An adaptive large neighborhood search for the multi-vehicle profitable tour problem with flexible compartments and mandatory customers

Vincent F. Yu [a,b] , Nabila Yuraisyah Salsabila [a,*] , Aldy Gunawan [c] , Anggun Nurfitriani Handoko [a]
[a] Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan
[b] Department of Civil and Environmental Engineering, Portland State University, Portland, OR 97201, USA
[c] School of Computing and Information Systems, Singapore Management University, Singapore

Published in Applied Soft Computing (2024) 156. DOI: 10.1016/j.asoc.2024.111482

Abstract

The home-refill delivery system is a business model that addresses the concerns of plastic waste and its impact on the environment. It allows customers to pick up their household goods at their doorsteps and refill them into their own containers. However, the difficulty in accessing customers' locations and product consolidations are undeniable challenges. To overcome these issues, we introduce a new variant of the Profitable Tour Problem, named the multi-vehicle profitable tour problem with flexible compartments and mandatory customers (MVPTPFC-MC). The objective is to maximize the difference between the total collected profit and the traveling cost. We model the proposed problem as Mixed Integer Linear Programming and present an Adaptive Large Neighborhood Search (ALNS) algorithm to solve it. Our ALNS outperforms the commercial solver, Gurobi, and Large Neighborhood Search (LNS), as proven by giving better solutions within reasonable computational times. Both ALNS and LNS can obtain optimal solutions for all small instances and three better solutions than Gurobi for medium problems. Furthermore, ALNS is also robust and effective in solving large MVPTPFC-MC, as proven by resulting in better solutions within less CPU time than LNS. Finally, more analyses are conducted to justify the utilization of flexible compartment sizes by comparing it with fixed compartment sizes and to evaluate the robustness of MVPTPFC-MC. The results show that utilizing flexible compartment sizes can yield more benefits than fixed compartment sizes, particularly when the fleet size is limited, and there are fewer mandatory customers to serve.

Keywords: Flexible compartment, Home-refill, Mandatory customer, Multi-compartment, Multi-trip

1. Introduction

The Home-Refill Delivery System (HRDS) is a grocery delivery service that brings customers' orders right to their doorstep. The HRDS business model is based on refilling household goods in bulk, which means that customers can either use their own containers or reusable containers provided by the company. This business model was inspired by the 18th-century milk delivery system, where milkmen would deliver milk from farms to customers' homes in bulk containers, and the customers would use their own containers to store the milk. HRDS aims to reduce packaging waste, promote sustainable practices, and address the issue of plastic waste.

According to data from the United Nations Environment Program in 2021, 400 million tons of plastic waste are disposed of every year. Out of this, 36% comes from product packaging, including single-use food and beverage containers [1]. This issue is driving society to increase environmental awareness, leading to the growing popularity of sustainable living. HRDS offers the circular economy concept that involves closed-loop material life cycles in the fast-moving consumer goods (FMCG) sector [2]. Using reusable packages with this system can reduce 90% of plastic packaging and decrease carbon footprints [3]. It is also one of the emerging last-mile delivery systems that can also be integrated with e-commerce or e-grocery [2]. Therefore, it is important to manage HRDS optimally, as it aims to improve profitability and eliminate waste through the closed-loop supply chain, supporting economic and environmental sustainability [4], [5].

This business model faces multiple challenges due to the fact that customers are located in different areas and are often difficult to access with larger vehicles. As a solution, many companies choose to use motorbikes, which have limited capacity. This leads to an increased need for more vehicles in order to meet the demand of all customers. The task of consolidating customer orders is also a challenge since household goods demand is often made up of various product types and small quantities. As a result, the distribution operation must be

* Corresponding author. E-mail address: vincent@mail.ntust.edu.tw.

managed optimally to avoid ineffective consolidation and loading of products into vehicles, which can lead to inefficient vehicle usage. To sum up, the key challenges of this delivery system are (1) limited vehicle capacity due to the use of small vehicles, (2) the need to accommodate various product types during delivery, and (3) the fact that each customer may order small quantities of multiple product types at the same time.

To increase the number of customers served without expanding the fleet, companies can use multi-trip deliveries [6]. Multi-Trip Vehicle Routing Problem (MTVRP) allows vehicles to visit the depot multiple times in a day [7]. Additionally, companies can increase profits by serving customers who order outside the designated order period, but this is not mandatory. Customers can be divided into two types based on their demand fulfillment requirements: mandatory and optional. Mandatory customers place their orders during the designated order period, and their demand should be fulfilled in a single visit. Optional customers place their orders outside the ordering period, and their demand does not have to be fulfilled. In this case, the Profitable Tour Problem (PTP) can be implemented when resources are insufficient to serve many customers. In PTP, each customer has a predetermined profit that affects their attractiveness. PTP determines the customers to be served and their visiting sequence while maximizing the difference between collected profit and travel cost [8,9]. In a specific case, PTP may consider mandatory customers, which is a subset of customers that must be served [10].

Multi-compartment vehicles (MCVs) can be used to deliver various types of products efficiently [11]. In the vehicle routing problem with multiple compartments (MCVRP), vehicle route planning also deals with compartment-related constraints, such as flexibility of the compartment, and delivery-related constraints, such as options for consolidating different customer demands on the same delivery route [12]. In order to maximize vehicle space utilization and serve more customers on every trip, the company can use flexible compartment sizes when distributing various product types, especially in HRDS, which can lead to increased profits. On the contrary, using fixed compartment sizes (FCS) may restrict the amount of products to be delivered on each trip, leading to fewer customers being served. Additionally, HRDS usually utilizes a set of different compartment sizes, which can be easily installed on the vehicle whenever the courier loads the products at the depot. Therefore, the configuration of compartment sizes on each trip must be determined in such a way that vehicle capacity is not exceeded. This variant of flexible compartment sizes is known as flexible discrete compartment sizes (FDCS) as introduced by Henke et al. [13].

We present a new variant of the Multi-vehicles Profitable Tour Problem (MVPTP) that includes three features: multi-trip, multi-compartment, and mandatory customers. Although MTVRP and MCVRP have been extensively studied in the literature, studies addressing the multi-trip and multi-compartment characteristics of PTP are scarce. Moreover, the classical PTP does not consider mandatory visits, which is considered in our study. We formulate the problem as the multi-vehicle profitable tour problem with flexible compartments and mandatory customers (MVPTPFC-MC), adopted from HRDS. MVPTPFC-MC is a mixed integer programming (MILP) that aims to determine (1) a set of optional customers to be visited in each trip, (2) a set of vehicle routes in each trip - i.e., the sequences of the visited customers, (3) the assignment of customers' orders to the vehicle compartments, which is associated with the product type, and (4) a set of compartment types stored in each vehicle. The objective function is to maximize the difference between the collected profit and traveling cost.

PTP belongs to the class of routing problem that is NP-hard [9]. Hence, it is hard to solve the problem within reasonable computing time, especially for large-scale problems. Metaheuristic has been known as a powerful computational method to solve large-scale NP-hard logistics problems [4]. One of the metaheuristic methods is the Adaptive Large Neighborhood Search (ALNS), which is known for its flexibility to be implemented on various problems. It enables designers to embed various well-performing heuristics as destroy or repair operators since the ALNS framework has a domain-free structure [14]. Unlike the preceding method (i.e., Large Neighborhood Search), the neighborhood selection is conducted on an adaptive mechanism, which enables effective neighborhoods to be more likely to be selected, because the probability is adjusted dynamically according to their performance [15,16]. Another advantage of ALNS is its ability to embed multiple insertions, which improves the diversification of its searching process and helps avoid local optima [14]. However, since ALNS involves various neighborhoods, it requires several parameters to function effectively. ALNS has been used effectively in solving MVPTP, MCVRP, and MTVRP problems. Examples of studies that have used ALNS include Wang et al. [17], who developed ALNS for a split-delivery MCVRP with multi-trips for the fuel replenishment problem, and Azi et al. [18], who used ALNS to solve MVPTP with multi-trips. Both studies showed that ALNS can obtain high-quality solutions within a reasonable computational time.

To sum up, the contributions of this study are listed as follows.

1. We propose a new variant of MVPTP, named the multi-vehicle profitable tour problem with flexible compartments and mandatory customers (MVPTPFC-MC), which is adopted from an emerging business model, the home-refill delivery system. A new Mixed Integer Linear Programming is presented to formulate the proposed MVPTPFC-MC.
2. An Adaptive Large Neighborhood Search is developed to solve MVPTPFC-MC. Some modifications on the destroy and repair operators are made to adapt the characteristics of MVPTPFC-MC, including the consideration of two customer types and compartment assignment.
3. A computational experiment is conducted to test the validity, efficiency, and robustness of ALNS to newly generated test instances. This includes a comparative study with a commercial solver, GUROBI, and the preceding method of ALNS, Large Neighborhood Search.
4. Further analysis is conducted to justify the utilization of flexible compartment sizes compared to fixed compartment sizes in several performance matrices. The robustness of MVPTPFC-MC under different mandatory percentage and fleet size scenarios is also evaluated through a sensitivity analysis.

The remaining sections of this study are as follows. Section 2 presents the related literature. Section 3 gives the problem description and formulation. Section 4 offers the solution methodology. Section 5 shows the numerical experiment. Finally, Section 6 concludes this study and suggests future research directions.

2. Literature review

In this section we discuss some studies related to MVPTPFC-MC, including PTP, MCVRP, and MTVRP, as well as some ALNS implementations for these problems.

2.1. Profitable tour problem

PTP generally operates a single vehicle and does not consider maximum trip duration and mandatory visits. In the case of PTP with multiple vehicles (MVPTP), Daniel Handoko et al. [19] developed a PTP-based Winner Determination Problem (WDP) for the Urban Consolidation Center (UCC). They formulated this problem as a bi-level model, which consists of a knapsack problem and VRP at the upper and lower levels, respectively. To solve the problem, they proposed the concept of knowledge adoption to efficiently use evolutionary bi-level programming. Furthermore, Gansterer et al. [20] developed a model for the pickup and delivery problem (MVPPDP). In this problem, multiple vehicles transport goods from pickup to delivery customers within a specified trip duration. Two variants of General Variable Neighborhood Search (GVNS) are proposed to solve the problem, namely Sequential

GVNS and Self-adaptive GVNS. The experimental results demonstrated that both proposed methods outperformed Guided Local Search (GLS) in terms of solution quality for solving all test instance sizes. However, GLS is found to be more computationally efficient for medium and large instances. Alhujaylan and Manar [21] constructed a Greedy Randomized Adaptive Search Procedure (GRASP) to solve the same problem. They compared their proposed GRASP with two greedy construction heuristics previously used in the literature. The proposed GRASP algorithm effectively solved MVPPDP by finding eight new best solutions and eight better solutions than both greedy construction heuristics.

PTP with mandatory visits were studied in Cortés-Murcia et al. [22] and Cortés-Murcia et al. [10]. Cortés-Murcia et al. [22] developed the capacitated profitable tour problem with mandatory stops for electric vehicles (EV). Here, the mandatory stops are the charging stations for EVs. They developed a branch-and-price algorithm as the solution method, which was able to find optimal solutions for 120 instances. Additionally, the proposed algorithm was able to solve instances with up to 100 customers and 13 mandatory stops in less than 7 min on average. Cortés-Murcia et al. [10] set up the same problem with multiple periods. They solved the problem by constructing a branch-and-price algorithm, which can obtain high-quality solutions for instances with up to 100 clients and 3 periods in a reasonable time.

2.2. *Multi-compartment vehicle routing problem*

Numerous industrial implementations of MCVRP have been studied in the literature, such as in fuel distribution [1,23–25], waste collection [13,26–28], and agricultural industry [29,30]. To the best of our knowledge, there are currently no studies on implementing MCVRP for home-refill delivery. According to Ostermeier et al. [12], special attributes are usually defined in MCVRP: (1) flexibility of compartment sizes, (2) assignment of product types to compartments, (3) shareability of compartments, (4) total number of visits per customer, and (5) the mode of demand fulfillment.

Only a few studies developed MCVRP with flexible compartment sizes. Henke et al. [13,27] implemented discrete compartment sizes (MCVRP-FDCS) for glass waste collection from customer locations. According to their experiment, the exact approach can only solve problem instances with limited sizes within a reasonable computation time. Hence, they proposed a variable neighborhood search (VNS) algorithm to solve large problem instances, which performs well in solving small problem instances. For large instances, the heuristic produces good-quality solutions within reasonable times. Henke et al. [27] also solved a similar problem with a modified model, which resulted in significantly improved computing times. They proposed a branch-and-cut algorithm to solve this problem to optimality. The experimental result showed that the algorithm can solve instances with up to 50 locations to optimality and reduces the computing time by 87% percent compared to Henke et al. [13].

Heßler [31] developed an exact algorithm for MCVRP with flexible compartment sizes. They considered two variants of flexible compartment sizes: discrete compartment sizes and continuous compartment sizes. They developed three branch-and-cut algorithms based on three-index formulation, two-index formulation, and route-indexed formulation. They conducted an extensive computational experiment to compare the performance of the algorithms in solving both variants. The branch-and-cut algorithm with a three-index formulation performs best for solving small instances with a low number of nodes. The other two algorithms perform best for solving medium instances with a low number of vehicles. Within two hours, the algorithm can optimally solve up to 50 nodes for the continuous compartment size variant. For the discrete compartment size variant, the algorithm can optimally solve 16 medium and 2 large instances.

To our knowledge, only Lahyani et al. [32] has incorporated PTP with multi-compartment vehicles. They developed a rich variant of capacitated PTP with fixed compartment sizes. In their problem, some products are incompatible and must be kept separated during transportation. Some products and some compartments also have some incompatibilities. Furthermore, customer demand may be satisfied partially by delivering only some of the products and must not be split. To solve the problem, they proposed a Variable Neighborhood Search algorithm that provides equal or close to optimal solutions for small instances.

2.3. *Vehicle routing problem with multi-trip*

In MTVRP, each vehicle is allowed to visit the depot multiple times a day [33]. This makes MTVRP very practical in our problem since MVPTPFC-MC has short-distance networks and limited vehicle capacity. Chbichib et al. [34] incorporated PTP with multiple trips and proposed a Profitable VRP with Multiple Trips (PVRPMT). Similar to our problem, their objective is to maximize the difference between the total collected profit and the transportation costs. Each vehicle is allowed to perform several trips within a workday time limit. They proposed two greedy constructive heuristics to solve the problem. Based on their computational results, both constructive heuristics produced suboptimal solutions for small instances and found feasible solutions for large instances that CPLEX cannot obtain.

