The Fuel Replenishment Problem: A Split-Delivery Multi-Compartment Vehicle Routing Problem with Multiple Trips

L. Wang[a,b], J. Kinable[a,*], T. Van Woensel[a]

[a]School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
[b]Department of Oil and Gas Storage and Transportation Engineering, China University of Petroleum, Beijing, China

Abstract

In this paper, we formally define and model the Fuel Replenishment Problem (FRP). The FRP is a multi-compartment, multi-trip, split-delivery VRP in which tanker trucks transport different types of petrol, separated over multiple vehicle compartments, from an oil depot to petrol stations. Large customer demands often necessitate multiple deliveries. Throughout a single working day, a tanker truck returns several times to the oil depot to resupply. A solution to the FRP involves computing a delivery schedule of minimum duration, thereby determining for each vehicle (1) the allocation of oil products to vehicle compartments, (2) the delivery routes, and (3) the delivery patterns. To solve FRP efficiently, an Adaptive Large Neighborhood Search (ALNS) heuristic is constructed. The heuristic is evaluated on data from a Chinese petroleum transportation company and compared against exact results from a MILP model and lower bounds from a column generation approach. In addition, we perform sensitivity analysis on different problem features, including the number of vehicles, products, vehicle compartments and their capacities. Computational results show that the ALNS heuristic is capable of solving instances with up to 60 customers and 3 different products in less than 25 minutes with an average optimality gap of around 10%. On smaller instances, the heuristic finds optimal solutions in significantly less time than the exact MILP formulation.

Keywords: Fuel Replenishment, Multi-compartment, Multi-trip, Split-delivery, Vehicle Routing, Adaptive Large Neighbourhood Search

1. Introduction

This paper deals with a multi-compartment, multi-trip and split-delivery routing problem that occurs, for example, in the context of petrol station replenishment. We denote this problem as the Fuel Replenishment Problem (FRP). The cost of vehicle transportation accounts for 60% to 70% of the total cost of the existing refined oil logistics system [22]. Therefore, efficient vehicle routing is crucial. In general, a vehicle is separated into two to five compartments storing several incompatible oil products. These compartments

*Corresponding author
Email addresses: mxdsjyd@163.com (L. Wang), j.kinable@tue.nl (J. Kinable), t.v.Woensel@tue.nl (T. Van Woensel)

are inflexible to avoid leakage and contamination. Using debit meters, the content in one compartment can be split among several customers. Likewise, a petrol station can be supplied by multiple vehicles (split-delivery).

Petrol stations are replenished by a fleet of heterogeneous vehicles. The vehicles load fuel at a central depot, and then travel to one or more petrol stations to deliver their fuel. Once empty, the vehicles return to the central depot to resupply after which they continue servicing other fuel stations. At the end of the day, the vehicles return to the central depot.

Each fuel station carries one or more types of fuel. The fuel demand of some of the bigger fuel stations, typically located near busy highways, often exceeds the capacity of a single vehicle. Therefore, multiple vehicles may be required to satisfy the demand of a single station. The latter contrasts to traditional VRP problems in which a customer is visited only once by a single vehicle.

Following the categorization proposed by Coelho and Laporte [6], the Multi-Compartment Delivery Problem considered in this paper can be classified as a split-split delivery problem. For split-split VRPs, three decisions need to be made: (1) the vehicle routes, (2) the compartment assignment, i.e. the capacity of each product, and, (3) the delivery pattern of each route and each product. Every vehicle is allowed to execute multiple trips.

The main contributions of this paper are as follows:

- We propose a comprehensive variant of the split-delivery vehicle routing problem (SD-VRP) considering multiple compartments and multiple trips.
- We develop a Mixed Integer Programing (MILP) formulation for the described routing problem. To benchmark, we obtain exact solutions for some small instances using CPLEX. We additionally develop a column generation approach to compute lower bounds as an additional benchmark.
- We adapt the Adaptive Large Neighborhood Search (ALNS) heuristic to solve our problem for larger instances. The solutions obtained by the ALNS approach are compared to the solutions of both the MILP model and the lower bound obtained by the column generation approach. We show that our ALNS heuristic consistently outperforms MILP. The ALNS obtains the same or better solutions than the MILP for all small instances within 10 seconds per instance, whereas, for larger instances, solving the MILP directly does not give a feasible solution within a two hour time limit. For the largest instance with 60 customers and 20 vehicles, computation times of the ALNS stay well below 25 minutes, while obtaining an average optimality gap of 10%.

A problem similar to the FRP discussed in this paper can be found in Coelho and Laporte [6]. Coelho and Laporte [6] consider several exact branch-and-cut algorithms for a class of multi-compartment delivery problems based on formulations for multi-period inventory-routing problems. Using a branch-and-cut approach, they are able to solve instances with up to 10 customers, 3 products, 3 compartments, and 6 vehicles to optimality. In this work, the focus is on the design of an efficient metaheuristic capable of solving a FRP for a single planning period. In contrast to the VRPs discussed in Coelho and Laporte [6], the tanker trucks in our problem typically perform multiple trips a day. Moreover, the goal is to compute schedules of minimum duration, thereby spreading the workload over the available vehicles, as opposed to merely minimizing the total travel time.

The remainder of this paper is organized as follows: Section 2 presents a brief review of the existing work related to this paper. Section 3 formally describes our problem on-hand and gives details on the mathematical formulation. Section 4 introduces a column generation method, obtaining a lower bound for our problem on-hand. In Section 5, we develop an adaptive large neighborhood search (ALNS). In Section 6, we report on the computational results. We conclude the paper in Section 7.

2. Related Literature

In this paper, we discuss a vehicle routing problem involving multiple compartments, split-delivery, and multiple trips. There are a few papers related to the problem discussed in this paper. These papers are based on the assumption that the compartments in the vehicle are not equipped with debit meters, implying that a compartment should be completely emptied once the delivery is started. Avella et al. [4] studied a petrol replenishment problem involving product packing and vehicle routing. The authors simplified the problem by an assumption that each compartment of the vehicle must travel either full or empty. Other authors discussed variants of this petrol replenishment problem: single period [8], multi-period [9], with time windows [10] and multi-depot [7]. In these papers, the authors limit the number of customers (up to 4) for each trip according to a distribution policy used in practice.

Popović et al. [31] propose a Variable Neighborhood Search(VNS) heuristic for a multi-period Inventory Routing Problem (IRP) involving the distribution of petrol with a multi-compartment vehicle fleet. Similar to the aforementioned papers, Popović et al. [31] also assume only full compartments are delivered, and each vehicle can visit up to 3 customers per trip. Vidović et al. [35] developed a Variable Neighborhood Search (VNS) heuristic for a multi-product multi-period IRP with a multi-compartment homogeneous fleet. In addition, they proposed a MILP model for the problem and limited each route to 4 customers. Benantar et al. [5] considered a multi-compartment Inventory Routing Problem (IRP) with time windows. In contrast to Cornillier et al. [10], they do not limit the number of customers visited in a route.

Table 1 presents a comparison of the main references after the year 2000.

The split-delivery vehicle routing problem (SDVRP) also received attention in the literature. Dror and Trudeau [13] and [14] proposed a local search approach to solve the SDVRP. Afterwards, more complex approaches, such as metaheuristics and hybrid algorithms were introduced. Sierksma and Tijssen [34] proposed an algorithm based on a column generation algorithm, combined with other heuristics to solve a flight scheduling problem. Archetti et al. [3] proposed a Tabu search algorithm for the SDVRP. These authors also introduced a hybrid Tabu search and integer programming algorithm. There are few papers

Table 1: Literature comparison

	Fleet	TW	MC	MP/MT	IRP	MD	# Cust	Split
Avella et al. [4]	Heterog.	no	no	-	no	no	unlim.	-
Abdelaziz et al. [1]	Heterog.	no	yes	-	no	no	unlim.	u-u
Ng et al. [28]	Heterog.	no	yes	-	no	no	unlim.	u-u
Vidović et al. [35]	Homog.	no	yes	MP	yes	no	max. 3	u-u
Popović et al. [31]	Homog.	no	yes	MP	yes	no	max. 4	u-u
Benantar et al. [5]	Heterog.	yes	yes	-	yes	no	unlim.	u-u
Cornillier et al. [8]	Heterog.	no	yes	-	yes	no	max. 2	u-u
Cornillier et al. [9]	Heterog.	no	yes	MP	yes	no	max. 2	u-u
Cornillier et al. [10]	Heterog.	yes	yes	MT	yes	no	max. 4	u-u
Cornillier et al. [7]	Heterog.	yes	yes	MT	yes	yes	max. 3	u-u
Coelho and Laporte [6]	Homog.	no	yes	MP	yes	no	unlim.	u-u, u-s, s-u, s-s
This paper	Heterog.	no	yes	MT	no	no	unlim.	s-s

TW: Time Window; MC: Multi-Compartment; MP: Multi-Period; MT: Multi Trip; IRP: Inventory Routing Problem; MD: Multi-Depot; # Cust: Maximum number of customers per route; Split: Split: Possibility to split the compartment and the demand of the customer or not (Coelho and Laporte [6] for the details),'u' means unsplit and 's' means split.

that studied exact algorithms for SDVRP. Lee et al. [21] developed a dynamic programming method with infinite state and action spaces. Jin et al. [19] proposed a two-stage algorithm with valid inequalities to solve the SDVRP to optimality. Feillet et al. [16] introduced a branch-price-cut method to solve the SDVRP. Another branch-price-cut algorithm capable of solving SDVRP instances up to 100 customers is presented by Desaulniers [12]. Archetti et al. [2] enhanced the latter work by introducing several valid inqualities, and by proposing an efficient Tabu search to solve the pricing subproblem. Finally, a variant of the SDVRP with time-windows, split delivery, and weight-related cost is addressed by Luo et al. [23].

The literature on multi-compartment vehicle routing problem (MCVRP) covers various application domains including waste collection (Muyldermans and Pang [27] and Henke et al. [17]), foods or groceries delivery (Derigs et al. [11], Hübner and Ostermeier [18]) and in livestock transportation (Oppen and Løkketangen [29]). Lahyani et al. [20] solve a problem involving the collection of olive oil with different qualities with a branch and cut branch-and-cut algorithm. El Fallahi et al. [15] augment classical VRP instances with vehicle compartments. Mendoza et al. [25] use a memetic algorithm to solve an MCVRP with stochastic demands and developed a constructive heuristics for their algorithm (Mendoza et al. [26]). Melechovskỳ [24] extend the MCVRP by considering time windows and solved the problem using a VNS algorithm.

