794.15	ABORT

at $(0, 0)$. The address of all customers are randomly generated within range $[-50, 50] \times [-50, 50]$. We assume the graph is complete, which means every pair of nodes has a direct connection. The distance is measured by Euclidean distance. We consider instances with three different scales. We first generate large instances with 100 orders and 50 customers, and randomly dispatch orders to these customers. Customers with no order will be deleted. The medium and small instance is generated by considering the first 50 and 25 orders in large instances. For each size, we generate three instances and name it by "MPD" plus No. and the number of orders.

For the implementation part of the column generation method, we generate initial routes by the Clarke-Wright savings heuristic. We also incorporate an accelerating strategy that the subproblem is stopped prematurely when $N = 10$ labels are generated with negative reduced cost, though N can be tuned for different scales of instances. For the ESPPRC algorithm, we consider the capacity as the only resource because the time constraint is loose.

We test three algorithms through all instances: pack-first route-second method in section 3 (P1R2), Clarke-Wright savings heuristic (CWS), and column generation method in section 4 (CG+ESPPRC). The performances are evaluated by the fleet size (V), the total distance (D), and running time (T). We abort any run that exceeds 10 minutes (and output the best solution) or generates less than $N = 10$ labels with negative reduced cost in one iteration to achieve faster implementation. Computational experiments were carried out on a PC with a 2.4-GHz processor and 16 GB of RAM. The results are presented in table 1.

It is no surprise to see that even the Clarke-Wright savings heuristic outperforms P1R2 in every criterion as we make weak and naive analysis about the routing process. Therefore, every single trip obtained by P1R2 is far from the optimal solution and this becomes the disadvantage when combining routes to form multi-trip routes. The results show that under P1R2 the fleet size is more sensitive to the scale of instances. While CWS delivers a similar fleet size solution as CG+ESPPRC.

For the CG+ESPPRC method, the algorithm can achieve the modified stopping criterion quickly for small instances. The results indicate that the improvement percentage decreases within the same time slot as the scale of the instance increases. This is especially obvious if we compare the results of CWS and CG+ESPPRC for large instances. Overall, this suggests that CG+ESPPRC is workable

Table 2: Results with different parameters for instance MDP1-25

Q \ T	10	11	12
40	(2,562)	(2,562)	(2,562)
60	(2,474)	(2,474)	(1,474)
80	(2,506)	(2,506)	(2,506)

for solving the subproblem and generating feasible routes with negative reduced cost efficiently, though more specific accelerating strategies are required for better implementation.

As suggested by [15], the time complexity of the exact algorithm of ESPPRC highly relies on the structure of the graph, the numbering of the nodes, and the tightness of resource constraints. The complete structure of the graph might be one of the main reasons that impact the time complexity, as all instances used in [15] are relatively sparse. Therefore, this algorithm is less versatile to handle instances with unexpected structure. However, it is pretty scalable to extend to other variants of VRP. For example, for VRPTW, we just need to add time into the resource list when solving the subproblem.

We also use the MDP1-25 instance to test different vehicle capacity Q and length of time window T. We record the fleet size and total routing distance (V, D) for each pair of parameters and present in table 2. Our implementation is sensitive to vehicle capacity as it affects the initial solution generated by the Clarke-Wright savings heuristic. For a fixed capacity or a fixed initial solution, the solution is pretty robust regarding different time parameters.

6 Conclusions and future works

We present two methodologies for solving the multi-package delivery problem. The first one is called pack-first route-second that emphasizes the packing part while delivers weak analysis for the routing solution. The second one is based on column generation and formulate the subproblem as a variant of the shortest path problem, then use an exact algorithm for ESPPRC to solve the subproblem. This algorithm delivers more reasonable solutions as we take packing and routing with equal consideration. Yet, there are still many interesting directions worth attempting but cannot be done given the time limitation, we put it here as future works.

- Though the pack-first route-second delivers bad results. The idea of separating the packing and routing procedure is still fascinating. The key part is how to incorporate the consideration for routing into the packing procedure. We may refer to relevant papers for better understanding.
- The implementation results of the column generation method suggests that running time increases drastically versus the scale of instances. Therefore, accelerating strategies are important for implementation in practice. There are rich results in how to speed up column generation implementation, e.g. [8] provides a survey for the accelerating strategies in col-

umn generation methods for VRP. Techniques like column elimination that discards a set of non-basic columns according to their marginal cost might be one direction to speed up the implementation.

- There is another popular method called branch-and-price that combines branch-and-bound and column generation. There are many branching schemes, e.g. branch the routes or arcs [14]. This is another strategy for accelerating the pure column generation method.
- Although we did not explore the multi-depot case, the works in this project can naturally extend to similar models. One easy adaption is to decompose the multi-depot instance into several single-depot instances by assigning each customer to the nearest depot. After routing for each depot, we can then apply some improvement heuristics like exchange heuristics, tabu search, etc. to search for better solutions.

References

[1] The split delivery vehicle routing problem: a survey. In C. Archetti and M. G. Speranza, editors, *The vehicle routing problem: Latest advances and new challenges*, pages 103–122. Springer, Bonston, MA, 2008.

[2] J. H. Andreas Bortfeldta. Packing first, routing second—a heuristic for the vehicle routing and loading problem. *Operations Research*, 40(3):873–885, 2013.

[3] O. Braysy and M. Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005.

[4] O. Braysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[5] E. Choi and D.-W. Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers Operations Research*, 34(7):2080–2095, 2007.

[6] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[7] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[8] G. Desaulniers, J. Desrosiersa, and M. M. Solomon. *Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems.* Springer US, New York, 2002.

[9] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[10] J. Desrosiers, F. Sournis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.

[11] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–978, 1994.

[12] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, 1989.

[13] M. Dror and P. Trudeau. Split delivery routing. *Naval Research Logistics*, 37(3):383–402, 1990.

[14] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. 8(4):407–424, 2010.

[15] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks: An International Journal*, 44(3):216–229, 2004.

[16] M. Gendreau, M. I. G. Laporte, and S. Martello. A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350, 2004.

[17] M. Jin, K. Liu, and B. Eksioglu. A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters*, 36(2):265–270, 2008.

[18] A. Moura and J. F. Oliveira. An integrated approach to vehicle routing and container loading problems. *OR spectrum*, 31(4):775–800, 2009.

[19] K. Sašo and V. Podgorelec. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27:519–532, 2015.

[20] L. V. Snyder and Z.-J. M. Shen. *Fundamentals of Supply Chain Theory*. John Wiley Sons, Incorporated, 2019.

[21] E. D. Taillard. A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO-Operations Research-Recherche Opérationnelle*, 33(1):1–14, 1999.

[22] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.

[23] A. Şen and K. Bülbül. A survey on multi-trip vehicle routing problem. pages 401–407, 2008.