The proposed problem in Azi et al. [18] is also similar to the characteristics of the PTP with multiple trips. Each customer is associated with a gain value (i.e., profit), and visiting each customer is not mandatory. Furthermore, each vehicle can perform multiple routes during its operation day. The objective is to maximize the total gain collected and minimize the total distance traveled. They developed an Adaptive Neighborhood Search (ALNS) by employing various operators in all hierarchical (i.e., multilevel) structures of the problem. The result shows that this approach leads to much better solutions than the classical customer-based approach of ALNS.

Wang et al. [17] developed a split-delivery multi-compartment vehicle routing problem with multiple trips for the fuel replenishment problem. The problem determines the allocation of oil products to vehicle compartments, the delivery routes, and the delivery patterns of each vehicle such that the total makespan and traveled time are minimized. They also constructed ALNS and tested it on data from a Chinese petroleum transportation company. The computational results showed that ALNS can solve instances with up to 60 customers and 3 different products in less than 25 min with an average optimality gap of around 10%. ALNS also finds optimal solutions in solving small instances with significantly shorter time than the exact approach.

2.4. *Adaptive large neighborhood search*

ALNS has produced high-quality solutions in many MVPTP studies. Li et al. [35] developed ALNS for the pickup and delivery problem with time windows, profits, and reserved requests. In addition to ad-hoc destroy and repair operators, they embedded six moves in a local search procedure at the end of each segment. The proposed ALNS is then tested to new instances generated based on benchmark instances from Ropke and Pisinger [36]. Their proposed ALNS can solve the problem to optimality for instances with 10 requests and better objective values for instances with 20 requests with less CPU time than CPLEX. For larger problems, ALNS significantly outperforms CPLEX in terms of solution quality and CPU time.

ALNS also has been constructed for solving time-dependent profitable pickup and delivery problem with time windows by Sun et al. [37]. They conducted an extensive computational study that showed that ALNS can find high-quality solutions quickly on instances with up to 75 requests. ALNS is able to compete with the dynamic programming heuristics of Sun et al. [38] in solving single-vehicle instances. The average gap is less than 0.60% and 2.00% for medium and large instances, respectively, and the CPU time is significantly less than the dynamic programming heuristic. For solving multi-vehicle instances, they compared their approach with an exact branch-and-price algorithm proposed by Sun et al. [39]. The proposed ALNS is able to find better solutions within a significantly shorter CPU time.

An extension of ALNS has been developed by Chentli et al. [40] to solve the profitable tour problem with simultaneous pickup and delivery. They developed a selective ALNS (sALNS) where two operator selection mechanisms, random selection and score-dependent selection, are used. According to their experiment, sALNS provides the best results among ALNS and CPLEX for instances with 50–199 customers within the shortest CPU time. sALNS provides better results than CPLEX in 97 cases and identical results in 17 cases.

Alinaghian and Shokouhi [41], Chen et al. [42], Mofid-Nakhaee and Barzinpour [43], and Eshtehadi et al. [44] implemented ALNS and its extension to MCVRP. Alinaghian and Shokouhi [41] hybridized ALNS with VNS (HALNS) to solve multi-depot MCVRP. Compared with exact, ALNS, and VNS algorithms, HALNS provides the best and optimal solutions for solving small instances within a significantly short time. Furthermore, HALNS also gives a 0.18% gap for the large-scale problem, whereas VNS and ALNS are 2.4% and 2.02%. However, ALNS excels in solving both small and large instances within the shortest CPU times.

ALNS for MCVRP in cold-chain food distribution was studied in Chen et al. [42]. ALNS is compared with CPLEX and a practical decision procedure (2SP) to newly generated instances based on historical data. ALNS can solve optimal solutions for small instances in much less CPU time than CPLEX and provides better solutions than 2SP. For solving medium and large instances, they modified ALNS by embedding 2SP as the initial solution (ALNS-M). The result showed that both ALNSs result in better solutions than 2SP, where ALNS is slightly better than ALNS-M in terms of the solution quality and CPU time.

Eshtehadi et al. [44] developed an ALNS in solving MCVRP for city logistics. They employed the Clark and Wright (C&W) algorithm for initiating temperature. The results highlighted that small instances have small gaps between CPLEX and ALNS, where the maximum value is 0.06%.

In solving MTVRP, ALNS and its extensions have been developed by Grangier et al. [45], François et al. [46], François et al. [47], Wenli Li and Li [48], and Pan et al. [49]. Grangier et al. [45] constructed ALNS for solving the two-echelon MTVRP with satellite synchronization. They followed Azi et al. [18] by using three levels of destroy and repair operators: workday, route, and customer. The computational results showed that the proposed ALNS can find good solutions in a reasonable time.

François et al. [46] constructed two adaptive large neighborhood search algorithms for MTVRP: ALNSM and ALNSP. The former includes multi-trip operators in heuristics that iteratively destroy and repair the relaxed MTVRP solutions, and the latter employs a bin-packing approach for solving CVRP and then subsequently assigns the trips. Their numerical experiments showed that both solution approaches produced competitive results with state-of-the-art results, whereas the proposed ALNSP algorithm produced new best solutions. Furthermore, François et al. [47] offered similar solution approaches for MTVRP with time windows. The computational results indicated that ALNSM outperforms ALNSP in the presence of time windows, because the trip assignment is more challenging in constrained problems.

Pan et al. [49] developed a hybrid metaheuristic (ALNS-VND) of ALNS and variable neighborhood descent (VND) to solve multi-trip time-dependent VRP with time windows. For the small problem instances, ALNS-VND outperforms CPLEX by obtaining optimal and better solutions for all instances in short-run times. Furthermore, ALNS-VND is also robust in solving large problem instances.

Table 1 summarizes the related research of this study. Few studies have proposed either MVPTP with multi-compartments or MVPTP with multi-trips. However, only our study considered MVPTP with both multiple compartments and trips. Furthermore, HRDS has not been extensively investigated. Qualitative research on the adoption intentions of HRDS for FMCG has only been taken up by Yu et al. [2] so far. However, there is no quantitative research on HRDS, especially on VRP. Our current study fills the gap by incorporating three problems, MTVRP, MCVRP, and MVPTP, adopted from HRDS. Mandatory visits are also considered in our study as it has not been extensively studied in MVPTP. We introduce HRDS where multi-compartment vehicles are adopted with the following attributes: (1) flexible discrete compartment sizes, (2) flexible compartment assignment, (3) shared compartment, (4) single visit per customer, and (5) unsplit demand fulfillment. We adopt the characteristic mentioned by Henke et al. [27], where the flexible compartment sizes are pre-determined discretely, and the number and the type of compartments loaded on each vehicle are variable. Hence, the decision of the number and the combinations of compartments add complexity to the problem. Furthermore, we also propose ALNS to solve the problem, which has proven its good performance in solving various extensions of MTVRP, MCVRP, and MVPTP.

3. Problem description and formulation

MVPTPFC-MC, which is adopted from the home-refill delivery system, is described as a direct geographical network $G = (V, A)$, where V is the set of nodes, and A is a set of arcs. The node-set V is comprised of two subsets of nodes, $V = \{0\} \cup N$, where node $\{0\}$ represents the location of a single depot, and N represents the set of customers' location. Furthermore, the set of customer locations N is subdivided into two subsets of customer location $N = N_M \cup N_O$: the location of mandatory customers N_M, and the location of the optional customer N_O. A denotes a set of arcs $A = \{(i,j)|i,j \in V, i \neq j\}$ that connects the nodes in V. A set of products P is available at the depot to be distributed to the customer. Each customer $i \in N$ has a non-negative demand of d_{ip} units for each $p \in P$ type of product. Each unit of the demand is associated with a profit equal to 1. The demand of each mandatory customer must be served. For the optional customers, the company is not required to serve the demand for all products. However, if the company decides to fulfill the demand for a certain product, then the full amount of that product demand must be fulfilled.

Let a set of homogeneous vehicles K with capacity Q and a set of compartment types M be available at the depot to perform a set of trips R. In each trip, each vehicle must leave and return to the depot $\{0\}$. For each arc $(i,j) \in A$, c_{ij} represents the travel time from nodes i to j. For each time unit, the travel cost is equal to 1. The service time in customer $i \in N$ is denoted by e_i. The total daily operation time for each vehicle must not exceed the working time $T_{\max}$. For each compartment type $m \in M$, the capacity is denoted as q_m. Since this model adapts the concept of FDCS, the type and number of the compartment loaded are not specified in advance, but must be determined individually for each vehicle and trip. The size of each compartment type can vary discretely based on the multiple of the basic compartment unit size q_{unit}. For instance, if the vehicle capacity Q is 1,000 units and the basic compartment unit size q_{unit} is 10 units, then the possible compartment size for this instance is a multiple of 10 units (e.g.: 10, 20, 30, up to 1000 units). To mimic the real condition, we assume that there are three types of compartments $|M| = 3$, small, medium, and large with $q_m = \{10, 20, 50\}$, respectively. We assume that there will always be available compartments for every size whenever the courier loads the products into the vehicle. In practice, the home-refill company usually uses reusable containers that can be easily refilled for the next trip. The total capacity of the loaded compartments must certainly not exceed the vehicle capacity Q.

The examples of FCS and the proposed FDCS model are illustrated in Figs. 1(a) and 1(b), respectively. They illustrate the delivery of a vehicle for serving six customers: 4 mandatory and 2 optional customers. Customers 1, 2, 3, and 4 are the mandatory customers, whereas customers 5 and 6 are the optional customers. Each customer has a non-negative

Table 1
Related literature.

Author	MV -PTP[1]	Compartment Attributes[2]					Multi -trips	Industrial applications[3]	Solution methodology[4]
		FlexS	FlexA	SC	SV	UD			
[34]	✓						✓	General	2-GC
[24]			✓			✓		Fuel	EA
[32]	✓				✓	✓		General	VNS
[18]	✓						✓	Perishable	ALNS
[29]			✓	✓	✓	✓		Agricultural	EA
[19]	✓							UCC	KA
[13]		✓	✓	✓		✓		Waste	VNS
[25]			✓	✓	✓	✓		Fuel	EA
[46]							✓	General	ALNS
[20]	✓							General	GVNS
[21]	✓							General	GRASP
[22]	M							EV	EA
[27]		✓	✓	✓		✓		Waste	EA
[47]							✓	General	ALNS
[28]			✓	✓		✓		Waste	MTH
[17]			✓	✓			✓	Fuel	ALNS
[49]							✓	General	ALNS & VNS
[31]		✓	✓	✓	✓	✓		General	EA
[10]	M							EV	EA
Current research	M	✓	✓	✓	✓	✓	✓	FMCG	ALNS

[1] M: mandatory visits.
[2] FlexS: flexible size; FlexA: flexible compartment assignment; SC: shared compartment; SV: single visit; UD: unsplit demand.
[3] FMCG: fast-moving consumer goods; UCC: urban consolidation center; EV: electric vehicles.
[4] 2-GC: two greedy constructive heuristics; EA: exact algorithm; VNS: variable neighborhood search; ALNS: adaptive large neighborhood search; KA: knowledge adoption; GVNS: general variable neighborhood search; GRASP: greedy randomized adaptive search procedure; MTH: matheuristic.

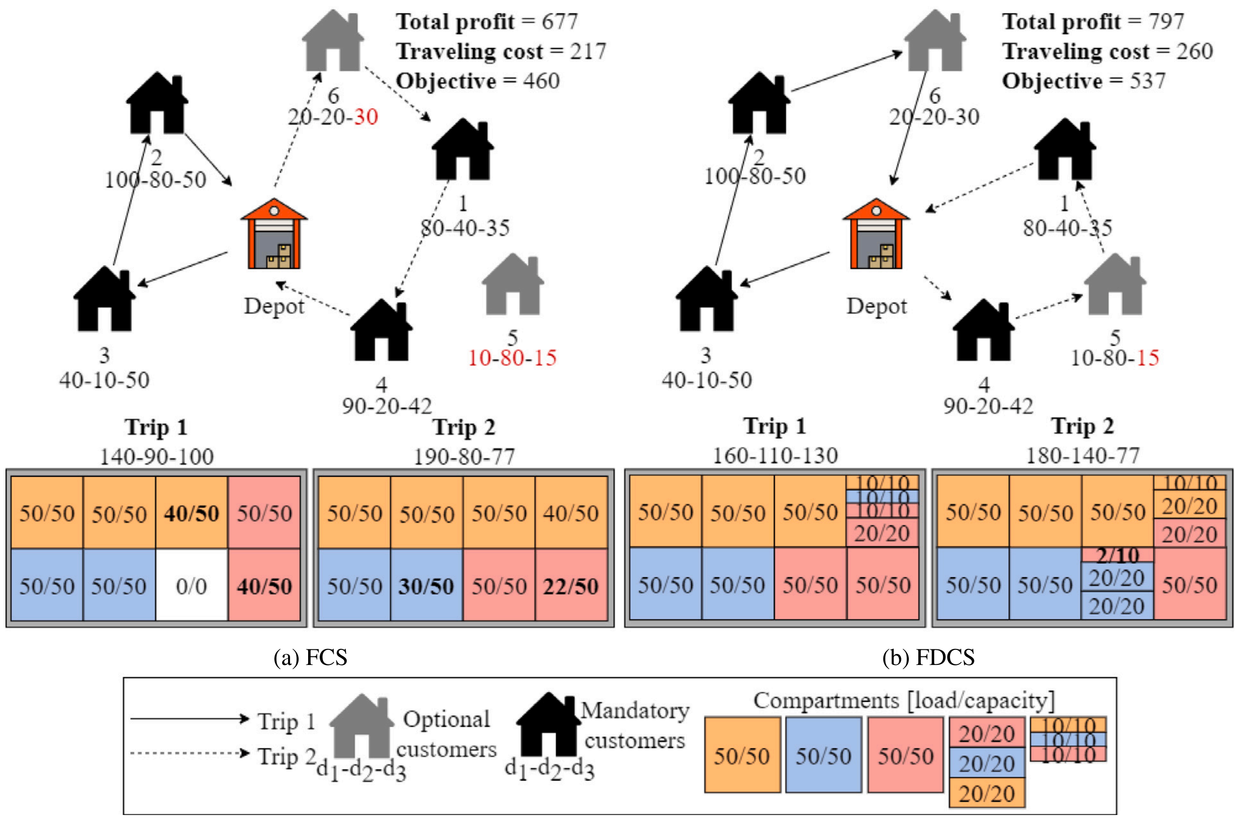


Fig. 1. Illustration of MVPTP with multiple compartments.

demand for three products. Let the values d_1, d_2, and d_3 below each node be the demand of each product. In this example, both settings have the same size of available vehicle ($|K| = 1$) and trips ($|R| = 2$). Furthermore, each trip has the same capacity ($Q = 400$). The difference between these two settings is the flexibility of the compartment size. In FCS, the number of compartments is fixed at equal to 8, and each capacity of the compartment is fixed at $q = 50$. In contrast, the number of compartments and their capacity are not determined in advance in FDCS. However, as aforementioned, the total size of the loaded compartments must not exceed the vehicle capacity. In each figure, the top illustrates the vehicle route, and the bottom illustrates the compartment assignment of each product in each trip. The assignment of compartments to each product is color-coded as follows: orange is for product 1, blue is for product 2, and red is for product 3.