3. Problem Definition and Mathematical Model

The FRP is formally defined on a complete, undirected graph $G = (V, E)$ with node set V, and edge set $E = \{(i, j) : i, j \in V, i < j\}$. The node set $V = \{0, 1, 2, \ldots, n\}$ consists of a supply depot 0, and a set of $1, 2, \ldots, n$ customers (e.g. the petrol stations). Each customer $i \in V$ has a non-negative demand $d_{i,p}$ for some (oil) product $p \in P$. For brevity, we will often denote the demand $d_i \in \mathbb{R}^{|P|}$ of some customer $i \in V$

Parameter/Set	Description
$V = \{0, 1, 2, \ldots, n\}$	Set of stations and depot
$E = \{(i, j)\|i, j \in V, i < j\}$	Set of undirected edges
K	Set of vehicles
P	Set of different products
q_{ip}	Demand of station $i \in V$ for product $p \in P$
Q^k	Capacity of a single compartment of vehicle $k \in K$
d_{ij}	Travel time associated with arc $(i, j) \in E$
C^k	Number of compartment in vehicle $k \in K$
T_k	Trips performed by vehicle k
T	Set of all trips: $\bigcup_{k \in K} T_k$

Table 2: Notation

as a vector, i.e. $d_i = (d_{i,1}, d_{i,2}, \ldots, d_{i,|P|})$. It is assumed that the depot has no demand ($d_0 = \mathbf{0}$). Servicing is performed by a heterogeneous fleet of vehicles K which are stationed at the supply depot. Each vehicle $k \in K$ has C^k compartments of capacity Q^k. For simplicity, we assume that the compartments in a vehicle are of equal size. Between any pair of locations $i, j \in V$, a positive travel time $d_{ij} > 0$ is specified (satisfying the triangle inequality). A summary of notation used throughout this paper is provided in Table 2.

A solution to FRP consists of a set of itineraries, one for each vehicle, which minimizes both makespan and total travel time, while collectively satisfying the demand of all customers. More precisely, a vehicle itinerary describes the order in which a vehicle must visit its customers, the quantity of each product it must load when the vehicle (re-)visits the supply depot, and the quantity of each type of fuel it must deliver during each of the customer visits. We will refer to a product delivery as a customer *visit*. Similarly, a *trip* is defined as the chain of visits performed by a vehicle between its departure from the supply depot and its next return to the supply depot. Note that, since a vehicle is able to resupply, a vehicle might re-visit the same customer during different trips.

Let T_k be the set of trips performed by vehicle $k \in K$. Trips in T_k can be performed in arbitrary order, since each trip starts and ends at the same depot, and there are no temporal constraints imposed on the customer visits. Each trip $t \in T_k$ constitutes an ordered sequence of visits. Each visit v belonging to some trip t is associated with a unique customer $i_v \in V$. A vector $qt_v = (qt_{v1}, qt_{v2}, \ldots, qt_{v|P|})$ specifies the quantity of each product delivered to customer i_v during visit v. The set of trips T is defined as $\bigcup_{k \in K} T_k$.

In the remainder of this paper, we use $\delta(S)$, for some subset of vertices $S \subseteq V$, to denote the set of edges with exactly one endpoint in S. When S is a singleton vertex, we use the shorthand $\delta(i)$ instead of $\delta(\{v_i\})$. The set $E(S)$ denotes the set of edges with both endpoints in S.

We formulate the FRP as a Mixed Integer Linear Program (MILP). For each vehicle $k \in K$, the MILP explicitly models a number of trips $T_k = \{0, 1, 2, \ldots, T_k^{max}\}$, where T_k^{max} is an upper bound on the number of trips vehicle k should make. Here T_k^{max} can be computed by assuming that all deliveries are performed by a single vehicle k, i.e.:

$$T_k^{max} = \left\lceil \frac{\sum_{i\in V}\sum_{p\in P}\left\lceil \frac{q_{ip}}{Q^k}\right\rceil}{C^k} \right\rceil \tag{1}$$

In the MILP formulation below, binary variables x_{ij}^t model whether vehicle $k \in K$ traverses edge $(i, j) \in E$ during trip $t \in T_k$. Similarly, continuous variables v_i^{tp} record the amount of product $p \in P$ delivered during trip t to customer $i \in V$. The number of vehicle compartments dedicated to a product $p \in P$ during a trip t is modeled by integer variables z^{tp}. Finally, binary variables y_i^t are used to identify whether customer i is visited during trip t.

The FRP model is formulated as follows:

$$\min \ \alpha\tau + \beta \sum_{k\in K}\sum_{t\in T_k}\sum_{(i,j)\in E} d_{ij}x_{ij}^t \tag{2}$$

$$\text{s.t. } \tau \geq \sum_{t\in T_k}\sum_{(i,j)\in E} d_{ij}x_{ij}^t \qquad \forall k \in K \tag{3}$$

$$\sum_{j\in V\setminus\{0\}} x_{0j}^t \leq 1 \qquad \forall k \in K, t \in T_k \tag{4}$$

$$\sum_{j\in V, i<j} x_{ij}^t + \sum_{j\in V, j<i} x_{ji}^t = 2y_i^t \qquad \forall k \in K, t \in T_k, i \in V \setminus \{0\} \tag{5}$$

$$\sum_{(i,j)\in E(S)} x_{ij}^t \leq |S| - 1 \qquad \forall S \subseteq N \setminus \{0\}, |S| \geq 2, k \in K, t \in T_k \tag{6}$$

$$\sum_{k\in K}\sum_{t\in T_k} v_i^{tp} = q_{ip} \qquad \forall i \in V, p \in P \tag{7}$$

$$\sum_{i\in V\setminus\{0\}} v_i^{tp} \leq z^{tp}Q \qquad \forall k \in K, t \in T_k, p \in P \tag{8}$$

$$\sum_{p\in P} z^{tp} \leq C \qquad \forall k \in K, t \in T_k \tag{9}$$

$$v_i^{tp} \leq y_i^t QC \qquad \forall k \in K, t \in T_k, p \in P, i \in V \setminus \{0\} \tag{10}$$

$$0 \leq z^{tp} \leq C \qquad \forall k \in K, t \in T_k, p \in P \tag{11}$$

$$0 \leq v_i^{tp} \leq min\{QC, q_{ip}\} \qquad \forall i \in V, k \in K, t \in T_k, p \in P \tag{12}$$

$$x_{ij}^t \in \{0, 1\} \qquad \forall (i, j) \in E, k \in K, t \in T_k \tag{13}$$

$$y_i^t \in \{0, 1\} \qquad \forall i \in V \setminus \{0\}, k \in K, t \in T_k \tag{14}$$

Using non-negative scalars α and β, the weighted objective function (Equation (2)) minimizes makespan as well as total travel time. Constraints (4) track the latest completion time (makespan) of the vehicle itineraries. Constraints (4)-(6) implement the trip structure. A vehicle can depart from the depot at most once during each trip (Constraints (4)). Constraints (5),(6) respectively implement flow preservation and eliminate subtours. Next, Constraints (7)-(9) deal with the vehicle capacities and customer demands. The

	p_1	p_2	p_3
i_1	145	260	260
i_2	0	370	75
i_3	0	295	150
i_4	0	150	110

(a) Demands of the customers

	depot	i_1	i_2	i_3	i_4
depot	0	13	12	12	24
i_1	13	0	8	23	33
i_2	12	8	0	18	26
i_3	12	23	18	0	12
i_4	24	33	26	12	0

(b) Distance Matrix

Table 3: Instance data

demand for each product for each customer must be satisfied (Constraint (7)). Each vehicle compartment can be dedicated to at most one product (Constraint (9)), but product allocations may change between trips. The maximum amount of fuel a vehicle can deliver during a single trip is bounded by the number of compartments dedicated to this type of fuel, multiplied by the compartment capacity (Constraint (8)). From Constraints (8), (9), it follows that the maximum amount of product provided by vehicle k during a single trip cannot exceed Q^kC^k. Finally, Constraints (10) link the v_i^{tkp} and y_i^{tk} variables.

To reduce symmetry in the model, the following constraints are added:

$$\sum_{i\in V}\sum_{j\in V, i<j} d_{ij}x_{ij}^{tk} \geq \sum_{i\in V}\sum_{j\in V, i<j} d_{ij}x_{ij}^{t+1k} \qquad \forall k\in K, t=1,2,\ldots,T_k^{max}-1 \tag{15}$$

These constraints order the trips performed by each vehicle, thereby ensuring that the duration of each consecutive trip is not longer than its preceding trip.

The FRP, as defined by Constraints (2)-(15), belongs to the class of NP-hard problems (reduction to the Capacitated Vehicle Routing Problem). An example problem instance of FRP having 4 customers, 3 products and five 3-compartment vehicles is given in Table 3. Each vehicle compartment has a capacity of 80 units. An optimal solution to this problem instance in which the makespan component is prioritized over the total travel distance is shown in Figure 1a. Observe that each vehicle has a maximum capacity of $80\times 3 = 240$ units, and that some customers require more than 240 units of products. It might be tempting to create dedicated Full Truck Load (FTL) trips for customers who's demand exceeds the capacity of a single vehicle. As an example, we could create an FTL trip for customer i_2 in which a vehicle delivers 240 units of p_2. In the resulting optimization problem we would have to assign the FTL trips to vehicles, as well as any additional trips required to serve the remaining demand of each customer. This approach however leads to suboptimal solutions as is shown by Figure 1b: despite a higher number of FTL trips, the makespan of the solution 1b is longer than the makespan of solution 1a.