Under the FCS setting, the company does not serve all of customer 5's products and one product of customer 6's. In comparison, all customers are visited under FDCS, resulting in a higher objective value. In the first setting, although there are still multiple idle compartments (i.e., 1 compartment is not used, and 1 compartment of products 1 and 2 is not fully loaded), the optimal decision is to not serve customer 5. It may be feasible to load products 1 and 3 of customer 5's demand, as well as product 3 of customer 6's demand, by examining the idle compartments. However, serving these customers is less profitable than the optimal solution due to additional travel costs. This issue can be tackled by utilizing FDCS, where the compartment configurations can be adjusted according to customer demands. As shown in Fig. 1, utilizing FDCS instead of FCS results in greater optimal compartment utilization. Although more travel distance is required, the objective value is higher in FDCS, because more profitable customers can be served. This can be achieved although one product of customer 5 not being delivered due to limited vehicle capacity.

MVPTPFC-MC is formulated as a Mixed Integer Linear Programming (MILP) model that optimizes the following decisions simultaneously.

1. x_{ij}^{kr} A binary variable that represents the vehicle route, which equals 1 if arc $(i,j) \in A$ is traversed by vehicle $k \in K$ on trip $r \in R$.
2. u_{ip}^{kr} A binary variable that represents the customer's product type that is served by the company, which equals 1 if vehicle $k \in K$ delivers the product type $p \in P$ of customer $i \in N_O$ on trip $r \in R$.
3. w_{pm}^{kr} An integer variable that represents the number of compartment $m \in M$ used to load product $p \in P$ on vehicle $k \in K$ on trip $r \in R$.
4. s_i Dummy decision variable for the subtour elimination constraint indicating the tour position where node $i \in V$ is visited.

The objective function of MVPTPFC-MC is to maximize the difference between the total collected profit and traveling cost.

Eqs. (1)–(14) are the mathematical model formulation of MVPTPFC-MC. The objective function (1) is to maximize the difference between the collected profit and the traveling cost. The collected profit is calculated from the total served demand. Constraint (2) ensures that all product demands of each mandatory customer must be fulfilled. The customer must be visited on a trip if at least one of the products is served by the corresponding vehicle as formulated in constraint (3). Constraint (4) imposes that each customer is only visited by a single vehicle. In addition, each vehicle can only visit each customer once, as formulated in Constraint (5). Constraint (6) is the flow continuity constraint that ensures each vehicle must depart from the current location whenever it visits any other nodes. Constraint (7) is the subtour elimination constraint. Constraint (8) ensures the demand for each product type does not exceed the corresponding compartment capacity. Constraints (9) and (10) ensure that the loaded products and the corresponding compartments do not exceed the vehicle capacity. Constraint (11) ensures that each vehicle's total travel time and service time must not exceed the daily operation time. Finally, constraints (12)–(15) are the domain of the decision variables.

Objective function:

$$\max \sum_{k\in K}\sum_{r\in R}\sum_{p\in P}\sum_{i\in N} u_{ip}^{kr} d_{ip} - \sum_{k\in K}\sum_{r\in R}\sum_{i\in V}\sum_{j\in V} c_{ij} x_{ij}^{kr} \tag{1}$$

Subject to:

$$\sum_{k\in K}\sum_{r\in R} u_{ip}^{kr} = 1 \qquad \forall i \in N_M, p \in P \tag{2}$$

$$\sum_{\substack{i\in V\\ i\neq j}} x_{ij}^{kr} \geq u_{jp}^{kr} \qquad \forall j \in N, k \in K, p \in P, r \in R \tag{3}$$

$$\sum_{\substack{j\in V\\ j\neq i}}\sum_{k\in K}\sum_{r\in R} x_{ij}^{kr} \leq 1 \qquad \forall i \in N \tag{4}$$

$$\sum_{\substack{j\in V\\ j\neq i}} x_{ij}^{kr} \leq 1 \qquad \forall i \in V, k \in K, r \in R \tag{5}$$

$$\sum_{\substack{j\in V\\ j\neq i}} x_{ij}^{kr} = \sum_{\substack{j\in V\\ j\neq i}} x_{ji}^{kr} \qquad \forall i \in N, k \in K, r \in R \tag{6}$$

$$s_i - s_j + x_{ij}^{kr}|V| \leq |V| - 1 \qquad \forall i,j \in N, k \in K, r \in R, i \neq j \tag{7}$$

$$\sum_{i\in N} d_{ip} u_{ip}^{kr} \leq \sum_{m\in M} q_m w_{pm}^{kr} \qquad \forall k \in K, p \in P, r \in R \tag{8}$$

$$\sum_{i\in N}\sum_{p\in P} d_{ip} u_{ip}^{kr} \leq Q \qquad \forall k \in K, r \in R, m \in M \tag{9}$$

$$\sum_{p\in P}\sum_{m\in M} q_m w_{pm}^{kr} \leq Q \qquad \forall k \in K, r \in R \tag{10}$$

$$\sum_{(i,j)\in A}\sum_{r\in R} (c_{ij} + e_i) x_{ij}^{kr} \leq T_{\max} \qquad \forall k \in K \tag{11}$$

$$x_{ij}^{kr} \in \{0,1\} \qquad \forall (i,j) \in A, k \in K, r \in R \tag{12}$$

$$u_{ip}^{kr} \in \{0,1\} \qquad \forall i \in N, p \in P, k \in K, r \in R \tag{13}$$

$$w_{pm}^{kr} \in \mathbb{Z}^+ \qquad \forall p \in P, m \in M, k \in K, r \in R \tag{14}$$

$$s_i \in \mathbb{Z}^+ \qquad \forall i \in V \tag{15}$$

4. Adaptive large neighborhood search for MVPTPFC-MC

We propose an Adaptive Large Neighborhood Search (ALNS) algorithm that is specifically constructed for MVPTPFC-MC inspired by the ALNS framework developed by Ropke and Pisinger [36]. ALNS integrates multiple destroy and repair operators to be executed along the searching process. The destroy operators remove some customers from the route, whereas the repair operators re-insert those customers. Each operator's selection probability is adjusted iteratively based on its performance. In contrast with the classical ALNS, our proposed framework considers the following modifications as follows.

1. In addition to the vehicle route, the proposed framework also determines the list of unvisited customers containing the optional customers that are not visited in a corresponding solution.
2. Two types of customers are considered (i.e., mandatory and optional customers). Hence, some adjustments are made to the destroy and repair operators according to each customer's type as follows.
    (a) In some destroy operators, the customers are not only removed from the vehicle route, but also from the list of unvisited customers.
    (b) In the repair operators, all mandatory customers must be inserted before the optional customers to avoid infeasibility. Furthermore, to insert an optional customer into the route, the repair operator must also decide the types of products to be served.
3. After inserting all customers into the route, the number and types of compartments must be assigned to each trip.
4. To enhance the solution exploration, the proposed ALNS incorporates the acceptance mechanism of the Simulated Annealing (SA).
5. The proposed ALNS also considers the number of non-improvements as one of the terminating conditions.

Our proposed ALNS considers thirteen parameters: initial temperature rate T_0^θ, final temperature T_f, maximum iteration MaxIter, the rate of maximum number of non-improvements $\eta_{\max}^\theta$, cooling rate α, the rate of iterations per segment N_{ALNS}^θ, random removal factor p, reaction factor ω, the range of the number of removed customers $q_{\min}$

Algorithm 1: Adaptive Large Neighborhood Search for MVPTPFC-MC

```
    Input: An MVPTPFC-MC instance, ALNS parameters
    Output: Best solution and best objective
 1  function ALNS(MVPTPFC-MC Instance)
 2      $s \leftarrow$ an initial solution generated by $initialSolutionGeneration(N)$;
 3      initialize solutions $s' \leftarrow s; s_{\text{best}} \leftarrow s$;
 4      initialize parameters $T \leftarrow T_0; It \leftarrow 0; \eta_0 \leftarrow 0$;
 5      while $It < MaxIter$ and $T > T_f$ and $\eta_0 < \eta_{max}$ do
 6          if mod $(It, N_{ALNS}) = 0$ then
 7              for $j \in Y \cup Z$ do
 8                  $p_j \leftarrow$ the updated probability according to $sc_j$;
 9                  reset score $sc_j = 0$;
10              end
11          end
12          let $s' \leftarrow s$;
13          $R_y \leftarrow$ a destroy operator selected by Roulette wheel from $Y$;
14          $I_z \leftarrow$ a repair operator selected by Roulette wheel from $Z$;
15          generate the number of removed customers $q \sim U\left[q_{min}, q_{max}\right]$;
16          destroy solution $\left[s', DestroySet_M, DestroySet_O\right] \leftarrow R_y(s', UnvisitedSet, q)$;
17          repair solution $s' \leftarrow I_z(s', DestroySet_M, DestroySet_O)$;
18          update the number of non-improvements $\eta_0 = \eta_0 + 1$;
19          if $f(s') > f(s_{best})$ then
20              update solution $s \leftarrow s'; s_{\text{best}} \leftarrow s'$;
21              update score $sc_y = sc_y + \delta_1; sc_z = sc_z + \delta_1$;
22              reset the number non-improvements $\eta_0 = 0$;
23          else if $f(s') > f(s)$ then
24              update solution $s \leftarrow s'$;
25              update score $sc_y = sc_y + \delta_2; sc_z = sc_z + \delta_2$;
26          else
27              $r \leftarrow U(0, 1)$;
28              if $r < \exp(\Delta/T)$ then
29                  update solution $s \leftarrow s'$;
30                  update score $sc_y = sc_y + \delta_3; sc_z = sc_z + \delta_3$;
31              end
32          end
33          update iteration $It = It + 1$;
34          update temperature $T = \alpha T$;
35      end
36      return $s_{best}, f(s_{best})$
37  end
```

and $q_{\max}$, and the score rates δ_1, δ_2, and δ_3. Each destroy $y \in Y$ and repair $z \in Z$ operator function are denoted as $R_y()$ and $I_z()$, where Y and Z are the sets of destroy and repair operators, respectively. Each operator $j \in Y \cup Z$ is associated with its attributes: weight w_j, score sc_j, probability p_j, and the number used θ_j. Each weight w_j^k and probability p_j are initialized with equal values, and each score sc_j is initialized with zero. Three types of solutions are generated throughout the computational process: current solution s, temporary solution s', and best solution s_{best}. Each of its objective function is denoted as $f(s), f(s')$, and $f(s_{\text{best}})$, respectively, referring to Eq. (1). Let q and $UnvisitedSet$ be the number of removed customers and the list of optional customers that are not visited in the corresponding solution, respectively. Furthermore, $DestroySet_M$ and $DestroySet_O$ are the list of removed mandatory and optional customers, respectively. The pseudocode of ALNS is detailed in Algorithm 1, which is described line-by-line as follows.

Beginning from Line 1 and ending on Line 37, the input of the proposed ALNS is an MVPTPFC-MC instance, and the outcomes are the best solution and its objective value. In Line 2, the initial solution s is generated based on Algorithm 2. The best solution s_{best} and the temporary solution s' are initialized in Line 3. Furthermore, the current temperature T, the current iteration It, and the number of non-improvements η_0 are initialized in Line 4. Let T_0 be the initial temperature controlled by the initial temperature rate T_0^θ. We follow the formulation from Ropke and Pisinger [36] to determine the initial temperature as follows: $T_0 = f(s_0) - T_0^\theta f(s_0)$. Here, $f(s_0)$ is the objective value of the initial solution. The iteration starts from Line 5 until Line 35.

Each operator's probability p_j and operator's score sc_j are updated and reset at every N_{ALNS} iteration where $j \in Y \cup Z$ (see: Line 8). N_{ALNS} is controlled by parameters N_{ALNS}^θ and formulated as follows: $N_{\text{ALNS}} = N_{\text{ALNS}}^\theta$MaxIter. Each operator's weight w_j and probability p_j are updated according to Eqs. (16) and (17), respectively. Here, w'_j denotes the previous weight of the corresponding operator j.

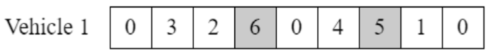
$$w_j = w'_j(1-\omega) + \omega \frac{sc_j}{\theta_j} \qquad \forall j \in Y \cup Z \tag{16}$$

$$p_j = \frac{w_j}{\sum_{i \in Y \vee Z} w_i} \qquad \forall j \in Y \cup Z \tag{17}$$

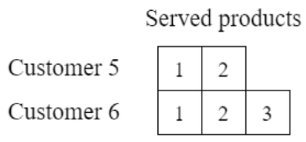
In each iteration, we let the temporary solution s' be the current solution s. In Lines 13 and 14, one destroy R_y operator and one repair I_z operator are selected randomly by the Roulette wheel selection mechanism. Furthermore, the number of removed customers q is generated randomly in Line 15 according to the minimum and maximum numbers of removed customers $\left[q_{\min}, q_{\max}\right]$. The destroy and repair operators are then employed in Lines 16 and 17. After a new temporary solution s' is obtained, the number of non-improvements η_0 is then updated.

The acceptance mechanism adopted from SA is employed from Line 19 until Line 32. The temporary solution s' is accepted in three conditions according to its objective value $f(s')$. The first condition is when the temporary solution s' is better than the best solution s_{best} (see: Lines 19–22). In this condition the temporary solution s' is updated as the current s and the best s_{best} solutions. Furthermore, the score of the selected destroy R_y and repair I_z operators are incremented by δ_1, and the number of non-improvements η_0 is reset as 0. The second condition is when the temporary solution s' is better than the current solution s. The temporary solution s' is then updated as the current solution s, and the scores of the selected operators are then incremented by δ_2. Finally, the third condition is when the temporary solution s' is worse than the current solution s, but can be accepted according to the Boltzmann function. Let Δ be the temporary solution s' subtracted by the current solution s. The temporary solution s' is updated as the current solution s if a random number is less than $\exp(\Delta/T)$. Furthermore, the selected operators' scores are also incremented by δ_3.

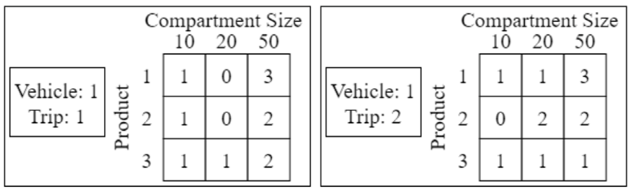
After employing the acceptance mechanism, the number of iterations It is incremented (Line 33), and the temperature T is updated (Line 34). Finally, the convergence criteria are evaluated. Let $\eta_{\max}$ be the maximum number of non-improvements, controlled by $\eta_{\max}^\theta$. Here, $\eta_{\max}$ is calculated based on the formula of $\eta_{\max}^\theta$MaxIter. The iteration will stop if either one of these three conditions is met: (1) the number of iterations It has reached the maximum iterations MaxIter, (2) the current temperature T has reached the final temperature T_f, and (3) the number of non-improvements η_0 has reached the maximum number of non-improvements $\eta_{\max}$.



(a) The route solution representation for Figure 1b



(b) The solution representation of served products for Figure 1b



(c) The solution representation of compartment arrangement for Figure 1b

Fig. 2. The solution representation for Fig. 1(b).

4.1. Solution representation

The solution structure of ALNS for MVPTPFC-MC is divided into three main parts: the route for each vehicle, the products served to each optional customer, and the compartment arrangement for each vehicle on each trip. Fig. 2 illustrates the solution structures for Fig. 1(b). The route solution is illustrated in Fig. 2(a), the served products of optional customers are illustrated in Fig. 2(b), and the solution of compartment arrangement is illustrated in Fig. 2(c). Each vehicle route is illustrated in an array. Hence, the number of arrays is equal to the number of vehicles $|K|$. Each trip is separated by 0, indicating each vehicle returns to and departs from the depot. The served optional customer is color-coded in gray in each vehicle route.

On the first trip, vehicle 1 visits customers 3, 2, and 6 sequentially. The products served to customer 6 are products 1, 2, and 3. Product 1 is loaded in one compartment of size $q = 10$ and three compartments of size $q = 50$. Product 2 is loaded in one compartment of size $q = 10$ and two compartments of size $q = 50$. Finally, product 3 is loaded in one compartment of size $q = 10$, one compartment of size $q = 20$, and two compartments of size $q = 50$. After that, vehicle 1 returns to the depot and visits customers 4, 5, and 1 on the second trip. Customer 5 is served with products 1 and 2. On this trip, one compartment of size $q = 10$, one compartment of size $q = 20$, and three compartments of size $q = 50$ are used to load product 1. Two compartments of size $q = 20$ and two compartments of size $q = 50$ are used to load product 2. Finally, one compartment with sizes of $q = 10$, $q = 20$, and $q = 50$ is used to load product 3.

4.2. Initial solution generation

The initial solution is generated by a random insertion, as illustrated in Algorithm 2. The input is the set of customers N, and the outputs are the initial solution s and the list of unvisited customers $UnvisitedSet$. In the initial solution, only the mandatory customers are inserted into the route, whereas the optional customers will be inserted into the route throughout the improvement stage. Based on our observation, we find that including only the mandatory customers during the initialization phase can improve the algorithm's efficiency. Adding some optional customers to certain positions in the route may not lead to better solutions, which would require further destroy and insertion phases. These additional iterations can negatively impact the computational efficiency of the algorithm.

To begin with, the $UnvisitedSet$ needs to be initialized. This involves adding each mandatory customer to $UnvisitedSet$. The insertion mechanism (Lines 3–13) is described as follows. First, let i be a customer selected randomly from $UnvisitedSet$ (Line 5). Second, let pos be a position selected randomly from the solution s' (Line 6). Third, the customer i is then inserted into the corresponding position pos (Line 7). The solution's feasibility is then evaluated in Line 8. If the solution is feasible, then the temporary solution s' is updated as the current solution s, and customer i is removed from $UnvisitedSet$. The insertion continues until all customers in $UnvisitedSet$ are inserted into the route.

After all mandatory customers are inserted into the route, the compartments of loaded products in the solution s are then assigned to each trip (Line 14). Finally, all optional customers N_O are then inserted into $UnvisitedSet$, which is later removed and inserted in the destroy and repair operators.

Algorithm 2: Initial solution generation for MVPTPFC-MC

```
    Input: The set of customers
    Output: Initial Solution, the list of unvisited customers
 1  function initialSolutionGeneration($N$)
 2      inserting all mandatory customers $N_M$ into
          $UnvisitedSet$;
 3      while $UnvisitedSet$ is not empty do
 4          let $s' \leftarrow s$;
 5          $i \leftarrow$ a random node in $UnvisitedSet$;
 6          $pos \leftarrow$ a random position in $s'$;
 7          insert $i$ into $s'[pos]$;
 8          Feasibility $\leftarrow FeasibilityCheck(s'[pos])$;
 9          if $Feasibility = TRUE$ then
10              update solution $s \leftarrow s'$;
11              remove $i$ from $UnvisitedSet$;
12          end
13      end
14      assign compartments for $s$;
15      inserting all optional customers $N_O$ into $UnvisitedSet$;
16      return $s, UnvisitedSet$
17  end
```

4.3. Destroy operators

Fig. 3 illustrates an example of the removal procedure in general. Let the figure on the left-hand side be the original route before removal and the figure on the right-hand side be the route after removal. The removed customers are color-coded in red. In this example, two customers are removed from the solution s', and one customer is removed from $UnvisitedSet$. These customers are then inserted into the list of removed customers according to their type (i.e., $DestroySet_M$ and $DestroySet_O$). In our study, four destroy operators $|Y| = 4$ are developed: random removal, distance-based Shaw removal, profit-distance-based Shaw removal, and worst removal. The following sections describe the difference between these destroy operators in terms of the selection of the removed customers.

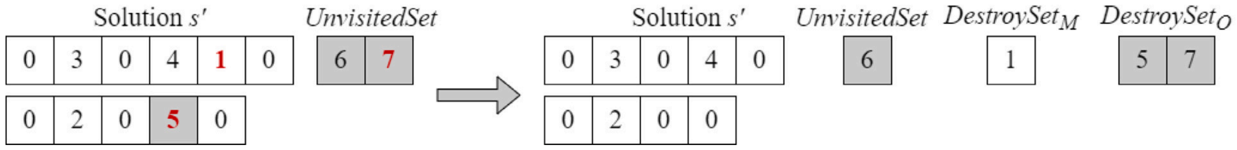


Fig. 3. Removal procedure.

4.3.1. Random removal

The idea of the random removal heuristic is randomly removing customers from current solutions Ropke and Pisinger [36]. This is one of the most straightforward heuristics that tends to generate perturbed solutions; as a result, it diversifies the search process [50]. The pseudocode for the random removal is illustrated in Algorithm 3. The input of this operator is the temporary solution s', the list of unvisited customers $UnvisitedSet$, and the number of removed customers q. The output of this operator is the temporary solution s' and the list of removed nodes $DestroySet_M$ and $DestroySet_O$.

Algorithm 3: Random removal operator for MVPTPFC-MC

```
      Input: Temporary solution, the list of unvisited customers,
             number of removed nodes
      Output: Temporary solution, the list of removed nodes
 1    function randomRemoval($s'$, $UnvisitedSet$, $q$)
 2        while $|DestroySet_M| + |DestroySet_O| < q$ do
 3            $t \leftarrow U[0,1]$;
 4            if $t = 1$ then
 5                $i \leftarrow$ a random customer in $s'$;
 6                remove $i$ from $s'$;
 7            else
 8                $i \leftarrow$ a random customer in $UnvisitedSet$;
 9                remove $i$ from $UnvisitedSet$;
10            end
11            if $i$ is a mandatory customer then
12                insert $i$ into $DestroySet_M$;
13            else
14                insert $i$ into $DestroySet_O$;
15            end
16        end
17        return $s'$, $DestroySet_M$, $DestroySet_O$
18    end
```

Let t be a binary number determining the type of removal, which is generated randomly in Line 3. If t equals 1, then a random customer i is removed from the route s' (Lines 5–6); otherwise, a random customer i is removed from $UnvisitedSet$ (Lines 8–9). After removal, customer i is then inserted into the list of removed nodes according to its type (Lines 11–15). The removal continues until the number of removed customers has reached q.

4.3.2. Distance-based shaw removal and profit distance-based shaw removal

We adopt the Shaw removal as introduced in Shaw [51]. In this operator, the node with high similarities will be removed and relocated to a more profitable position. In contrast with the classical Shaw removal, our study measures two types of relatedness, the distance (SD) and the profit distance (SP), which are formulated in Eqs. (18) and (19), respectively. The generic procedure of the Shaw removal is described in Algorithm 4.

$$SD(i,j) = c_{i,j} \qquad \forall i,j \in N \tag{18}$$

$$SP(i,j) = |d_i - c_{0,i}| - |d_j - c_{0,j}| \qquad \forall i,j \in N \tag{19}$$

The input of the Shaw removal algorithm covers the temporary solution s', the list of unvisited customers $UnvisitedSet$, and the number of removed nodes q. The output is the temporary solution s' and the list of removed nodes $DestroySet_M$ and $DestroySet_O$. The procedure starts by generating a seed by selecting a random mandatory customer i from the route s' (Lines 2–3). Another customer is then removed one by one inside the loop 4–29.

To remove a node, first, a customer d is randomly selected from the $DestroySet_M$. Second, a removal type t is decided by generating a random binary variable. If it is equal to 1, then a customer is removed from the route s' (Lines 7–14); otherwise, a customer is removed from the unvisited customers' list $UnvisitedSet$ (Lines 15–23). The procedure to remove a customer from the route s' is detailed as follows. Let $sortedList$ be the list of customers that are sorted in ascending order according to their relatedness value. For each node j in the route s', the relatedness measure is calculated and then sorted into $sortedList$ (Lines 8–11). Let m be a random value generated between 0 and 1, which is used for selecting customer i to be removed from $sortedList$. Line 13 employs the random selection, where p is a random removal factor that is previously mentioned as a parameter. It is important to note that this selection mechanism favors higher-positioned customers for removal (i.e., customers with higher similarity). The procedure to remove a customer from $UnvisitedSet$ is similar (Lines 15–23). However, we sort each customer j in $UnvisitedSet$ instead of the route. The selection mechanism is similar to the previous procedure. After removing a customer j, we insert it into the list of removed customers according to its type (Lines 24–28). The removal continues until the removed customers have reached q.

4.3.3. Probabilistic worst removal

Probabilistic worst removal is adopted from [36], which is removing a customer from the worst position. Here, the worst position means the position that is less beneficial for the corresponding node. In our study we calculate the benefit of not serving a customer as the difference between the total profit of serving that customer and the total profit of not serving that customer. This benefit is denoted as the removal profit $RP(i)$, which is formulated in (20). Here, i^- and i^+ are the nodes located before and after customer i, respectively.

$$RP(i) = \sum_{p \in P} d_{ip} - \left(c_{i^-,i} + c_{i,i^+} - c_{i^-,i^+}\right) \forall i \in N \tag{20}$$

The algorithm of probabilistic worst removal is illustrated in Algorithm 5. The input of this algorithm is the temporary solution s' and the number of removed nodes q. The output is the temporary solution s' and the list of removed nodes $DestroySet_M$ and $DestroySet_O$. The procedure of removing each node is detailed in Lines 2–10. First, we calculate the removal profit $RP(j)$ of each node j in solution s' (Line 4). Second, each node j is sorted in ascending order according to its removal profit $RP(j)$ into $sortedList$. We employ a similar mechanism as the Shaw removal operator to select the removed customer i as shown in Lines 7–9. It is worth noting that a lower removal profit has a higher chance of being removed. Finally, we insert each removed customer i into list $DestroySet_M$ or $DestroySet_O$ according to its type. The customers continue being removed until the number of removed customers has reached q.

Algorithm 4: Shaw removal for MVPTPFC-MC

```
Input: Temporary solution, the list of unvisited customers,
       number of removed nodes
Output: Temporary solution, the list of removed nodes
1  function shawRemoval($s'$, $UnvisitedSet$, $q$)
2      $i \leftarrow$ a random mandatory customer in $s'$;
3      insert $i$ into $DestroySet_M$;
4      while $|DestroySet_M| + |DestroySet_O| < q$ do
5          $d \leftarrow$ a random node in $DestroySet_M$;
6          $t \leftarrow U[0,1]$;
7          if $t = 1$ then
8              for $j \leftarrow$ each customer in $s'$ do
9                  calculate $SD(d,j)$ or $SP(d,j)$;
10                 sort $j$ in ascending order according to its
                     $SD(d,j)$ or $SP(d,j)$ value into $sortedList$;
11             end
12             $m \sim U(0,1)$;
13             $i \leftarrow sortedList[m^p|sortedList|]$;
14             remove $i$ from $s'$;
15         else
16             for $j \leftarrow$ each customer in $UnvisitedSet$ do
17                 calculate $SD(d,j)$ or $SP(d,j)$;
18                 sort $j$ in ascending order according to its
                     $SD(d,j)$ or $SP(d,j)$ value into $sortedList$;
19             end
20             $m \sim U(0,1)$;
21             $i \leftarrow sortedList[m^p|sortedList|]$;
22             remove $i$ from $UnvisitedSet$;
23         end
24         if $i$ is a mandatory customer then
25             insert $i$ into $DestroySet_M$;
26         else
27             insert $i$ into $DestroySet_O$;
28         end
29     end
30     return $s'$, $DestroySet_M$, $DestroySet_O$
31 end
```

Algorithm 5: Probabilistic worst removal for MVPTPFC-MC

```
Input: Temporary solution, number of removed nodes
Output: Temporary solution, the list of removed nodes
1  function worstRemoval($s'$, $q$)
2      while $|DestroySet_M| + |DestroySet_O| < q$ do
3          for $j \leftarrow$ each customer in $s'$ do
4              calculate $RP(j)$;
5              sort $j$ in ascending order according to its $RP(j)$
                 value into $sortedList$;
6          end
7          $m \sim U(0,1)$;
8          $i \leftarrow sortedList[m^p|sortedList|]$;
9          remove $i$ from $s'$;
10     end
11     if $i$ is a mandatory customer then
12         insert $i$ into $DestroySet_M$;
13     else
14         insert $i$ into $DestroySet_O$;
15     end
16     return $s'$, $DestroySet_M$, $DestroySet_O$
17 end
```

4.4. Repair operators

Fig. 4 illustrates an example of the general procedure for repair operators. Let the figure on the left be the solution that resulted from removal and the figure on the right be the solution after insertion. In this example, customer 1 is inserted into vehicle 2 on trip 2, and customer 7 is inserted into vehicle 1 on trip 2. Customer 5 is inserted into $UnvisitedSet$. It is important to note that both $DestroySet_M$ and $DestroySet_O$ must be emptied in each repair operator.