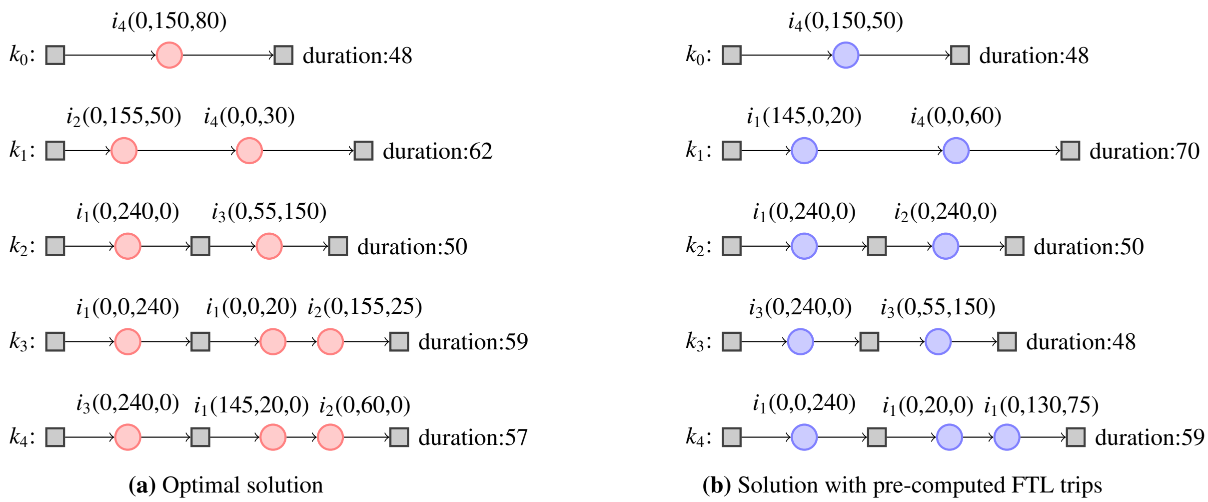


Figure 1: Solutions to the FRP instance given in Table 3

4. Lower Bounds

The model presented in Section 3 explicitly models trips for each vehicle. As a consequence, the performance of this model is strongly correlated to the number of trips allocated to each vehicle. The bound T_k^{max} computed in Equation (1) is typically weak. Computing stronger bounds is non-trivial. Consequently, the scalability of the MILP model is limited.

In order to assess the quality of the ALNS heuristic presented in Section 5 on large problem instances, we propose an alternative mathematical model that can be used to compute strong bounds on the optimal objective value.

For a given vehicle $k \in K$, let T_k be the set of all potential resource feasible trips vehicle k can make. Each trip must visit one or more customers, and start and end at the supply depot. Moreover, let δ_t denote the duration of trip t, v_{ip}^t the amount of product $p \in P$ delivered to customer $i \in V$ during trip t, and let binary parameter e_{ij}^t indicate whether edge $(i, j) \in E$ is traversed in trip t. Using binary assignment variables w_k^t which assign trips to vehicles, we then formulate the FRP as follows:

$$MP:$$

$$\min \alpha\tau + \beta \sum_{t\in T} \sum_{k\in K} \delta_t w_k^t \tag{16}$$

$$\text{s.t. } \tau \geq \sum_{t\in T} \delta_t w_k^t \qquad \forall k \in K \tag{17}$$

$$\sum_{t\in T} \sum_{k\in K} v_{ip}^t w_k^t \geq q_{ip} \qquad \forall i \in V, p \in P \tag{18}$$

$$w_k^t \in \mathbb{Z}_0 \qquad \forall t \in T, k \in K \tag{19}$$

$$\tau \geq 0 \tag{20}$$

Constraints (16) and (17) implement the objective function while Constraints (18) ensure that the demand of each customer is satisfied.

Due to the large number of potential trips for each vehicle, this model cannot be solved directly by an ILP solver. Instead a Column Generation (CG) procedure is developed. Let LPM be the Linear Program Relaxation obtained from model MP by relaxing the integrality conditions on the w_k^t variables (Equation 19). Associate dual variables $\zeta_k \leq 0$, $\eta^{ip} \geq 0$ with constraints (17), (18) respectively. Following a standard CG procedure [e.g. 12] LPM is solved for a small subset of trips $T'_k \subset T_k$ for all vehicles $k \in K$. At every iteration of the CG procedure, a pricing problem is solved for each vehicle $k \in K$ to identify columns with negative reduced cost, i.e. columns for which $\delta_t(\zeta^k - \alpha) + \sum_{i\in V} \sum_{p\in P} v_{ip}^t \eta^{ip} > 0$, which are then added to the set T'_k. A simplified version of the MILP model in Section 3 can be used to solve the pricing problem for a given vehicle $k \in K$; the definitions of the variables remain the same, but their t index is dropped. The pricing problem then becomes:

$$PRICING(k):$$

$$\max \ (\zeta^k - \alpha) \sum_{(i,j)\in E} d_{ij} x_{ij} + \sum_{i\in V\setminus\{0\}} \sum_{p\in P} \eta^{ip} v_{ip} \tag{21}$$

$$\text{s.t. } Eq.\ (4)-(6), (8)-(10) \tag{22}$$

When solving the Master Problem for a homogeneous set of vehicles as opposed to a heterogeneous set, it suffices to solve the pricing problem only once for a single vehicle k' during each iteration of the column generation algorithm, where $k' = \underset{k\in K}{\text{argmax}}\ \zeta^k$. This observation follows directly from the definition of the pricing problem and the domains of the dual variables.

The master problem is strengthened by adding the following Capacity Inequalities:

$$\sum_{k\in K}\sum_{t\in T}\sum_{(i,j)\in\delta(S)} e_{ij}^{t} w_{k}^{t} \geq 2\left\lceil \frac{\sum_{p\in P}\left\lceil \frac{d_p(S)}{Q}\right\rceil}{C}\right\rceil \qquad \forall S\subseteq V\backslash\{0\}, |S|\geq 2 \tag{23}$$

Associating dual variables λ_S with inequalities (23) leads to the following adjusted pricing problem:

$$\delta_t(\zeta^k-\alpha)+\sum_{i\in V}\sum_{p\in P} v_{ip}^{t}\eta^{ip}+\sum_{S\subseteq V\backslash\{0\},|S|\geq 2}\sum_{(i,j)\in\delta(S)} e_{ij}^{t}\lambda_S > 0 \tag{24}$$

This change to the pricing problem can be easily accommodated for in the objective function (Equation (21)) of our pricing problem formulation. Since the set of Capacity Inequalities is exponentially large, it is undesirable to add them all at once to the master problem. Instead, a separation procedure is used to separate violated inequalities. We first run the CG procedure until no more columns with negative reduced cost can be identified. Next the separation procedure is invoked. Whenever violated inequalities are identified, they are added to MP and the CG procedure is resumed. To separate Capacity Inequalities, we first invoke the heuristic route-based algorithm proposed by Archetti et al. [2] (see Section 4.2 in Archetti et al. [2]). If this method fails to identify violated inequalities, we subsequently invoke the computationally more expensive partial enumeration routine by Desaulniers [12] (see 5.2.1 in Desaulniers [12]).

Finally, when no more columns with negative reduced cost or violated inequalities can be identified, an optimal solution to LPM is obtained. Note that, due the relaxation of the integrality conditions of the w_k^t variables, the resulting solution can be fractional. The CG procedure can be embedded in a Branch-and-Price branch-and-price framework in order to guarantee integer feasible solutions. The latter is however outside the scope of this paper. We refer interested readers to Desaulniers [12], Archetti et al. [2], Luo et al. [23]. The optimal solution to LPM constitutes a valid lower bound on the optimal objective value of our FRP.

5. ALNS Heuristic

This section presents an Adaptive Large Neighborhood Search (ALNS) heuristic for the FRP, based on the ALNS framework outlined in Ropke and Pisinger [32], Pisinger and Ropke [30]. Similar to a conventional Large Neighborhood Search (LNS) heuristics [33], ALNS follows a ruin-and-recreate paradigm. However, in contrast to LNS, ALNS relies on multiple destroy and repair operators, which are randomly selected through preferential selection. An overview of our ALNS framework is provided in Alg. 1. During each iteration of the heuristic, a destroy operator is chosen to ruin the current solution, thereby removing visits, trips, or even entire routes from the schedule. Next, a repair operator is selected to repair the (partially) destroyed solution. The destroy/repair procedure is embedded in a local search framework, in our case Simulated Annealing, to determine whether the new solution is accepted as the starting point of the next iteration, or rejected in favor of the old solution. The ALNS heuristic terminates when either a time limit is reached, or a maximum number of non-improving moves have been performed.

The ALNS heuristic is initialized with a feasible solution obtained through a constructive heuristic (Section 5.1). At each iteration of the ALNS heuristic, a destroy and a repair operator are selected based on their past performance. A description of this selection procedure is provided in Section 5.2, followed by an overview of destroy and repair operators (Section 5.3). Each invocation of a destroy operator removes one or more visits from the schedule and adds them to a pool of removed visits. Notice that multiple visits belonging to the same customer can be removed during a single invocation. These visits are added separately to the pool of removed visits; we do not aggregate these visits into a larger visit. After the destroy operator removed a number of visits, a repair operator is selected to re-insert the visits stored in the removal pool back into the schedule. Typically there exist multiple feasible alternatives to re-insert a visit back into the schedule, each leading to a different solution with a different objective value. The repair operator ultimately decides which of these alternatives is selected.

When inserting a visit somewhere in the schedule, we ensure that the resulting (partial) schedule remains feasible. This implies that a visit can only be inserted into an existing trip if this does not exceed the capacity of the vehicle performing the trip. Whenever a visit u is inserted into a trip which already contains a visit v for the same customer, i.e. $i_v = i_u$, we simply merge these visits into one large visit.

We consider three strategies to insert an individual visit into the schedule:

1. Integral Insertion (II): Insert the visit integral into an existing trip.
2. Split Insertion (SI): Split the visit into multiple smaller visits, and insert them into existing trips.
3. Split Trip Insertion (STI): Split an existing trip t into 2 trips t_1, t_2, and assign the visit to either one of the two trips. Splitting a trip is performed by inserting a depot visit in between the sequence of customer visits. If this depot visit is inserted after the last customer visit, then this is equivalent to creating a new empty trip. To limit the number of alternative solutions to be evaluated, we require that the visit that has to be inserted becomes either the last visit in trip t_1, or the first visit in t_2.

In contrast to the II strategy, the SI strategy does not reinsert a visit v back into the schedule; instead a number of smaller visits are inserted such that the amounts of each product supplied by the smaller visits add up to qt_v. In doing so, the SI strategy aims to increase vehicle utilization by claiming left-over capacity of vehicles. Clearly, there are many ways to split a visit into smaller visits and to distribute these over existing trips. Since performing insertions rapidly is essential to the performance of the ALNS heuristic, we cannot afford to exhaustively evaluate all possible split insertions.

Instead, we devised a fast, but inexact approach to compute SI's which does not guarantee to identify any feasible SI even if one exists. An overview of the SI strategy is given in Algorithm 2. Let $c_i(t)$ be the minimum cost incurred when inserting a visit v' for customer i_v into trip $t \in T$, where $c_i(t) = 0$ when trip t already visits customer i_v. Similarly, we define a function $a(t, qt) : (T, \mathbb{R}^{|P|}) \to \mathbb{R}^{|P|}$ which determines a

Algorithm 1: ALNS with SA

Input: Destroy operators Δ, repair operators Ψ, initial temperature T, cooling rate κ, initial solution X_{init}

Output: A feasible solution X_{best}

$X_{current} \leftarrow X_{best} \leftarrow X_{init}$
$it_{nonImp} \leftarrow 0$
repeat
  Select a destroy operator $\delta \in \Delta$ with probability $w_d / \sum_{i=1}^{|\Delta|} w_i$
  Select a repair operator $\psi \in \Psi$ with probability $w_\psi / \sum_{i=1}^{|\Psi|} w_i$
  $X_{new} \leftarrow \psi(\delta(X_{current}))$
  if $obj(X_{new}) < obj(X_{current})$ **then**
    $it_{nonImp} \leftarrow 0$
    $X_{current} \leftarrow X_{new}$
    if $obj(X_{current}) < obj(X_{best})$ **then**
      $X_{best} \leftarrow X_{current}$
  else
    $it_{nonImp} \leftarrow it_{nonImp} + 1$
    Generate a random number $\epsilon \in [0, 1]$
    if $\epsilon < e^{-(obj(X_{new}) - obj(X_{current}))/T}$ **then**
      $X_{current} \leftarrow X_{new}$
  $T \leftarrow \kappa T$
  Update operator probabilities
until *timeLimit or* $it_{nonImp} \geq nonImp_{max}$

vector $qt_{v'}$, $\mathbf{0} \leq qt_{v'} \leq qt$, specifying the quantity of products that would be supplied to customer i_v if a visit v' for customer i_v were to be inserted into trip t. Alg. 2 proceeds as follows. At each iteration, trips are ranked based on the relative cost $\frac{c_{i_v}(t)}{|a(t,qt)|}$ of inserting a visit for customer i_v into trip t. The trip with the lowest relative cost is selected and a new visit v' for customer i_v is inserted. This process is repeated until visits equivalent to a supply of qt_v are inserted, or the set of trips with residual capacity is exhausted. In the latter case, the operator fails to split delivery v. Note that, to prevent too much fragmentation, the SI operator is only allowed to insert visits into trips which cannot be used by the II operator (trips that do not have enough residual capacity to serve qt_v entirely).

A precise definition of the function $a(t, qt)$ is provided in Alg. 3. The function first attempts to fill up any partially filled compartments of the vehicle serving trip t. Next, a greedy procedure is used to fill up any unused compartments of the same vehicle. Preference is given to products for which there is a large demand and for which there is little spare capacity in any of the other trips (line 5).

5.1. Initial Solution

An initial feasible solution X_{init} for the ALNS heuristic is obtained through a simple constructive procedure. This procedure works in two separate phases:

1. Generate a set of vehicle-independent trips T such that the trips collectively satisfy the demand of all customers.

Algorithm 2: Split insertion - splits a visit into multiple smaller visits

Input: visit v for some customer $i_v \in V$

$cost \leftarrow 0$ ▷ total cost incurred by splitting visit v
$qt \leftarrow qt_v$
$T^s = \{t \in T : \mathbf{0} < a(t, qt) < qt\}$
while $qt > \mathbf{0}$ *and* $T^s \neq \emptyset$ **do**
  $t' = \underset{t \in T^s}{\operatorname{argmin}} \frac{c_{i_v}(t)}{|a(t,qt)|}$ ▷ trip with lowest weighted insertion cost
  $cost \leftarrow cost + c_{i_v}(t')$
  $insert(v', t')$ and set $qt_{v'} = a(t', qt)$, $i_{v'} = i_v$
  $qt \leftarrow qt - qt_{v'}$
  $T^s = \{t \in T^s : \mathbf{0} < a(t, qt) \leq qt\}$
if $qt > \mathbf{0}$ **then**
  fail()

2. Assign each trip $t \in T$ to a vehicle.

When vehicles are heterogeneous, we only generate trips which meet the capacity requirements of the smallest vehicle.

The process of generating trips is subdivided into three steps, which attempt to generate trips with high vehicle utilization. The heuristic first generates dedicated trips for each customer $i \in V$ independently, which utilize the maximum capacity ($C \times Q$) of each vehicle. Clearly, for a given customer $i \in V$, there exist $\left\lfloor \frac{\sum_{p \in P} \lfloor q_{ip}/Q \rfloor}{C} \right\rfloor$ such trips.

Let $\overline{q_i}$, $i \in V$, be a vector representing for each product the remaining, unsatisfied demand of a customer. After generating the Full Truck Load (FTL) trips, for a given customer $i \in V$, it follows that: $|\{p \in P : \overline{q_{ip}} \geq Q\}| < C$, and $\sum_{\substack{p \in P: \\ \overline{q_{ip}} \geq Q}} \overline{q_{ip}} < CQ$. Next, a number of dedicated, Less than Truck Load (LTL) trips for each customer are derived. We require that for each dedicated LTL trip, each vehicle compartment must hold some product, and that precedence is given to products for which the customer has the largest residual demand. As an example, let $C = 3$, $Q = 80$, and $d_i = (100, 20, 50)$ for some customer i. We create a dedicated LTL trip for customer i containing a single visit v with $qt_v = (100, 0, 50)$. The residual demand of customer i becomes $\overline{q_i} = (0, 20, 0)$. Again, for a single customer $i \in V$, there exist $\lfloor \frac{\sum_{p \in P} \lceil q_{ip}/Q \rceil}{C} \rfloor$ such LTL trips.

After updating $\overline{q_i}$ with the LTL trips for each customer $i \in V$, the following conditions hold: $|\{p \in P : \overline{q_{ip}} > 0\}| < C$ and $\sum_{p \in P} \overline{q_{ip}} < CQ$, i.e. the residual demand of customer i can be satisfied by a single vehicle. In the third, and last step of the trip generation phase, for each of the customers $i \in V$ with residual demand, we create a single visit v having $qt_v = \overline{q_i}$. Next, these visits are greedily inserted into the existing trips, following the insertion strategies outlined in the previous section, thereby always selecting the strategy that increases the objective the least.

Once all trips have been generated, we must assign them to vehicles. This assignment problem is solved by a greedy heuristic which allocates trips one-by-one, to the vehicle with the shortest itinerary, with ties

Algorithm 3: Implementation of $a(\tau, qt)$

Input: demand vector $qt \in \mathbb{R}^{|P|}$, trip $\tau \in T$

Output: vector $sv \in \mathbb{R}^{|P|}$, $\mathbf{0} \leq sv \leq qt$: quantity of products that would be supplied to customer i_v if a new visit v' for customer i_v were to be inserted into trip τ

▷ Fill up partially filled compartments. Let res_{tp} be the total unused capacity of *non-empty* compartments holding product $p \in P$ of the vehicle serving trip $t \in T$.

$sv = [\min\{res_{\tau p}, qt_p\}]_{p \in P}$

$\overline{qt} \leftarrow qt - sv$

▷ Allocate products to empty compartments. Let c_τ be the number of empty compartments of the vehicle serving trip τ.

$c' \leftarrow c_\tau$

while $c' > 0$ **do**

  $p' = \underset{p \in P}{\text{argmax}}\ \overline{qt}_p - \sum_{t \in T \setminus \tau} res_{tp}$

  if $\overline{qt}_p > 0$ **then**

    $sv_{p'} = sv_{p'} + \min\{Q^k, \overline{qt}_{p'}\}$

    $\overline{qt}_{p'} = \min\{0, \overline{qt}_{p'} - Q^k\}$

    $c' \leftarrow c' - 1$

return sv

broken arbitrarily. In case the vehicles are heterogeneous, a trip is only assigned to vehicle if that vehicle meets the capacity requirements to perform that trip.

5.2. Operator selection

As is conventional in ALNS, the destroy (resp. repair) operators are selected with a probability proportional to their past performance. Let Γ be the set of available operators (e.g. destroy operators), and w_γ a weight associated with each operator $\gamma \in \Gamma$. The probability that an operator $\gamma \in \Gamma$ is selected is set equal to $w_\gamma / \sum_{i \in \Gamma} w_i$.

The ALNS search is partitioned into segments of 100 consecutive iterations. Throughout a segment, the performance of each operator is monitored, and, at the end of the segment the weights of the operators are adjusted according to their performance. Each time an operator is invoked during an ALNS iteration, it is able to collect a reward; a summary of potential rewards and their eligibility criteria is given in Table 4. Let π_γ be the total reward accumulated by operator γ during a segment, and β_γ the number of times the operator was invoked during the segment. At the end of each segment, the normalized performance of an operator is given by $\frac{\pi_\gamma}{\beta_\gamma}$, and a new weight w'_γ for each operator $\gamma \in \Gamma$ is calculated as follows:

$$w'_\gamma = w_\gamma(1 - \xi) + \xi \frac{\pi_\gamma}{\beta_\gamma} \tag{25}$$

,where scalar ξ ($0 \leq \xi \leq 1$) controls how sensitive the weights react to changes in the operator's performance. In our implementation, at the beginning of the ALNS search, all operators have equal weight.

Moreover, destroy and repair operators are selected independently from each other.

Table 4: Adaptive adjustment of the operator scores

Reward	Value	Eligibility criteria
σ_1	5	The last ALNS iteration produced a new best solution
σ_2	3	The last ALNS iteration produced a solution that was better than the solution from the previous iteration
σ_3	1	The solution from the last ALNS iteration is worse than the solution from the previous iteration, but was nevertheless accepted

5.3. Destroy and Repair Operators

Six removal operators are used, the first three of them are directly adopted from Shaw [33], Ropke and Pisinger [32], Pisinger and Ropke [30]; the remaining 3 operators are tailored to our problem. To keep this paper self-contained, we include a brief description of each operator. For more details, we refer to the aforementioned references.

- Random Removal (RR): randomly remove $\phi \in [1, \rho|V|]$ visits from the schedule.

- Worst Removal (WR): Iteratively remove $\phi \in [1, \rho|V|]$ visits. Each removal yields a new schedule. In every iteration, we remove the visit who's removal from the schedule will benefit the objective value the most. Ties are broken arbitrarily.