In the insertion procedure, the following rules must be satisfied in order to obtain a feasible solution.

1. The total load of each trip must not exceed the vehicle capacity.
2. The total working time of each vehicle must not exceed the maximum working time.
3. For each removed customer, all mandatory customers must be inserted before the optional customers.
4. For each mandatory customer, the demand for each product must be served.
5. If an optional customer is not feasible to be inserted into a position, then the optional customer will not be visited (i.e., inserted into $UnvisitedSet$).
6. For each optional customer, the products to be served are randomly generated before selecting a position. If the product is feasibly served, then the product can be served and the corresponding customer is inserted into the route array; otherwise, the corresponding product will not be served. If no products are feasible to be served, then the corresponding customer is not visited (i.e., inserted into $UnvisitedSet$).

Let $FeasibilityCheck(.)$ be a function to check solution feasibility with respect to rules 1 and 2. The general insertion procedure, which is explained in the following sections, will ensure the remaining rules are met. Five repair operators $|Z| = 5$ are developed: adjusted random insertion, first available position insertion, last available insertion position, greedy insertion, and greedy insertion with noise.

4.4.1. Adjusted random insertion

The adjusted random insertion is detailed in Algorithm 6. The input of this algorithm is the temporary solution s' and the list of removed nodes $Destroy_M$ and $Destroy_O$. The output of this algorithm is the temporary solution s'. This insertion procedure is employed in two loops. The first loop is the insertion of the mandatory customers (Lines 2–10). Let i be a customer selected randomly from $DestroySet_M$, and pos is a random position in solution s'. Line 5 is to check whether inserting customer i into position $s'[pos]$ is feasible or not. This includes checking the feasibility of loading all products' demand of customer i (Rule 4). If it is feasible, then the customer i can be inserted into $s'[pos]$ and removed from $DestroySet_M$. This insertion continues until all customers in $Destroy_M$ have been inserted into the route s'. This is associated with rule 3.

The insertion of the optional customers is covered in the second loop (Lines 11–23). The general procedure is similar to the insertion of mandatory customers, except for determining the products to be served (Line 14). We follow rule 6 to determine the products. Additionally, if the optional customer cannot feasibly be inserted into a position, then it will not be served as mentioned in rules 5 and 6.

4.4.2. First available position insertion and last available position insertion

The first available position insertion operator was proposed by Hammami et al. [52], which inserts the nodes into the first feasible position in a route. The process is similar to Algorithm 6, but the first feasible position is selected instead of choosing the position randomly. Similarly, in the last available position insertion, the removed node is inserted into the last feasible position.

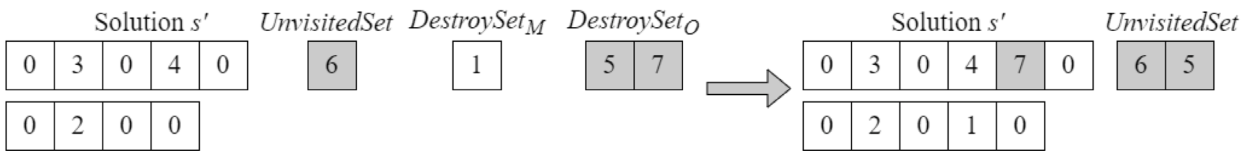


Fig. 4. Insertion procedure.

Algorithm 6: Adjusted random insertion for MVPTPFC-MC

```
Input: Temporary solution, the list of removed nodes
Output: Temporary solution
function randomInsertion($s'$, $DestroySet_M$, $DestroySet_O$)
    while $DestroySet_M$ is not empty do
        $i \leftarrow$ a random customer from $DestroySet_M$;
        $pos \leftarrow$ a random position in $s'$;
        Feasibility $\leftarrow FeasibilityCheck(s'[pos])$;
        if Feasibility = TRUE then
            insert $i$ into $s'[pos]$;
            remove $i$ from $DestroySet_M$;
        end
    end
    while $DestroySet_O$ is not empty do
        $i \leftarrow$ a random customer from $DestroySet_O$;
        $pos \leftarrow$ a random position in $s'$;
        $p \leftarrow$ random products to be loaded into $s'[pos]$;
        Feasibility $\leftarrow FeasibilityCheck(s'[pos])$;
        if Feasibility = TRUE then
            insert $i$ into $s'[pos]$;
        end
        else
            insert $i$ into $UnvisitedSet$;
        end
        remove $i$ from $DestroySet_O$;
    end
    assign compartments for $s'$;
    return $s'$
end
```

4.4.3. *Greedy insertion and greedy insertion with noise*

The greedy insertion heuristic inserts a node into a position that maximizes the objective function. The detailed procedure is similar to Algorithm 6 except for determining the position. In the greedy insertion we try to insert customer i into each feasible position in solution s' and calculate insertion profit $IP(i)$. After each trial, the customer is then inserted into the best position. Note that the highest insertion profit determines the best position. The formulation of the insertion profit is shown in Eq. (21).

$$IP(i) = \sum_{p \in P} d_{ip} - \left(c_{i^-,i} + c_{i,i^+} - c_{i^-,i^+}\right) \forall i \in N \tag{21}$$

Similarly, the greedy insertion with noise also inserts a node according to the insertion profit. However, a noise factor is considered by multiplying the insertion profit by a random number. This process enables the algorithm to not only select the best node but also the second or third best node. The range of noise factor is $[0.7, 1.2]$ as in Hemmelmayr et al. [53].

5. Numerical experiment

This section describes the generation of test instances, parameter tuning, computational results, and sensitivity analysis for MVPTPFC-MC. The proposed ALNS is coded in Microsoft Visual C++ and executed on a computer with Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz and 32 GB of RAM.

5.1. *Test instances*

We generate a set of instances for MVPTPFC-MC based on the MCVRP instances by Henke et al. [13]. Three sizes of instances are generated to test the proposed ALNS. We select 30 small instances with 10 customers, 8 medium instances of 20 customers and 12 large instances with 50, 100, 200, 300, and 400 customers for this study. Since there are only two sizes of customers available (10 and 50 customers), the medium-size instances are generated by splitting instances with 50 customers into 20 customers, and the customers more than 50 are generated by merging 50 customers into 100, 200, 300, and 400 customers. The original instances consist of the demand of multiple products for each customer, the distance matrix, the fleet size, and the vehicle capacity. In our study we only consider three types of products $|P| = 3$. Therefore, we select instances with three product types from Henke et al. [13] with non-zero demand values. However, some additional parameters need to be added and modified to imitate the MVPTPFC-MC scenario as follows.

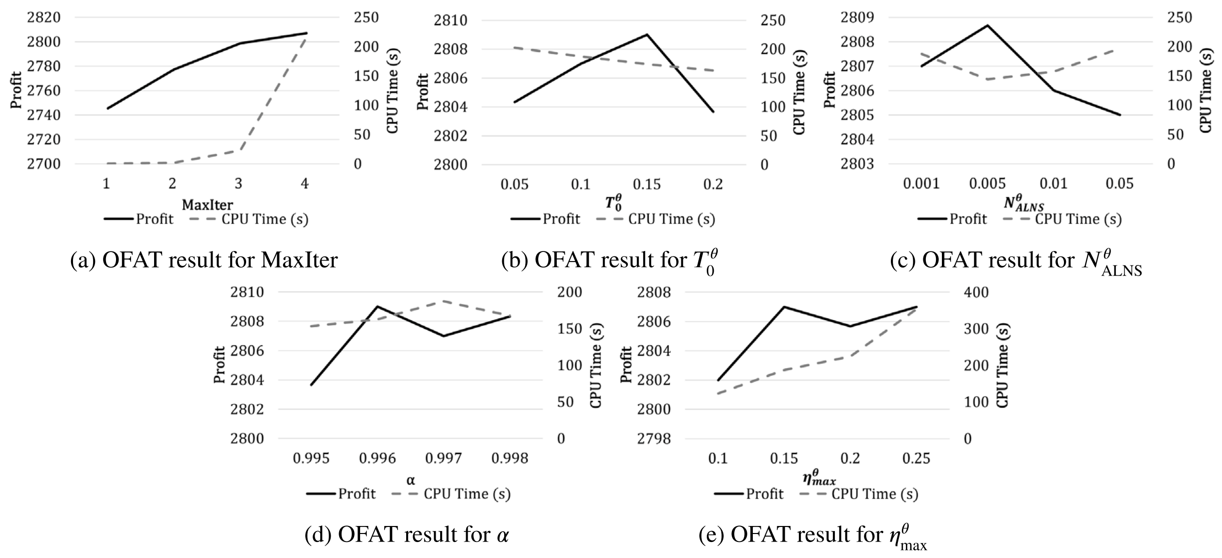
1. The service time is added for each customer, and the maximum working time is included for each vehicle.
2. The vehicle capacity is reduced by 50% as the number of available vehicles is increased by 50%.
3. The customers are divided into two types: optional and mandatory.

Each instance is named as "$|N|\_|N_M|\_J$". Here, $|N|$ represents the size of customers, $|N_M|$ represents the size of mandatory customers, and J is the serial number of each instance if there is more than one instance with the same size. For instance, consider 10_5_4, in which the fourth instance consists of 10 customers, out of which 5 are mandatory customers. Similarly, 50_10_2 is the second instance consisting of 50 customers, out of which 10 are mandatory customers.

5.2. *Parameter tuning*

Since many parameters are considered in our proposed ALNS, we only conduct an experimental design to tune some parameters that have a significant impact on solution quality and CPU time. These parameters are the maximum iteration MaxIter, the rate of iterations per segment N^{θ}_{ALNS}, the initial temperature rate T^{θ}_0, the cooling rate α, and the maximum non-improvement factor η^{θ}_0. We follow an experimental design-based parameter tuning procedure of Yu et al. [54,55,56]. We first start by tuning each parameter by the one factor at a time (OFAT) method. Four different levels of each parameter are initialized, and two instances from each problem size are selected to test the parameters. Fig. 5 illustrates the OFAT results for each parameter. According to this result, two levels are selected based on the average gap percentage and computational time for the 2^k factorial experiment. We select parameter values that obtain good objective values within a reasonable time.

The values of T_f, $q_{\min}$, and $q_{\max}$ can be determined based on the characteristics of the problem at hand. In order to enlarge the solution exploration, we set T_f to 0.0 and $q_{\min}$ to 0. Additionally, we set the value of $q_{\max}$ as the proportion of $|N_M|/|N|$ to increase the likelihood of finding feasible solutions. The other parameter values such as p, ω, δ_1, δ_2, and δ_3 are adopted from Ropke and Pisinger [36]. Table 2 summarizes the selected parameter values. For solving small, medium, and large problems efficiently, we set different values of MaxIter: i.e., $10^4, 10^6$, and 10^7, respectively.



(a) OFAT result for MaxIter (b) OFAT result for T_0^θ (c) OFAT result for N_{ALNS}^θ

(d) OFAT result for α (e) OFAT result for η_{max}^θ

Fig. 5. OFAT results.

Table 2
Selected parameters.

Parameter	Value
MaxIter	$\{10^4, 10^6, 10^7\}$
T_0^θ	0.15
N_{ALNS}^θ	0.005
α	0.996
η_{max}^θ	0.15
T_f	0.0
q_{min}	0
q_{max}	$\lvert N_M \rvert / \lvert N \rvert$
p	3
ω	0.1
$\{\delta_1, \delta_2, \delta_3\}$	$\{33, 9, 13\}$

5.3. *Performance of ALNS for solving MVPTPFC-MC instances*

We solve the mathematical model with a commercial solver, GUROBI, with a time limit of 47,000 s. Since no state-of-the-art algorithm exists, we compare the proposed ALNS with its preceding method, the Large Neighborhood Search (LNS) algorithm. There are 50 test instances in total generated according to Section 5.1. ALNS and LNS are executed 10 times in solving each instance. The performance of the proposed algorithm is evaluated by calculating the gap percentage of the best solutions (Gap_b) and average solutions (Gap_a) as follows.

$$\text{Gap}_\text{b} = \frac{\text{Obj}_\text{GUROBI} - \text{Best}_\text{ALNS/LNS}}{\text{Obj}_\text{GUROBI}} \times 100\% \qquad (22)$$

$$\text{Gap}_\text{a} = \frac{\text{Obj}_\text{GUROBI} - \text{Average}_\text{ALNS/LNS}}{\text{Objective}_\text{GUROBI}} \times 100\% \qquad (23)$$

Tables 3 and 4 summarize the computational results for the small- and medium-size problems, respectively. The first column is the name of the instance, and the second and the third columns are the objective function and CPU time in seconds of GUROBI. The computational results of LNS are presented in the fourth to eighth columns, consisting of the average objective, best objective, CPU time in seconds, Gap_a, and Gap_b, respectively. Finally, the ninth to thirteenth columns represent the computational results of ALNS, including the average and best objectives, CPU time in seconds, Gap_a, and Gap_b, respectively.

According to Table 3, GUROBI can solve each instance optimally within approximately 30 s on average. LNS and ALNS can produce these optimal solutions for each instance within a significantly less CPU time. Specifically, LNS obtains the optimal solution in each run within 0.17 s on average, as shown by 0.00% gaps for both average and best objectives. Similarly, ALNS also obtains 0.00% for the average gap percentage for both average and best objectives. However, ALNS outperforms LNS by obtaining these optimal solutions within 0.032 s. This justifies the convergence ability of the proposed ALNS.

In solving medium-size problems, GUROBI only obtain feasible solutions within the time limit (see: Table 4). Furthermore, LNS and ALNS produce solutions equal to or better than GUROBI within a significantly shorter time. LNS and ALNS produce better solutions for three instances and equal solutions with equal gaps, particularly −0.22% for both average and best run gaps. However, only one among eight instances is solved in a longer time by ALNS than by LNS. Specifically, ALNS solves the medium problem within 90.45 s on average, whereas LNS solves them in 98.21 s on average.

In solving large-size problems, GUROBI cannot produce feasible solutions within the time limit. Hence, both algorithms are evaluated with the best solution. To calculate Gap_a and Gap_b, we replace Obj_GUROBI with the best solution in (22) and (23), respectively. Table 5 presents the computational results for the large-size problems. The first and second columns present the instance name and the best solution obtained by the algorithms. The third to seventh columns present the computational results of LNS, consisting of the average and the best objectives, CPU time in seconds, Gap_a, and Gap_b, respectively. The eighth to twelfth columns are the computational results of ALNS, which include the average and the best objectives, CPU time in seconds, Gap_a, and Gap_b.

While LNS finds the best solutions for three instances, ALNS is able to find the remaining nine best solutions. Among 12 instances, only three instances are solved longer than LNS. Moreover, as the size of the node increases, the difference in CPU time between both algorithms becomes more significant. In terms of solution quality, ALNS also outperforms LNS with a 0.74% average gap and 0.07% best gap, whereas LNS obtains 0.92% and 0.30% average and best gaps.

In summary, the ALNS and LNS algorithms are both strong options for solving various problem sizes of MVPTPFC-MC. However, ALNS is better than LNS in terms of both solution quality and efficiency. This is supported by significant differences in computational results as the problem size increases. Therefore, ALNS algorithm is capable of solving real-world HRDS problems effectively, as evidenced by the data presented in Table 5.

5.4. *Analysis on the flexibility of the compartment sizes*

This section describes how FDCS can increase HRDS profitability under different scenarios. We run this experiment by testing FDCS and

Table 3
Computational results of the small instances.

Instance	GUROBI				LNS				ALNS			
	Obj.	t(s)	Avg.	Best	t(s)	Gap_a	Gap_b	Avg.	Best	t(s)	Gap_a	Gap_b
10_5_1	**1095**	42.72	1095	1095	0.142	0.00%	0.00%	1095	1095	0.036	0.00%	0.00%
10_5_2	**1177**	156.31	1177	1177	0.144	0.00%	0.00%	1177	1177	0.024	0.00%	0.00%
10_5_3	**1072**	12.97	1072	1072	0.148	0.00%	0.00%	1072	1072	0.023	0.00%	0.00%
10_5_4	**859**	53.06	859	859	0.146	0.00%	0.00%	859	859	0.028	0.00%	0.00%
10_5_5	**1055**	7.30	1055	1055	0.138	0.00%	0.00%	1055	1055	0.023	0.00%	0.00%
10_4_1	**1320**	28.11	1320	1320	0.157	0.00%	0.00%	1320	1320	0.045	0.00%	0.00%
10_4_2	**1159**	54.00	1159	1159	0.151	0.00%	0.00%	1159	1159	0.026	0.00%	0.00%
10_4_3	**1008**	21.16	1008	1008	0.154	0.00%	0.00%	1008	1008	0.028	0.00%	0.00%
10_4_4	**906**	26.77	906	906	0.163	0.00%	0.00%	906	906	0.026	0.00%	0.00%
10_4_5	**902**	10.78	902	902	0.151	0.00%	0.00%	902	902	0.034	0.00%	0.00%
10_3_1	**1231**	38.61	1231	1231	0.157	0.00%	0.00%	1231	1231	0.028	0.00%	0.00%
10_3_2	**1227**	18.17	1227	1227	0.166	0.00%	0.00%	1227	1227	0.037	0.00%	0.00%
10_3_3	**1239**	21.00	1239	1239	0.168	0.00%	0.00%	1239	1239	0.024	0.00%	0.00%
10_3_4	**1085**	6.81	1085	1085	0.173	0.00%	0.00%	1085	1085	0.029	0.00%	0.00%
10_3_5	**860**	15.53	860	860	0.173	0.00%	0.00%	860	860	0.029	0.00%	0.00%
10_2_1	**1115**	10.17	1115	1115	0.184	0.00%	0.00%	1115	1115	0.032	0.00%	0.00%
10_2_2	**1019**	12.34	1019	1019	0.166	0.00%	0.00%	1019	1019	0.031	0.00%	0.00%
10_2_3	**1003**	24.02	1003	1003	0.168	0.00%	0.00%	1003	1003	0.029	0.00%	0.00%
10_2_4	**846**	15.41	846	846	0.184	0.00%	0.00%	846	846	0.035	0.00%	0.00%
10_2_5	**1269**	23.55	1269	1269	0.186	0.00%	0.00%	1269	1269	0.034	0.00%	0.00%
10_1_1	**960**	31.83	960	960	0.191	0.00%	0.00%	960	960	0.033	0.00%	0.00%
10_1_2	**1004**	39.81	1004	1004	0.202	0.00%	0.00%	1004	1004	0.045	0.00%	0.00%
10_1_3	**894**	42.49	894	894	0.184	0.00%	0.00%	894	894	0.037	0.00%	0.00%
10_1_4	**908**	29.95	908	908	0.185	0.00%	0.00%	908	908	0.039	0.00%	0.00%
10_1_5	**1056**	13.64	1056	1056	0.188	0.00%	0.00%	1056	1056	0.036	0.00%	0.00%
10_0_1	**858**	17.45	858	858	0.196	0.00%	0.00%	858	858	0.032	0.00%	0.00%
10_0_2	**1022**	19.81	1022	1022	0.192	0.00%	0.00%	1022	1022	0.030	0.00%	0.00%
10_0_3	**996**	9.53	996	996	0.180	0.00%	0.00%	996	996	0.037	0.00%	0.00%
10_0_4	**969**	6.75	969	969	0.185	0.00%	0.00%	969	969	0.032	0.00%	0.00%
10_0_5	**1131**	28.95	1131	1131	0.174	0.00%	0.00%	1131	1131	0.035	0.00%	0.00%
Average	1041.50	27.97	1041.50	1041.50	0.170	0.00%	0.00%	1041.50	1041.50	0.032	0.00%	0.00%

Values in **bold** are the optimal solution obtained by GUROBI.

Table 4
Computational results of the medium instances.

Instance	GUROBI				LNS				ALNS			
	Obj.	t(s)	Avg.	Best	t(s)	Gap_a	Gap_b	Avg.	Best	t(s)	Gap_a	Gap_b
20_15_1	2077	47 000	2077	2077	105.82	0.00%	0.00%	2077	2077	93.03	0.00%	0.00%
20_15_2	2315	47 000	2318	2318	83.83	−0.13%	−0.13%	2318	2318	86.91	−0.13%	−0.13%
20_15_3	2272	47 000	2272	2272	123.41	0.00%	0.00%	2272	2272	98.81	0.00%	0.00%
20_15_4	1943	47 000	1971	1971	93.22	−1.44%	−1.44%	1971	1971	90.05	−1.44%	−1.44%
20_15_5	2181	47 000	2181	2181	97.53	0.00%	0.00%	2181	2181	91.49	0.00%	0.00%
20_15_6	1976	47 000	1976	1976	91.89	0.00%	0.00%	1976	1976	88.88	0.00%	0.00%
20_15_7	1815	47 000	1818	1818	88.21	−0.17%	−0.17%	1818	1818	84.07	−0.17%	−0.17%
20_15_8	1839	47 000	1839	1839	101.79	0.00%	0.00%	1839	1839	90.39	0.00%	0.00%
Average	2052.25	47 000	2056.5	2056.5	98.21	−0.22%	−0.22%	2056.5	2056.5	90.45	−0.22%	−0.22%

Table 5
Computational results of the large instances.

Instance	Best Solution	LNS					ALNS				
		Avg.	Best	t(s)	Gap_a	Gap_b	Avg.	Best	t(s)	Gap_a	Gap_b
50_40_1	2485	2444	2480	632.17	1.65%	0.20%	2456	**2485**	581.38	1.18%	0.00%
50_40_2	2689	2654	**2689**	789.30	1.30%	0.00%	2671	2688	502.42	0.67%	0.04%
50_35_1	5483	5428	5452	564.44	1.00%	0.57%	5437	**5483**	482.97	0.84%	0.00%
50_35_2	5012	4982	**5012**	552.35	0.60%	0.00%	4988	5004	454.11	0.48%	0.16%
50_30_1	3805	3772	**3805**	565.03	0.87%	0.00%	3761	3782	609.86	1.16%	0.60%
50_30_2	3826	3794	3811	604.91	0.84%	0.39%	3802	**3826**	659.90	0.63%	0.00%
50_25_1	3045	3019	3044	636.81	0.85%	0.03%	3021	**3045**	709.00	0.79%	0.00%
50_25_2	3617	3587	3615	667.66	0.83%	0.06%	3603	**3617**	584.09	0.39%	0.00%
100_50	10 634	10 578	10 602	2492.11	0.53%	0.30%	10 603	**10 634**	2431.03	0.29%	0.00%
200_100	18 466	18 387	18 411	8708.76	0.43%	0.30%	18 392	**18 466**	8347.39	0.40%	0.00%
300_150	25 314	25 072	25 162	18 004.60	0.96%	0.60%	25 052	**25 314**	17 186.3	1.04%	0.00%
400_200	30 818	30 438	30 471	26 374.90	1.23%	1.13%	30 517	**30 818**	25 958.9	0.98%	0.00%
Average	12 440.63	12 330.88	12 365.13	5049.42	0.92%	0.30%	12 343.93	12 437.75	4875.61	0.74%	0.07%

Values in **bold** are the best solution obtained by the algorithms.

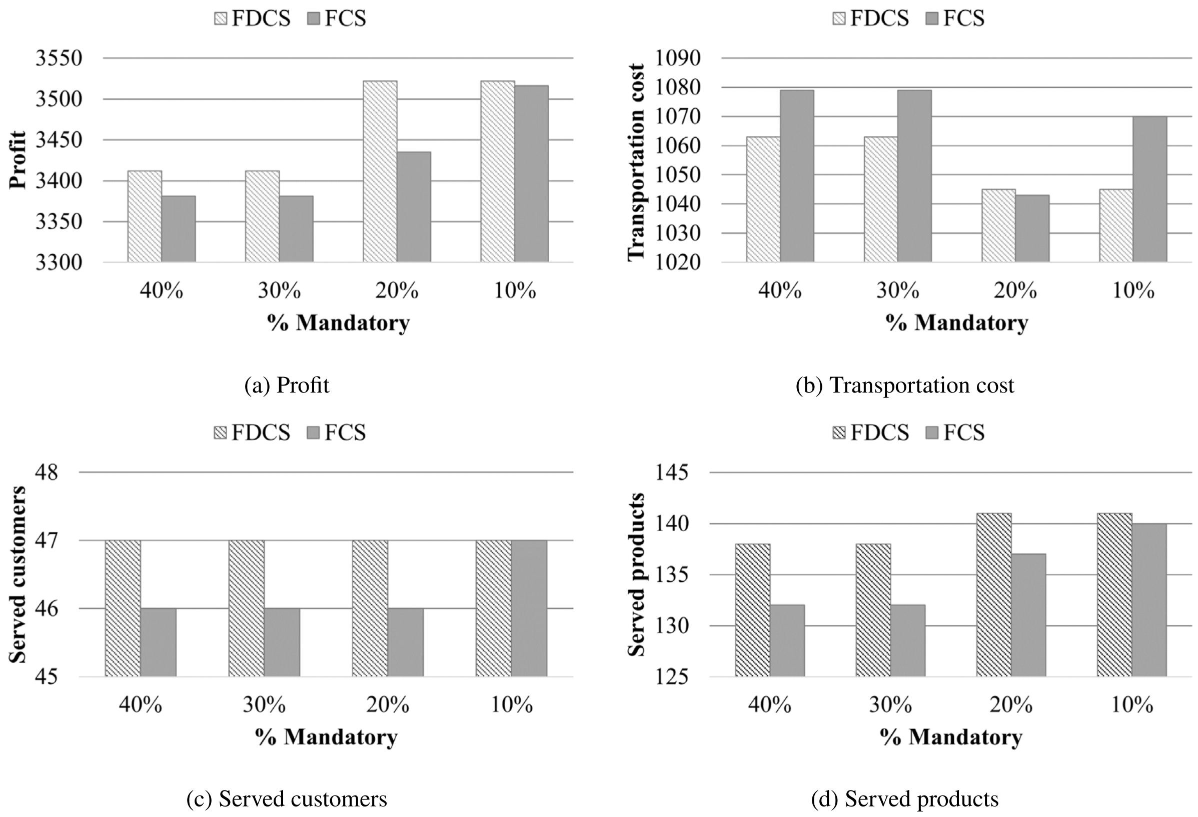


Fig. 6. The performance matrices affected by the flexibility of the compartment sizes.

FCS models for 50 customers. In FCS the configuration of compartments is predetermined. In contrast, the compartment configuration of FDCS must be determined in the model. For this particular test, we set the fleet size to $|K| = 5$ with vehicle capacity $Q = 500$. In the FCS model the compartments are all identical in size with a capacity of $q = 50$, whereas in FDCS three sizes of compartments are available with capacities $q = \{10, 20, 50\}$. Fig. 6 compares some performance matrices of using FDCS and FCS in HRDS. The horizontal axis represents the percentage of mandatory customers, and the vertical axis is the performance matrix. For instance, 10% includes 5 mandatory customers and 45 optional customers, and 40% consists of 20 mandatory customers and 30 optional customers.

Based on the results shown in Fig. 6(a), it can be concluded that the use of FDCS yields a relatively larger profit than the use of FCS in every scenario. Additionally, both profits tend to increase as the number of mandatory customers increases. Both models have the flexibility to choose which customers to serve. This means that they tend to select customers with higher profit margins and lower transportation costs when more optional customers are considered. As a result, they prioritize serving more profitable customers. Additionally, Fig. 6(b) illustrates that utilizing FDCS results in lower transportation costs compared to using FCS in most scenarios. This highlights the advantage of using FDCS as it optimizes vehicle utilization, resulting in serving more customers with less travel.