- Shaw Removal (SR): Remove similar visits, where the similarity between a pair of visits u, v, $u \neq v$, is expressed by a similarity metric $s(u, v)$. Let $l_{uv} = -1$ if visits u, v are performed during the same trip, $l_{uv} = 1$ otherwise. We define $s(u, v) = \varphi d_{i_v i_u} + \psi l_{vu} + \chi \|qt_u - qt_v\|_1$, where φ, ψ, χ are independent scalars, and where the terms $d_{i_v i_u}$, resp. $\|qt_u - qt_v\|_1$ have been normalized by resp. the maximum distance between a pair of customers, and a vector representing the maximum demand for each product. The SR operator iteratively removes $\phi \in [1 : \rho|V|]$ visits from the schedule, each time selecting the visit v^* which is most similar to the visit u removed in the preceding iteration, i.e. $v^* = \underset{v}{\text{argmin}}\ s(u, v)$. The first visit to be removed is selected at random.

- Longest Route Removal(LRR): remove the longest route from the schedule.

- Visits in Long Routes Removal(VLRR): randomly remove $\phi \in [1, 0.5V_{num}]$ visits from the $\eta = 0.5|K|$ longest routes, where V_{num} is the total number of visits in these routes.

Next to the above removal operators, we use one additional operator which does not remove visits, but reallocates trips in their entirety:

- Trip Reallocation (TR): Randomly select and remove $\zeta \in [1 : 0.45|T|]$ trips from the schedule. Sort the removed trips in descending order based on their duration, and iteratively re-insert them back into the schedule, each time assigning a trip to the vehicle with the shortest itinerary.

Note that, unlike the other removal operators, the TR operator is not followed by one of the insertion operators described below.

To repair the (partially) destroyed schedules, four repair operators are used to reschedule visits that have been removed by the destroy operators:

- Greedy Insertion (GI): For each of the unscheduled visits in the removal pool, this operator computes the lowest cost to re-insert the visit somewhere in the schedule using the II, SI and STI insertion strategies. Next, the operator inserts the visit with the lowest insertion cost into the schedule, and updates the insertion costs of the remaining unscheduled visits accordingly. This procedure is repeated until all visits have been rescheduled.

- Greedy with Split Preference (GSP): Same as GI, but instead of considering all insertion strategies, this operator attempts to insert each visit using SI. Only in the event that a visit cannot be inserted through SI, e.g. when the existing trips do not have enough residual capacity, the operator resorts to the II and STI strategies.

- Regret Insertion [32] (RI-2, RI-3): For each unscheduled visit v, we compute the cheapest option f_v^1 to reschedule visit v through the 3 different insertion strategies, as well as the ith cheapest alternative f_v^i for when options $f_j^1, \dots, f_v^{i-1}$ are unavailable. For each unscheduled visit v, a so-called regret cost Δ_v^i is computed, where Δ_v^i is the difference in cost between f_v^1 and f_v^i. At each iteration, the operator greedily selects the visit v with lowest cost Δ_v^i, reschedules it using f_v^1, and updates f_u^j for $j = 1, \dots, i$ for the remaining unscheduled visits accordingly. We include 2 different Regret Insertion operators, RI-2 and RI-3, for $i = 2$ and $i = 3$ respectively.

6. Experiments

In this section, we perform an experimental evaluation of the proposed ALNS heuristic based on a large testbed of FRP instances derived from real-world data. All experiments are conducted on a system having an Intel Core i5-3320M 2.67 GHz Processor, 8 GB RAM, and running Windows 10 Professional. The algorithms are implemented in Java 8, the code can be downloaded from `HiddenUntilFinalVersion`. The MILP models are solved through IBM ILOG CPLEX 12.8 using default settings and a time-limit of 7200 seconds. An overview of the parameters used in our ALNS implementation is provided in Table 5. These parameters are determined empirically. The scalars α, β (Equation (2)) in the objective function are fixed to 100 and 1 respectively.

6.1. Problem instances

Computational experiments are performed on a set of 112 instances, which are partitioned into 64 *small* and 48 *large* instances. Each instances specifies a set of customers, their locations, a demand per product per customer, the number of available vehicles, the number of compartments in a vehicle, and the volume of

Table 5: The parameters of ALNS

Notation	Description	Value
$nonImp_{max}$	Maximum number of non-improving iterations	1000
$timeLimit$	Time limit (in seconds)	3600
T	Initial temperature SA	200
κ	Cooling rate SA	0.9995
Nsg	Number of iterations per segment	100
ξ	Reaction factor	0.1
ρ	Destroy rate for removal operators	0.45
φ	First Shaw parameter	0.2
ψ	Second Shaw parameter	0.25
χ	Third Shaw parameter	0.1

a compartment. The *small* instances have up to 16 customers, up to 5 vehicles each having 3 compartments with a fixed volume of 8k units, and up to 3 different petrol products. The *large* instances on the other hand have up to 60 customers which are served by up to 20 vehicles having the same number of compartments as in the small instances, but with capacities of 8k or 16k units.

To generate realistic benchmark instances, we used data from NingBo Chaoyang Oil Transportation Co., Ltd., a Chinese petroleum transportation company. The customer demands for each product have been sampled from their customer delivery data. Similarly, the vehicle sizes (number of compartments and compartment dimensions) are based on vehicles used by the company. Unfortunately, the company was unable to provide us with the actual locations of the fuel stations (customers), or the travel distances between each pair of locations. Instead we were provided with the travel distance r_i between the depot and each customer location $i \in V$. To mimic actual travel data, we assigned each customer i a location relative to the depot by a polar coordinate (r_i, ϕ_i), where ϕ_i is a random angle uniformly selected from $[0, 360[$. The distance d_{ij} between a pair of locations $i, j \in V$ is given by the Euclidean distance, rounded down to the nearest integer. By computing the distance matrix in this manner, we preserve the travel distances r_i, while simultaneously guaranteeing that the distance matrix satisfies the triangle inequality.

In what follows, we use the following naming convention to uniquely identify problem instances: sα-vβ-pγ-cδ-ϵ, where α denotes the number of customers, β the number of vehicles, γ the number of products, δ the number of compartments per vehicle and ϵ the capacity of a single compartment.

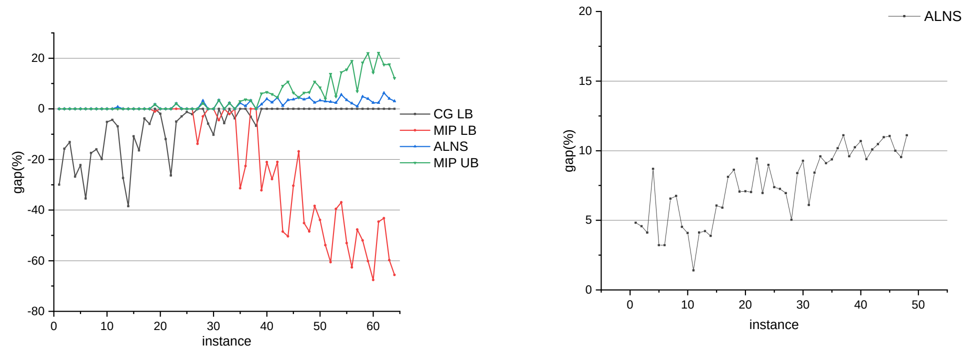
6.2. *Performance of ALNS*

In this section, we analyze the performance of the ALNS heuristic by comparing it against the bounds produced by the MILP model from Section 3 and the CG model from Section 4. We solve the set of *small* instances with the MILP model, the CG model and the ALNS heuristic. The *large* instances are solved through CG and ALNS since these instances are too large to solve through the MILP formulation. Detailed results for each instance is found in Tables A.6 and A.7 in the Appendix. Column LB provides the *strongest* lower bound derived from the CG and, when available, the MILP model. For ALNS, we report the best f_b, average f_a and worst f_w solution observed in 10 independent invocations of the ALNS algorithm, as well as the average computation time $t(s)$. Column *gap* reports the percentage gap between f_b and LB, calculated as $\frac{f_b - LB}{f_b}$. For MILP, we report the best solution f obtained within a time limit of 2h, the percentage gap between f and LB, and the computation time $t(s)$ in seconds. Finally for CG we report the best objective value f and the computation time $t(s)$ in seconds. All instances are ordered in ascending order of the number of customers, number of vehicles, number of products, number of compartments and compartment size.

As can be observed from Table A.6, the MILP model can only solve instances up to 10 customers within a time limit of 2h. Out of the 64 small instances, MILP solves 18 instances to proven optimality. The same optimal solutions are found by our ALNS heuristic. For the instances with 12 or more customers, MILP never terminates before the 2h time limit. The ALNS heuristic however is able to find high quality solutions (average gap 1.7%) for all small instances in less than 10 seconds per instance. When comparing the best ALNS solutions $f(b)$ with the MILP solutions f we observe that ALNS consistently outperforms MILP as the ALNS solutions are either identical or better than the MILP solutions for all small instances. To assess the scalability of the ALNS heuristic, we repeat these computational experiments on the set of medium instances (Table A.7). For these instances, we can only compare against the bounds obtained from our CG model. Even for the largest instances in this set, having 60 customers and 20 vehicles, computation times of the ALNS heuristic stay well below 25 minutes, while obtaining an average gap ($\frac{f_b - LB}{f_b}$) of 7.5%. When expressing the gap between LB and average objective value f_a (as opposed to f_b), we witness only a minor gap increase of 1.3%. This shows that the ALNS results are robust across multiple runs.

Figure 2a depicts a summary of the bounds computed for each of the instances in the *small* data set. The 0% gap line corresponds to the LB column in Table A.6, i.e. the strongest lower bound. The blue line (ALNS) and green line (MILP UB) correspond to the gap(%) columns in the same table. Finally, the black line (CG LB) and the red line (MILP LB) represent the gaps between these two lower bounds and the strongest lower bound (0% line). For instance, the black line is calculated as $-\frac{LB - CG}{LB}$. As can be observed from Figure 2, for the first 18 instances the MILP model produces the strongest lower bounds. Once the instances become too large (12 customers or more), the MILP model is no longer able to derive strong bounds: for instances 40-64 the best lower bound is provided by the CG model. Figure 2a depicts the gaps computed between ALNS and CG for the large instances. Even for the largest instances in this set, the gap stays well below 12%.

Figure 3 analyses various properties of the solutions obtained through the ALNS heuristic. $|Cap|$ is the total vehicle capacity (sum of compartment capacities). Figure 3a shows the average number of trips performed by a vehicle. This number ranges from 0.5 to 10.5, and tends to grow linear in the number of customers. The slope of these curves is proportional to the number of available vehicles, and their capacities. Figure 3b depicts the average number of customers visited during a single trip. Clearly this number is limited by the vehicle capacity. In the FRP the average number of customers per trip is quite



(a) Small instances **(b)** Large instances

Figure 2: Gap comparison

low due to the fact that customer demands are high compared to the vehicle capacities. Finally Figure 3c measures the vehicle utilization, computed as the average fraction of the vehicle capacity occupied during a trip. Clearly, the ALNS heuristic is able to compute very dense solutions, with average utilization rates of up to 97%. The average utilization rate increases inversely proportional to the total vehicle capacity, and is relatively independent of the number of available vehicles.