We note that the use of FDCS for 20% mandatory customers results in a slightly higher transportation cost compared to FCS, but it generates a significantly higher profit. This is because FDCS serves more customers and products than FCS, as shown in Figs. 6(c) and 6(d). Although the difference in served customers may not be significant, the served products are significantly higher in FDCS. Additionally, the served products tend to increase with an increase in optional customers, but a significant difference is observed in higher mandatory customers. To sum up, using FDCS in HRDS can increase profit, especially when more mandatory customers need to be served and the fleet size is limited.

5.5. *Sensitivity analysis*

In this particular test, we consider MVPTPFC-MC with 50 customers and different percentages of mandatory customers ranging from 10% up to 100%. It is worth noting when the percentage is 100% that the problem becomes multi-trip and multi-compartment VRP with FDCS. Furthermore, fleet sizes $|K|$ range from 4 to 8, with each vehicle having a capacity Q of 500 available. Fig. 7 illustrates the changes in different matrices due to different scenarios of mandatory customers and fleet size. The horizontal axis represents the percentage of mandatory customers, and the vertical axis represents the matrices.

As depicted in Fig. 7(a), reducing the fleet size results in a decrease in the total profit. This is because, as Figs. 7(c) and 7(d) demonstrate, the number of customers and products served also decreases with the reduction in vehicle fleet. Additionally, even if the number of customers and products served remains the same, the total profit may differ due to the total transportation cost, as highlighted in Fig. 7(b) for $|K| = \{8, 7, 6\}$. This is due to reduced vehicle capacity, leading to more travel or trips required to serve the same amount of customers and products.

It is important to note that there may be some problems that cannot feasibly be solved due to insufficient vehicles. This is the case when the set of available vehicles is either 5 or 4, and the percentage of mandatory customers is above 40%. However, if the mandatory percentage is 40% or less, then some customers can still be served, because there is some flexibility in deciding which customers to serve. Furthermore, the profit increases as the mandatory percentage decreases. Again, this can be explained by the fact that more profitable customers can be selected when there are more optional customers to consider. This demonstrates the significance of modeling HRDS with the PTP model instead of with the VRP model, particularly when the fleet size is limited. In

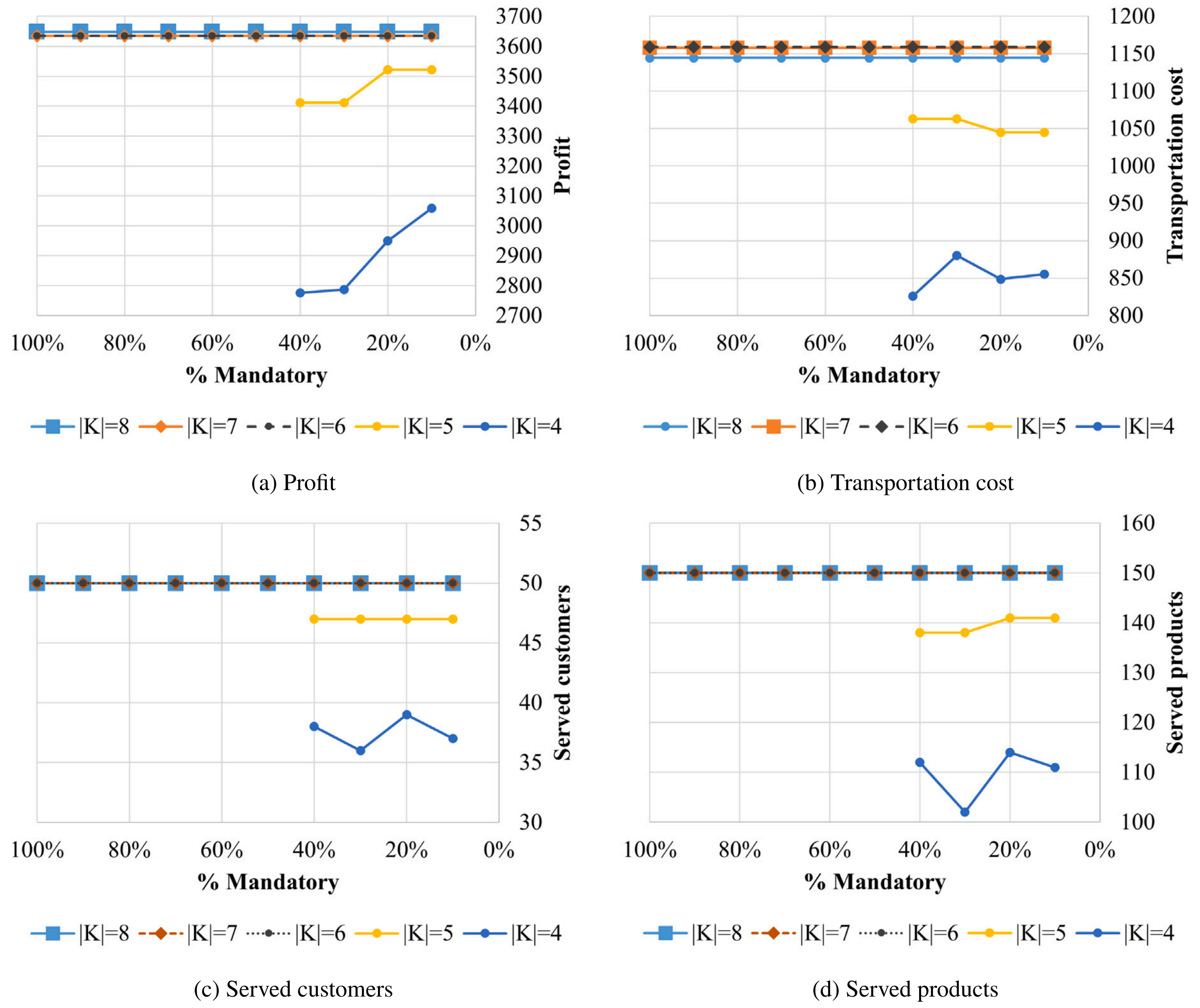


Fig. 7. The performance matrices affected by the fleet size and the number of mandatory customers.

summary, although the size of the fleet can have a significant impact on MVPTPFC-MC, it can be adjusted flexibly to increase profits.

6. Conclusion and future research

This study introduces a new variant of the Profitable Tour Problem, the multi-vehicle profitable tour problem with flexible compartments and mandatory customers, by incorporating MTVRP and MCVRP into PTP. Adopting the home-refill delivery system, the key characteristics of the proposed problem are: (1) two types of customers according to the demand fulfillment are considered: optional and mandatory; (2) a limited size of the vehicle fleet and capacity is utilized; hence, to serve all customers, multiple delivery trips are required; and (3) multiple products with small quantities of demand are considered; hence, multiple compartments with flexible discrete sizes are utilized. The decisions of MVPTPFC-MC are to determine which optional customer to be served by each vehicle on each trip, the route of each vehicle on each trip to serve the customers, and the compartment arrangement of each vehicle on each trip such that the objective is maximized. The objective function is to maximize the difference between the total profit and routing cost, which is the main problem of PTP.

To solve this problem, we develop a MILP model and an adaptive large neighborhood search. Additionally, some adjusted operators are also introduced according to the problem characteristics. This involves the procedures for destroying and repairing configurations, as well as the assignment of compartments based on a flexible discrete compartment size scheme. New instances with various sizes of small, medium, and large are also generated by considering MVPTPFC-MC's characteristics. This includes real-sized problems with 100 up to 400 customers. We compare ALNS performance with its preceding method, LNS, and a commercial solver, GUROBI, executing them on MVPTPFC-MC test instances. Based on the computational experiments conducted, both ALNS and LNS are effective solution approaches that efficiently obtain high-quality solutions, particularly for small- and medium-sized problems. However, ALNS outperforms LNS in terms of the CPU time. In fact, both ALNS and LNS are capable of solving small MVPTPFC-MC problems to optimality with significantly less CPU time than GUROBI. For medium-sized MVPTPFC-MC problems, both ALNS and LNS provide three better and five equal solutions in significantly less CPU time than GUROBI. Moreover, out of twelve instances, ALNS obtains the best solutions in nine instances, while LNS could only obtain three best solutions for large MVPTPFC-MC problems. Furthermore, in most instances, ALNS can solve them with less CPU time than LNS, including real-sized instances of up to 400 customers. To sum up, the proposed ALNS is robust and efficient in solving MVPTPFC-MC problems.

We present additional experiments to prove the effectiveness of using flexible discrete compartment sizes (FDCS) in comparison to fixed compartment sizes models. Our research highlights the benefits of using FDCS, which optimizes vehicle utilization. By using FDCS, one can serve more customers while traveling a shorter distance, resulting in greater profits. The use of FDCS is especially beneficial when the fleet size is limited and there are more mandatory customers to serve. In addition, sensitivity analysis is conducted to evaluate the performance metrics in different scenarios of fleet size and mandatory customers to serve. The analysis shows that the HRDS profit increases as the number of mandatory customers decreases, particularly when the fleet size is limited.

While the proposed study has contributed to filling an existing research gap, there are still a few limitations that need to be addressed in the future. First, more realistic problem characteristics should be incorporated into MVPTPFC-MC. For instance, time windows can be added to the problem, such as in MTVRPTW [6,24,33,47,57,58], VRP with profits (i.e., Team Orienteering Problem with Time Windows) [59], and soft time windows for multiple trips and multiple compartment vehicle routing problem [60]. The multi-depot problem also can be considered in the home-refill delivery system because, in real applications, the company may have multiple warehouses [57].

Second, there could be some improvements to the proposed ALNS to enhance its efficiency in solving MVPTPFC-MC. One way to achieve this is by using advanced parameter tuning techniques such as IRACE or reinforcement learning-based parameter tuning. Additionally, implementing a hybrid ALNS approach by integrating the ALNS results with the exact approach as the initial solution could be beneficial, especially in solving complex problems that cannot be solved using GUROBI. Lastly, as new instances have been generated for MVPTPFC-MC in this study, more comprehensive solution methodologies can be developed as a benchmark approach. This includes other metaheuristic approaches such as Variable Neighborhood Search and Tabu Search as they have been developed for solving MVPTP, MCVRP, and PTVRP [13,30,49]. Constructing the exact approaches could be one future research direction as it has been proven to be effective in solving MVPTP, MTVRP, and MCVRP [6,25,31].

CRediT authorship contribution statement

Vincent F. Yu: Conceptualization, Funding acquisition, Methodology, Resources, Supervision, Validation, Writing – review & editing. **Nabila Yuraisyah Salsabila:** Conceptualization, Formal analysis, Investigation, Methodology, Visualization, Writing – original draft, Data curation, Software, Validation. **Aldy Gunawan:** Formal analysis, Supervision, Visualization, Writing – review & editing. **Anggun Nurfitriani Handoko:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This research is partially supported by the National Science and Technology Council of Taiwan under grant NSTC 112-2410-H-011-004-MY3 and the National Taiwan University of Science and Technology under grant NTUST-ITB-111-05.

References

[1] P. Avella, M. Boccia, A. Sforza, Solving a fuel delivery problem by heuristic and exact approaches, European J. Oper. Res. 152 (1) (2004) 170–179, http://dx.doi.org/10.1016/S0377-2217(02)00676-8, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221702006768.

[2] V.F. Yu, G. Aloina, T. Eccarius, Adoption intentions of home-refill delivery service for fast-moving consumer goods, Transp. Res. E 171 (2023) 103041, http://dx.doi.org/10.1016/j.tre.2023.103041, URL https://linkinghub.elsevier.com/retrieve/pii/S1366554523000297.

[3] V.A. Lofthouse, T.A. Bhamra, R.L. Trimingham, Investigating customer perceptions of refillable packaging and assessing business drivers and barriers to their use, Packaging Technol. Sci. 22 (6) (2009) 335–348, http://dx.doi.org/10.1002/pts.857.

[4] P. Karakostas, A. Sifaleras, Recent trends in sustainable supply-chain optimization, in: M. Fathi, E. Zio, P.M. Pardalos (Eds.), Handbook of Smart Energy Systems, Springer International Publishing, Cham, 2021, pp. 1–23, http://dx.doi.org/10.1007/978-3-030-72322-4_181-1.

[5] E.-G.T. Sohrab Faramarzi-Oghani, R. Tavakkoli-Moghaddam, Meta-heuristics for sustainable supply chain management: a review, Int. J. Prod. Res. 61 (6) (2023) 1979–2009, http://dx.doi.org/10.1080/00207543.2022.2045377.

[6] F. Hernandez, D. Feillet, R. Giroudeau, O. Naud, A new exact algorithm to solve the multi-trip vehicle routing problem with time windows and limited duration, 4OR 12 (3) (2014) 235–259, http://dx.doi.org/10.1007/s10288-013-0238-z, URL http://link.springer.com/10.1007/s10288-013-0238-z.

[7] B. Fleischmann, The vehicle routing problem with multiple use of vehicles, 1990.

[8] C. Tarantilis, F. Stavropoulou, P. Repoussis, The capacitated team orienteering problem: A bi-level filter-and-fan method, European J. Oper. Res. 224 (1) (2013) 65–78, http://dx.doi.org/10.1016/j.ejor.2012.07.032, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221712005735.

[9] C. Archetti, M.G. Speranza, D. Vigo, Chapter 10: Vehicle routing problems with profits, in: P. Toth, D. Vigo (Eds.), Vehicle Routing, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014, pp. 273–297, http://dx.doi.org/10.1137/1.9781611973594.ch10, URL http://epubs.siam.org/doi/10.1137/1.9781611973594.ch10.

[10] D.L. Cortés-Murcia, H.M. Afsar, C. Prodhon, Multi-period profitable tour problem with electric vehicles and mandatory stops, Int. J. Sustain. Transp. (2022) 1–17.

[11] S. Martins, M. Ostermeier, P. Amorim, A. Hübner, B. Almada-Lobo, Product-oriented time window assignment for a multi-compartment vehicle routing problem, European J. Oper. Res. 276 (3) (2019) 893–909, http://dx.doi.org/10.1016/j.ejor.2019.01.053, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221719300888.

[12] M. Ostermeier, T. Henke, A. Hübner, G. Wäscher, Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions, European J. Oper. Res. 292 (3) (2021) 799–817, http://dx.doi.org/10.1016/j.ejor.2020.11.009, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221720309553.