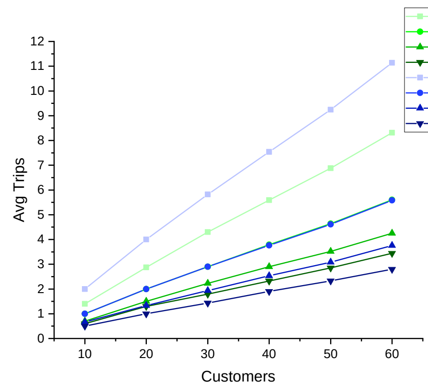
6.3. Computation times

Figure 4 shows the influence of the various instance features on the average ALNS computation times. Naturally, computation times go up when the number of customers increases. For a fixed number of customers, we observe a *decrease* in computation time when the number of vehicle compartments or their capacity increases (Figure 4b,4c). A strong *increase* in computation time is observed when the number of different products increases (Figure 4d). Finally, in contrast to the aforementioned instance features, we do not witness a strong correlation between the number of vehicles and the total computation time of the ALNS heuristic (Figure 4a).

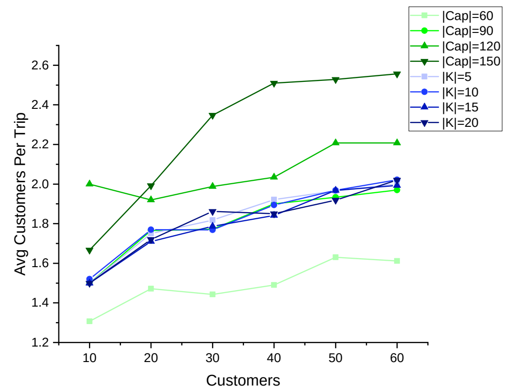
6.4. Impact of compartment splitting

Individual compartments of fuel delivery vehicles are typically equipped with debit meters to deliver precise quantities of fuel to customers. Moreover, it becomes possible to split the fuel stored into a *single* compartment over multiple customers. From an operational as well as a computational perspective, allowing compartment splitting increases the complexity of the problem due to a drastic increase in search space. A reasonable question to ask is how beneficial it is to allow compartment splitting? One might conjecture that there is a diminishing effect to compartment splitting when the number of customers increases due to an increase in the number of potential routes and delivery patterns.

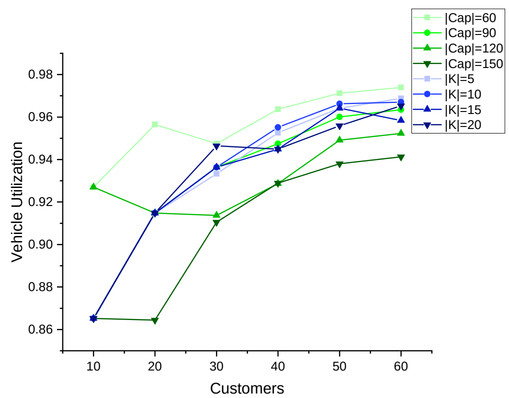
A comparison of the results obtained through ALNS *with* and *without* compartment-splitting over multiple customers on a subset of 32 instances containing 20-40 customers is presented in Figure 5. As is



(a) Average number of trips per vehicle.



(b) Average number of customers visited per trip.



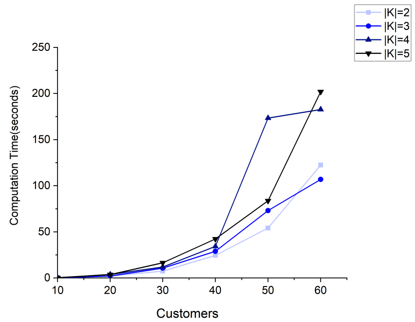
(c) Average vehicle capacity utilization.

Figure 3: Solution statistics.

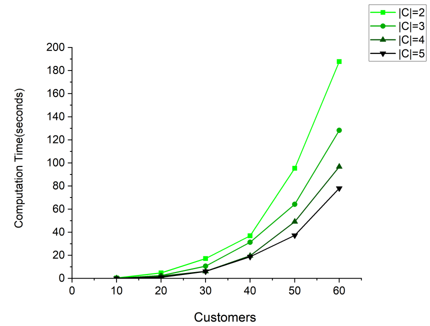
shown in the figure, the objective value of all 32 instances improves when compartment-splitting is allowed. The average improvement in objective value equals 12.5%. Moreover, we observed a marginal increase in the average objective improvement obtained from splitting compartments when the number of customers increases.

6.5. ALNS operator performance

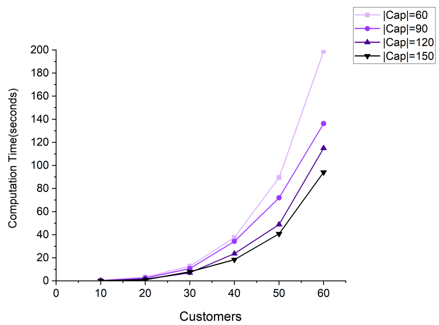
Figure 6 investigates the performance of the ALNS operators described in Section 5.3. This analysis is performed using average results obtained from 16 instances, each having 40 customers, and which have been solved 10 times per instance. The x-axis of each graph shows the progress of the heuristic: the heuristic starts at 0% 'completion', and finishes at 100%. This completion metric allows us to compare different ALNS runs mutually, independent of the number of iterations performed by the time the heuristic terminates. Figure 6a depicts the cumulative contribution of each operator, expressed in the number of times an operator discovered a new incumbent solution and normalized by the total number of times a new incumbent solution was found (percentage). Solid lines represent destroy operators, whereas dashed lines are the repair operators. Naturally, the largest number of improving solutions are discovered early in the



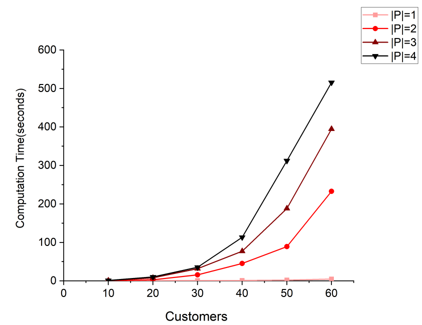
(a) Number of vehicles



(b) Number of vehicle compartments



(c) Compartment capacity



(d) Number of products

Figure 4: Impact of instance features on average ALNS computation time.

search. Towards the end of the search, the graph gradually flattens out. The SR destroy operator (Shaw Removal, green) and the RI-3 repair operator (Remove-Insert, brown) are responsible for the discovery of the largest number of new best solutions.

Recall from Section 5.2 that during each ALNS iteration, the destroy and repair operators are selected proportional to their normalized performance $w_\gamma / \sum_{i\in\Gamma} w_i$, and that the performance of an operator $\gamma \in \Gamma$ is measured in terms of collected rewards (Table 4). The graphs in Figure 6b and 6c plot for each operator $\gamma \in \Gamma$ the average ratio $w_\gamma / \sum_{i\in\Gamma} w_i$ over the course of the heuristic. Interesting to observe is that although percentage wise the SR removal operator discovers the largest number of incumbent solutions (Figure 6a), the performance of this operator degrades over time (Figure 6b) and is surpassed by the LRR (black) and VLRR operators (pink). The RI-2, RI-3 repair operators consistently outperform the GI and GSP operators (Figure 6c). Finally we observe from Figures 6a, 6b that the performance of the Worst-Removal (WR, green) operator is relatively poor. Also in the work by Ropke and Pisinger [32], this operator has the lowest performance. To investigate whether it would be better to disable the WR operator altogether, we reran our experiments. We observed that disabling the WR operator benefits some instances, but negatively affects other instances. We suspect that this operator is useful in some cases to escape local minima. Therefore, we decided to leave the operator in, and let the operator selection mechanism (Section 5.2) deal with it.

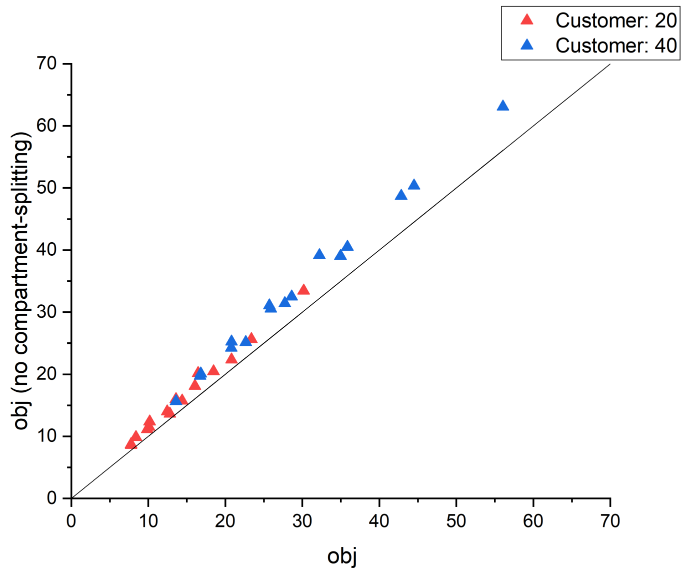


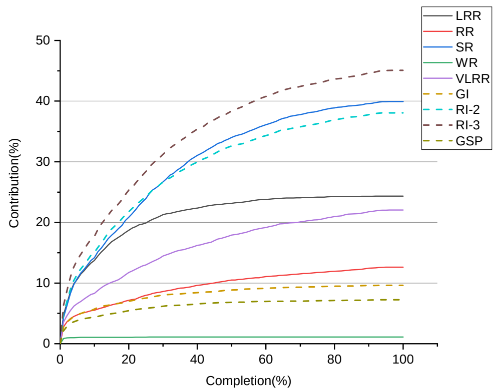
Figure 5: Impact of compartment splitting. Each triangle represents an instance from the *large* data set; the triangle colors correspond to the number of customers in the instance. A slight increase in the benefit obtained through compartment splitting can be observed when the number of customers increases.

7. Conclusions

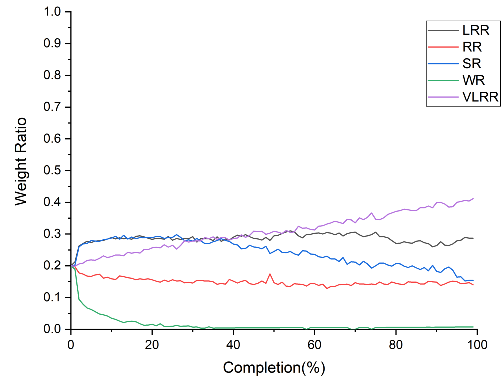
In this paper, we addressed a Fuel Replenishment Problem (FRP). Specifically, we considered a split delivery vehicle routing problem with multiple compartments and multiple trips. The objective is to determine the routing of the vehicles, the delivery pattern of each trip and the allocation of products to vehicle compartments, while minimizing the makespan of the resulting routes. We formally defined the problem and proposed a MILP model for it. In order to solve this problem effectively, we proposed an Adaptive Large Neighborhood Search (ALNS) algorithm. Additionally, we also proposed a column generation approach to compute tight lower bounds.

We generated small and large-sized instance sets to evaluate the performance of ALNS. We showed that the ALNS is able to find optimal solutions much faster than the exact MILP model using CPLEX. Moreover, the ALNS heuristic can handle instances that are significantly larger: we obtain very good solutions within 1000 seconds for a large instance with 60 customers, 20 vehicles, 3 products and 4 compartments. Besides, we did sensitivity analysis on the impact of important problem instance feature (vehicles, products, compartments and their capacity). Our numerical results show that significant savings are obtained by allowing for multiple trips and splitting customer demand.

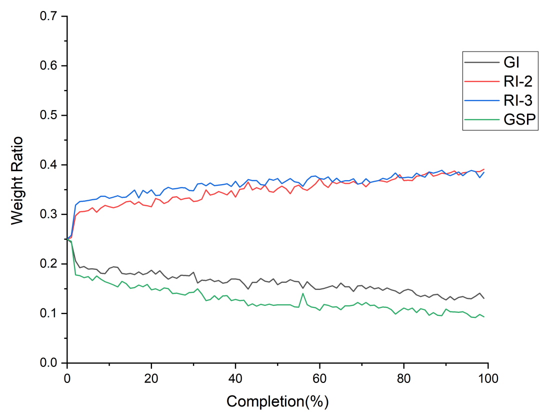
Future research could involve extending our models and solution approach to consider stochastic demands. In this case, Inventory-Routing type of models are highly relevant to consider.



(a) Cumulative contribution of each operator.



(b) Weight ratio of destroy operators



(c) Weight ratio of repair operators

Figure 6: ALNS operator performance, averaged over 10 invocations of the ALNS heuristic over 16 instances with 40 customers each.

Acknowledgements

This research was supported in part by the Netherlands Organisation for Scientific Research (NWO) grant 629.002.211.

Reference

[1] Abdelaziz, F. B., Roucairol, C., Bacha, C., 2002. Deliveries of liquid fuels to sndp gas stations using vehicles with multiple compartments. In: IEEE international conference on systems, man and cybernetics. Vol. 1. IEEE, pp. 478–483.

[2] Archetti, C., Bouchard, M., Desaulniers, G., 2011. Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. Transportation Science 45 (3), 285–298.

[3] Archetti, C., Speranza, M. G., Hertz, A., 2006. A tabu search algorithm for the split delivery vehicle routing problem. Transportation science 40 (1), 64–73.

[4] Avella, P., Boccia, M., Sforza, A., 2004. Solving a fuel delivery problem by heuristic and exact approaches. European Journal of Operational Research 152 (1), 170–179.

[5] Benantar, A., Ouafi, R., Boukachour, J., 2016. A petrol station replenishment problem: new variant and formulation. Logistics Research 9 (1), 6.

[6] Coelho, L. C., Laporte, G., 2015. Classification, models and exact algorithms for multi-compartment delivery problems. European Journal of Operational Research 242 (3), 854–864.

[7] Cornillier, F., Boctor, F., Renaud, J., 2012. Heuristics for the multi-depot petrol station replenishment problem with time windows. European Journal of Operational Research 220 (2), 361–369.

[8] Cornillier, F., Boctor, F. F., Laporte, G., Renaud, J., 2008. An exact algorithm for the petrol station replenishment problem. Journal of the Operational Research Society 59 (5), 607–615.

[9] Cornillier, F., Boctor, F. F., Laporte, G., Renaud, J., 2008. A heuristic for the multi-period petrol station replenishment problem. European Journal of Operational Research 191 (2), 295–305.

[10] Cornillier, F., Laporte, G., Boctor, F. F., Renaud, J., 2009. The petrol station replenishment problem with time windows. Computers & Operations Research 36 (3), 919–935.

[11] Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., Vogel, U., 2011. Vehicle routing with compartments: applications, modelling and heuristics. OR spectrum 33 (4), 885–914.

[12] Desaulniers, G., 2010. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Operations research 58 (1), 179–192.

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

[14] Dror, M., Trudeau, P., 1990. Split delivery routing. Naval Research Logistics (NRL) 37 (3), 383–402.

[15] El Fallahi, A., Prins, C., Calvo, R. W., 2008. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Computers & Operations Research 35 (5), 1725–1741.

[16] Feillet, D., Dejax, P., Gendreau, M., Gueguen, C., 2006. Vehicle routing with time windows and split deliveries. Technical Paper 851.

[17] Henke, T., Speranza, M. G., Wäscher, G., 2015. The multi-compartment vehicle routing problem with flexible compartment sizes. European Journal of Operational Research 246 (3), 730–743.

[18] Hübner, A., Ostermeier, M., 2018. A multi-compartment vehicle routing problem with loading and unloading costs. Transportation Science.

[19] Jin, M., Liu, K., Bowden, R. O., 2007. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. International Journal of Production Economics 105 (1), 228–242.

[20] Lahyani, R., Coelho, L. C., Khemakhem, M., Laporte, G., Semet, F., 2015. A multi-compartment vehicle routing problem arising in the collection of olive oil in tunisia. Omega 51, 1–10.

[21] Lee, C.-G., Epelman, M. A., White III, C. C., Bozer, Y. A., 2006. A shortest path approach to the multiple-vehicle routing problem with split pick-ups. Transportation research part B: Methodological 40 (4), 265–284.

[22] Lihua, S., 2012. Study on the logistics optimization informationization of oil products in petrochemical enterprises. Computers and Applied Chemistry 29 (5), 620–624.

[23] Luo, Z., Qin, H., Zhu, W., Lim, A., 2016. Branch and price and cut for the split-delivery vehicle routing problem with time windows and linear weight-related cost. Transportation Science 51 (2), 668–687.

[24] Melechovskỳ, J., 2013. A variable neighborhood search for the selective multi-compartment vehicle routing problem with time windows. Lecture Notes in Management Science 5, 159–166.

[25] Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., Velasco, N., 2010. A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. Computers & Operations Research 37 (11), 1886–1898.

[26] Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., Velasco, N., 2011. Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. Transportation Science 45 (3), 346–363.

[27] Muyldermans, L., Pang, G., 2010. On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. European Journal of Operational Research 206 (1), 93–103.

[28] Ng, W. L., Leung, S., Lam, J., Pan, S., 2008. Petrol delivery tanker assignment and routing: a case study in hong kong. Journal of the Operational Research Society 59 (9), 1191–1200.

[29] Oppen, J., Løkketangen, A., 2008. A tabu search approach for the livestock collection problem. Computers & Operations Research 35 (10), 3213–3229.

[30] Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers & operations research 34 (8), 2403–2435.

[31] Popović, D., Vidović, M., Radivojević, G., 2012. Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. Expert Systems with Applications 39 (18), 13390–13398.

[32] Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation science 40 (4), 455–472.

[33] Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: International conference on principles and practice of constraint programming. Springer, pp. 417–431.

[34] Sierksma, G., Tijssen, G. A., 1998. Routing helicopters for crew exchanges on off-shore locations. Annals of Operations Research 76, 261–286.

[35] Vidović, M., Popović, D., Ratković, B., 2014. Mixed integer and heuristics model for the inventory routing problem in fuel delivery. International Journal of Production Economics 147, 593–604.

AppendixA. Detailed computational results

Table A.6: Small instances

		ALNS					MILP			CG	
instance	LB	f_b	f_a	f_w	*gap*	$t(s)$	f	*gap*	$t(s)$	f	$t(s)$
s2-v3-p2-c3-8000	4900.0	**4900**	4900	4900	0.0	0.2	**4900**	0.0	4.6	3433.3	0.1
s2-v3-p2-c4-8000	3383.0	**3383**	3383	3383	0.0	0.2	**3383**	0.0	3.3	2849.7	0.1
s2-v3-p3-c3-8000	5335.0	**5335**	5335	5335	0.0	0.2	**5335**	0.0	2.4	4635.0	0.1
s2-v3-p3-c4-8000	5109.0	**5109**	5109	5109	0.0	0.1	**5109**	0.0	3.9	3742.3	0.2
s2-v5-p2-c3-8000	2700.0	**2700**	2700	2700	0.0	0.1	**2700**	0.0	2.4	2100.0	0.1
s2-v5-p2-c4-8000	2700.0	**2700**	2700	2700	0.0	0.0	**2700**	0.0	2.5	1743.0	0.1
s2-v5-p3-c3-8000	3435.0	**3435**	3435	3435	0.0	0.1	**3435**	0.0	2.9	2835.0	0.1
s2-v5-p3-c4-8000	2726.0	**2726**	2726	2726	0.0	0.1	**2726**	0.0	2.7	2289.0	0.1
s4-v3-p2-c3-8000	7372.0	**7372**	7372	7372	0.0	0.1	**7372**	0.0	6.7	5905.3	0.2
s4-v3-p2-c4-8000	5249.0	**5249**	5249	5249	0.0	0.1	**5249**	0.0	2.6	4978.3	0.3
s4-v3-p3-c3-8000	9875.0	**9875**	9875	9875	0.0	0.2	**9875**	0.0	320.7	9441.7	0.4
s4-v3-p3-c4-8000	7339.0	**7399**	7459	7599	0.8	0.1	**7339**	0.0	5.5	6832.3	0.3
s4-v5-p2-c3-8000	4972.0	**4972**	4972	4972	0.0	0.1	**4972**	0.0	6.7	3612.0	0.2
s4-v5-p2-c4-8000	4949.0	**4949**	4949	4949	0.0	0.1	**4949**	0.0	4.3	3045.0	0.3
s4-v5-p3-c3-8000	6476.0	**6476**	6555.9	7275	0.0	0.1	**6476**	0.0	242.5	5775.0	0.4
s4-v5-p3-c4-8000	5003.0	**5003**	5023.2	5205	0.0	0.1	**5003**	0.0	6.2	4179.0	0.6
s6-v3-p2-c3-8000	7410.0	**7410**	7410	7410	0.0	0.1	**7410**	0.0	19.1	7129.7	0.6
s6-v3-p2-c4-8000	6375.0	**6375**	6375	6375	0.0	0.1	**6375**	0.0	9.0	5995.3	0.5
s6-v3-p3-c3-8000	11126.9	**11327**	11378.1	11534	1.8	0.2	**11327**	1.8	7204.8	11126.9	1.0
s6-v3-p3-c4-8000	8649.0	**8649**	8649.9	8650	0.0	0.1	**8649**	0.0	426.9	8480.3	2.3
s6-v5-p2-c3-8000	5010.0	**5010**	5010	5010	0.0	0.1	**5010**	0.0	72.0	4410.0	0.8
s6-v5-p2-c4-8000	4979.0	**4979**	4979	4979	0.0	0.1	**4979**	0.0	123.9	3667.1	0.7
s6-v5-p3-c3-8000	7083.6	**7236**	7275.6	7335	2.1	0.2	**7236**	2.1	7210.1	6727.0	1.1
s6-v5-p3-c4-8000	5347.0	**5349**	5409.2	5550	0.0	0.1	**5347**	0.0	397.3	5187.0	2.9
s8-v3-p2-c3-8000	8547.0	**8547**	8547	8547	0.0	0.2	**8547**	0.0	483.3	8438.8	1.0