[13] T. Henke, M.G. Speranza, G. Wäscher, The multi-compartment vehicle routing problem with flexible compartment sizes, European J. Oper. Res. 246 (3) (2015) 730–743, http://dx.doi.org/10.1016/j.ejor.2015.05.020, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221715004105.

[14] S.T. Windras Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, A.P. Rifai, A survey of adaptive large neighborhood search algorithms and applications, Comput. Oper. Res. 146 (2022) 105903, http://dx.doi.org/10.1016/j.cor.2022.105903, URL https://www.sciencedirect.com/science/article/pii/S0305054822001654.

[15] M. Sevaux, K. Sörensen, N. Pillay, Adaptive and multilevel metaheuristics, 2015.

[16] P. Karakostas, A. Sifaleras, A double-adaptive general variable neighborhood search algorithm for the solution of the traveling salesman problem, Appl. Soft Comput. 121 (2022) 108746, http://dx.doi.org/10.1016/j.asoc.2022.108746, URL https://www.sciencedirect.com/science/article/pii/S1568494622001879.

[17] L. Wang, J. Kinable, T. Van Woensel, The fuel replenishment problem: A split-delivery multi-compartment vehicle routing problem with multiple trips, Comput. Oper. Res. 118 (2020) 104904, http://dx.doi.org/10.1016/j.cor.2020.104904, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054820300216.

[18] N. Azi, M. Gendreau, J.-Y. Potvin, An adaptive large neighborhood search for a vehicle routing problem with multiple routes, Comput. Oper. Res. 41 (2014) 167–173, http://dx.doi.org/10.1016/j.cor.2013.08.016, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054813002220.

[19] S. Daniel Handoko, L.H. Chuin, A. Gupta, O.Y. Soon, H.C. Kim, T.P. Siew, Solving multi-vehicle profitable tour problem via knowledge adoption in evolutionary bi-level programming, in: 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, Sendai, Japan, 2015, pp. 2713–2720, http://dx.doi.org/10.1109/CEC.2015.7257225, URL http://ieeexplore.ieee.org/document/7257225/.

[20] M. Gansterer, M. Küçüktepe, R.F. Hartl, The multi-vehicle profitable pickup and delivery problem, OR Spectrum 39 (1) (2017) 303–319, http://dx.doi.org/10.1007/s00291-016-0454-y, URL http://link.springer.com/10.1007/s00291-016-0454-y.

[21] A.I. Alhujaylan, I. Manar, A GRASP-based solution construction approach for the multi-vehicle profitable pickup and delivery problem, Int. J. Adv. Comput. Sci. Appl. 10 (4) (2019) http://dx.doi.org/10.14569/IJACSA.2019.0100412, URL http://thesai.org/Publications/ViewPaper?Volume=10&Issue=4&Code=IJACSA&SerialNo=12.

[22] D.L. Cortés-Murcia, H.M. Afsar, C. Prodhon, A branch and price algorithm for the electric capacitated profitable tour problem with mandatory stops, IFAC-PapersOnLine 52 (13) (2019) 1572–1577, http://dx.doi.org/10.1016/j.ifacol.2019.11.424, URL https://www.sciencedirect.com/science/article/pii/S2405896319314053, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[23] F. Cornillier, F.F. Boctor, G. Laporte, J. Renaud, An exact algorithm for the petrol station replenishment problem, J. Oper. Res. Soc. 59 (5) (2008) 607–615, http://dx.doi.org/10.1057/palgrave.jors.2602374, URL https://www.tandfonline.com/doi/full/10.1057/palgrave.jors.2602374.

[24] F. Cornillier, F. Boctor, J. Renaud, Heuristics for the multi-depot petrol station replenishment problem with time windows, European J. Oper. Res. 220 (2) (2012) 361–369, http://dx.doi.org/10.1016/j.ejor.2012.02.007, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221712001245.

[25] L.C. Coelho, G. Laporte, Classification, models and exact algorithms for multi-compartment delivery problems, European J. Oper. Res. 242 (3) (2015) 854–864, http://dx.doi.org/10.1016/j.ejor.2014.10.059, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221714008984.

[26] L. Muyldermans, G. Pang, On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm, European J. Oper. Res. 206 (1) (2010) 93–103, http://dx.doi.org/10.1016/j.ejor.2010.02.020, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221710001256.

[27] T. Henke, M.G. Speranza, G. Wäscher, A branch-and-cut algorithm for the multi-compartment vehicle routing problem with flexible compartment sizes, Ann. Oper. Res. 275 (2) (2019) 321–338, http://dx.doi.org/10.1007/s10479-018-2938-4, URL http://link.springer.com/10.1007/s10479-018-2938-4.

[28] H. Zbib, G. Laporte, The commodity-split multi-compartment capacitated arc routing problem, Comput. Oper. Res. 122 (2020) 104994, http://dx.doi.org/10.1016/j.cor.2020.104994, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054820301118.

[29] R. Lahyani, L.C. Coelho, M. Khemakhem, G. Laporte, F. Semet, A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia, Omega 51 (2015) 1–10, http://dx.doi.org/10.1016/j.omega.2014.08.007, URL https://linkinghub.elsevier.com/retrieve/pii/S0305048314001017.

[30] A.E. Fallahi, C. Prins, R. Wolfler Calvo, A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem, Comput. Oper. Res. 35 (5) (2008) 1725–1741, http://dx.doi.org/10.1016/j.cor.2006.10.006, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054806002516.

[31] K. Heßler, Exact algorithms for the multi-compartment vehicle routing problem with flexible compartment sizes, European J. Oper. Res. 294 (1) (2021) 188–205, http://dx.doi.org/10.1016/j.ejor.2021.01.037, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221721000679.

[32] R. Lahyani, M. Khemakhem, F. Semet, Solving rich profitable tour problems, in: Eight Triennal Symposium on Transportation Analysis (TRISTAN VIII), San Pedro de Atacama, Chile, 2013.

[33] D. Cattaruzza, N. Absi, D. Feillet, Vehicle routing problems with multiple trips, 4OR 14 (3) (2016) 223–259, http://dx.doi.org/10.1007/s10288-016-0306-2, URL http://link.springer.com/10.1007/s10288-016-0306-2.

[34] A. Chbichib, R. Mellouli, H. Chabchoub, Profitable vehicle routing problem with multiple trips: Modeling and constructive heuristics, in: 2011 4th International Conference on Logistics, IEEE, Hammamet, Tunisia, 2011, pp. 500–507, http://dx.doi.org/10.1109/LOGISTIQUA.2011.5939450, URL https://ieeexplore.ieee.org/document/5939450.

[35] Y. Li, H. Chen, C. Prins, Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests, European J. Oper. Res. 252 (1) (2016) 27–38, http://dx.doi.org/10.1016/j.ejor.2015.12.032, URL https://www.sciencedirect.com/science/article/pii/S0377221715011716.

[36] S. Ropke, D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, Transp. Sci. 40 (4) (2006) 455–472, http://dx.doi.org/10.1287/trsc.1050.0135, URL https://pubsonline.informs.org/doi/10.1287/trsc.1050.0135.

[37] P. Sun, L.P. Veelenturf, M. Hewitt, T. Van Woensel, Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows, Transp. Res. E 138 (2020) 101942, http://dx.doi.org/10.1016/j.tre.2020.101942, URL https://www.sciencedirect.com/science/article/pii/S1366554519303734.

[38] P. Sun, L.P. Veelenturf, S. Dabia, T. Van Woensel, The time-dependent capacitated profitable tour problem with time windows and precedence constraints, European J. Oper. Res. 264 (3) (2018) 1058–1073, http://dx.doi.org/10.1016/j.ejor.2017.07.004, URL https://www.sciencedirect.com/science/article/pii/S0377221717306318.

[39] P. Sun, L.P. Veelenturf, M. Hewitt, T. Van Woensel, The time-dependent pickup and delivery problem with time windows, Transp. Res. B 116 (2018) 1–24, http://dx.doi.org/10.1016/j.trb.2018.07.002, URL https://www.sciencedirect.com/science/article/pii/S0191261518301231.

[40] H. Chentli, R. Ouafi, W.R. Cherif-Khettaf, A selective adaptive large neighborhood search heuristic for the profitable tour problem with simultaneous pickup and delivery services, RAIRO-Oper. Res. 52 (4–5) (2018) 1295–1328.

[41] M. Alinaghian, N. Shokouhi, Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search, Omega 76 (2018) 85–99, http://dx.doi.org/10.1016/j.omega.2017.05.002, URL https://www.sciencedirect.com/science/article/pii/S0305048316303553.

[42] L. Chen, Y. Liu, A. Langevin, A multi-compartment vehicle routing problem in cold-chain distribution, Comput. Oper. Res. 111 (2019) 58–66, http://dx.doi.org/10.1016/j.cor.2019.06.001, URL https://www.sciencedirect.com/science/article/pii/S0305054819301534.

[43] E. Mofid-Nakhaee, F. Barzinpour, A multi-compartment capacitated arc routing problem with intermediate facilities for solid waste collection using hybrid adaptive large neighborhood search and whale algorithm, Waste Manage. Res. 37 (1) (2019) 38–47, http://dx.doi.org/10.1177/0734242X18801186, PMID: 30319052.

[44] R. Eshtehadi, E. Demir, Y. Huang, Solving the vehicle routing problem with multi-compartment vehicles for city logistics, Comput. Oper. Res. 115 (2020) 104859, http://dx.doi.org/10.1016/j.cor.2019.104859, URL https://www.sciencedirect.com/science/article/pii/S0305054819303016.

[45] P. Grangier, M. Gendreau, F. Lehuédé, L.-M. Rousseau, An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization, European J. Oper. Res. 254 (1) (2016) 80–91, http://dx.doi.org/10.1016/j.ejor.2016.03.040, URL https://www.sciencedirect.com/science/article/pii/S0377221716301862.

[46] V. François, Y. Arda, Y. Crama, G. Laporte, Large neighborhood search for multi-trip vehicle routing, European J. Oper. Res. 255 (2) (2016) 422–441, http://dx.doi.org/10.1016/j.ejor.2016.04.065, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221716303034.

[47] V. François, Y. Arda, Y. Crama, Adaptive large neighborhood search for multitrip vehicle routing with time windows, Transp. Sci. 53 (6) (2019) 1706–1730, http://dx.doi.org/10.1287/trsc.2019.0909, URL https://pubsonline.informs.org/doi/10.1287/trsc.2019.0909.

[48] P.N.R.K. Wenli Li, K. Li, Multi-trip vehicle routing problem with order release time, Eng. Optim. 52 (8) (2020) 1279–1294, http://dx.doi.org/10.1080/0305215X.2019.1642880.

[49] B. Pan, Z. Zhang, A. Lim, Multi-trip time-dependent vehicle routing problem with time windows, European J. Oper. Res. 291 (1) (2021) 218–231, http://dx.doi.org/10.1016/j.ejor.2020.09.022, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221720308262.

[50] M.G. Avci, M. Avci, An adaptive large neighborhood search approach for multiple traveling repairman problem with profits, Comput. Oper. Res. 111 (2019) 367–385, http://dx.doi.org/10.1016/j.cor.2019.07.012, URL https://linkinghub.elsevier.com/retrieve/pii/S030505481930187X.

[51] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in: G. Goos, J. Hartmanis, J. Van Leeuwen, M. Maher, J.-F. Puget (Eds.), Principles and Practice of Constraint Programming — CP98, vol. 1520, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 417–431, http://dx.doi.org/10.1007/3-540-49481-2_30, URL http://link.springer.com/10.1007/3-540-49481-2_30, Series Title: Lecture Notes in Computer Science.

[52] F. Hammami, M. Rekik, L.C. Coelho, A hybrid adaptive large neighborhood search heuristic for the team orienteering problem, Comput. Oper. Res. 123 (2020) 105034, http://dx.doi.org/10.1016/j.cor.2020.105034, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054820301519.

[53] V.C. Hemmelmayr, J.-F. Cordeau, T.G. Crainic, An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics, Comput. Oper. Res. 39 (12) (2012) 3215–3228.

[54] V.F. Yu, P. Jewpanya, A.P. Redi, Y.-C. Tsao, Adaptive neighborhood simulated annealing for the heterogeneous fleet vehicle routing problem with multiple cross-docks, Comput. Oper. Res. 129 (2021) 105205, http://dx.doi.org/10.1016/j.cor.2020.105205, URL https://www.sciencedirect.com/science/article/pii/S0305054820303221.

[55] V.F. Yu, P. Jodiawan, A. Gunawan, An adaptive large neighborhood search for the green mixed fleet vehicle routing problem with realistic energy consumption and partial recharges, Appl. Soft Comput. 105 (2021) 107251, http://dx.doi.org/10.1016/j.asoc.2021.107251, URL https://www.sciencedirect.com/science/article/pii/S1568494621001745.

[56] V.F. Yu, P. Jodiawan, M.-L. Hou, A. Gunawan, Design of a two-echelon freight distribution system in last-mile logistics considering covering locations and occasional drivers, Transp. Res. E 154 (2021) 102461, http://dx.doi.org/10.1016/j.tre.2021.102461, URL https://www.sciencedirect.com/science/article/pii/S1366554521002246.

[57] L. Zhen, C. Ma, K. Wang, L. Xiao, W. Zhang, Multi-depot multi-trip vehicle routing problem with time windows and release dates, Transp. Res. E 135 (2020) 101866, http://dx.doi.org/10.1016/j.tre.2020.101866, URL https://linkinghub.elsevier.com/retrieve/pii/S1366554519301486.

[58] E. Babaee Tirkolaee, P. Abbasian, M. Soltani, S.A. Ghaffarian, Developing an applied algorithm for multi-trip vehicle routing problem with time windows in urban waste collection: A case study, Waste Manage. Res.: J. Sustain. Circular Econ. 37 (1_suppl) (2019) 4–13, http://dx.doi.org/10.1177/0734242X18807001, URL http://journals.sagepub.com/doi/10.1177/0734242X18807001.

[59] S.-W. Lin, V.F. Yu, Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing, Comput. Ind. Eng. 114 (2017) 195–205, http://dx.doi.org/10.1016/j.cie.2017.10.020, URL https://linkinghub.elsevier.com/retrieve/pii/S0360835217305053.

[60] P. Kabcome, T. Mouktonglang, Vehicle routing problem for multiple product types, compartments, and trips with soft time windows, Int. J. Math. Math. Sci. 2015 (2015) 1–9, http://dx.doi.org/10.1155/2015/126754, URL http://www.hindawi.com/journals/ijmms/2015/126754/.