Continued on next page

Table A.6 – continued from previous page

instance	LB	ALNS f_b	ALNS f_a	ALNS f_w	ALNS *gap*	ALNS $t(s)$	MILP f	MILP *gap*	MILP $t(s)$	CG f	CG $t(s)$
s8-v3-p2-c4-8000	7515.0	**7515**	7537	7619	0.0	0.1	**7515**	0.0	184.1	7356.1	0.9
s8-v3-p3-c3-8000	12978.0	**12978**	13088.3	13181	0.0	0.4	**12978**	0.0	7205.1	12978.0	4.4
s8-v3-p3-c4-8000	10065.7	**10400**	10419.7	10500	3.2	0.2	**10297**	2.2	7200.6	10065.7	5.8
s8-v5-p2-c3-8000	5450.0	**5450**	5502.1	5647	0.0	0.1	**5450**	0.0	3293.2	5130.3	1.4
s8-v5-p2-c4-8000	5019.0	**5019**	5020.8	5022	0.0	0.1	**5019**	0.0	208.6	4504.2	1.8
s8-v5-p3-c3-8000	7800.0	**8078**	8088.6	8178	3.4	0.4	**8078**	3.4	7201.2	7800.0	6.3
s8-v5-p3-c4-8000	6602.0	**6602**	6671.3	6705	0.0	0.2	**6602**	0.0	4443.3	6226.5	4.9
s10-v3-p2-c3-8000	9357.6	**9577**	9751.8	9883	2.3	0.3	**9577**	2.3	7203.6	9357.6	2.1
s10-v3-p2-c4-8000	8237.0	**8237**	8257.6	8342	0.0	0.2	**8237**	0.0	765.6	7923.5	3.1
s10-v3-p3-c3-8000	14093.0	**14420**	14440	14520	2.3	0.8	14523	3.0	7217.9	14093.0	5.8
s10-v3-p3-c4-8000	11306.4	**11432**	11554.5	11636	1.1	0.4	11736	3.7	7209.3	11306.4	19.5
s10-v5-p2-c3-8000	5876.9	**6077**	6090.6	6189	3.3	0.3	**6077**	3.3	7200.2	5693.3	2.5
s10-v5-p2-c4-8000	5238.0	**5238**	5238.1	5239	0.0	0.2	**5238**	0.0	565.7	4886.2	2.4
s10-v5-p3-c3-8000	8663.0	**8820**	8850	8920	1.8	0.7	9220	6.0	7220.4	8663.0	12.6
s10-v5-p3-c4-8000	6853.4	**7132**	7153.1	7235	3.9	0.4	7335	6.6	7215.4	6853.4	10.5
s12-v3-p2-c3-8000	11740.9	**12049**	12059.4	12150	2.6	0.4	12461	5.8	7223.1	11740.9	12.3
s12-v3-p2-c4-8000	9445.5	**9888**	9917.9	9988	4.5	0.3	**9888**	4.5	7213.9	9445.5	7.8
s12-v3-p3-c3-8000	17899.2	**18126**	18178.1	18230	1.3	2.2	19672	9.0	7206.8	17899.2	35.2
s12-v3-p3-c4-8000	13421.9	**13904**	14025.9	14107	3.5	0.8	15034	10.7	7204.3	13421.9	23.6
s12-v5-p2-c3-8000	7266.0	**7549**	7590.8	7750	3.7	0.5	7757	6.3	7208.6	7266.0	8.7
s12-v5-p2-c4-8000	5911.7	**6188**	6241	6391	4.5	0.4	6189	4.5	7205.9	5911.7	8.9
s12-v5-p3-c3-8000	10712.5	**11128**	11188.8	11231	3.7	2.8	11437	6.3	7211.8	10712.5	27.0
s12-v5-p3-c4-8000	8229.4	**8607**	8647.4	8711	4.4	1.0	8805	6.5	7213.8	8229.4	35.6
s14-v3-p2-c3-8000	12859.2	**13183**	13223.9	13284	2.5	0.5	14399	10.7	7214.6	12859.2	7.5
s14-v3-p2-c4-8000	10648.9	**11018**	11029.2	11120	3.4	0.6	11627	8.4	7209.9	10648.9	16.6
s14-v3-p3-c3-8000	19387.0	**19981**	20082.7	20388	3.0	4.4	20184	3.9	7208.7	19387.0	37.1
s14-v3-p3-c4-8000	14821.0	**15243**	15263	15343	2.8	2.0	17188	13.8	7213.1	14821.0	59.6
s14-v5-p2-c3-8000	7983.8	**8184**	8194.1	8285	2.4	0.6	8392	4.9	7205.8	7983.8	31.2
s14-v5-p2-c4-8000	6439.3	**6819**	6898.7	7118	5.6	0.6	7528	14.5	7210.6	6439.3	6.3
s14-v5-p3-c3-8000	11852.4	**12281**	12301.9	12386	3.5	3.4	14030	15.5	7212.4	11852.4	38.1
s14-v5-p3-c4-8000	9142.1	**9343**	9454.2	9548	2.2	1.4	11264	18.8	7210.9	9142.1	61.1
s16-v3-p2-c3-8000	14380.4	**14522**	14562.5	14623	1.0	0.9	15443	6.9	7210.6	14380.4	45.0
s16-v3-p2-c4-8000	11366.2	**11947**	11978.1	12050	4.9	0.9	13899	18.2	7211.1	11366.2	19.2
s16-v3-p3-c3-8000	20866.8	**21731**	21916.3	22141	4.0	6.7	26740	22.0	7209.8	20866.8	121.3
s16-v3-p3-c4-8000	15880.6	**16272**	16426.4	16685	2.4	3.0	18534	14.3	7224.7	15880.6	50.7
s16-v5-p2-c3-8000	8706.1	**8922**	8962.6	9122	2.4	1.2	11177	22.1	7216.1	8706.1	25.9
s16-v5-p2-c4-8000	6977.6	**7447**	7508.9	7650	6.3	0.9	8449	17.4	7225.8	6977.6	20.8
s16-v5-p3-c3-8000	12789.3	**13331**	13476.9	13640	4.1	6.2	15510	17.5	7216.3	12789.3	55.2
s16-v5-p3-c4-8000	9766.5	**10072**	10169.9	10388	3.0	2.9	11119	12.2	7209.3	9766.5	63.1

Table A.7: large instances

		ALNS				
instance	LB	f_b	f_a	f_w	gap(%)	t(s)
s20-v5-p2-c3-8000	12188.5	12806	12909.6	13115	4.8	4.1
s20-v5-p2-c3-16000	7409.5	7765	8015	8184	4.6	1.4
s20-v5-p2-c4-8000	9462.0	9868	10010.8	10287	4.1	2.7
s20-v5-p2-c4-16000	7087.8	7763	7795.3	7870	8.7	2.8
s20-v5-p3-c3-8000	17881.4	18476	18737.1	19000	3.2	18.0
s20-v5-p3-c3-16000	9854.0	10181	10501	10702	3.2	5.8
s20-v5-p3-c4-8000	13438.6	14381	14557.1	14797	6.6	9.0
s20-v5-p3-c4-16000	7826.4	8393	8601.7	8914	6.8	4.5
s20-v3-p2-c3-8000	19863.7	20806	20937.9	21112	4.5	3.4
s20-v3-p2-c3-16000	11951.3	12461	12625.4	12770	4.1	2.1
s20-v3-p2-c4-8000	15843.3	16068	16159.2	16377	1.4	2.4
s20-v3-p2-c4-16000	9671.3	10087	10212.4	10297	4.1	1.5
s20-v3-p3-c3-8000	28904.7	30178	30465.6	30896	4.2	20.1
s20-v3-p3-c3-16000	15840.2	16480	16848.3	17301	3.9	6.6
s20-v3-p3-c4-8000	21964.4	23380	23646.4	23892	6.1	9.1
s20-v3-p3-c4-16000	12789.5	13593	13788.3	14205	5.9	4.4
s40-v8-p2-c3-8000	26291.0	28614	28959.1	29348	8.1	47.0
s40-v8-p2-c3-16000	15283.1	16726	17125.5	17365	8.6	20.9
s40-v8-p2-c4-8000	21064.5	22666	22906.6	23308	7.1	31.8
s40-v8-p2-c4-16000	12637.0	13601	13799.5	14086	7.1	9.0
s40-v8-p3-c3-8000	33332.9	35855	36513.8	37031	7.0	143.7
s40-v8-p3-c3-16000	18848.1	20811	21144.2	21373	9.4	54.6
s40-v8-p3-c4-8000	25807.2	27737	28040.5	28710	7.0	87.3
s40-v8-p3-c4-16000	15313.6	16824	17167.5	17646	9.0	23.0
s40-v5-p2-c3-8000	41228.1	44515	44891.1	45773	7.4	40.0
s40-v5-p2-c3-16000	24041.8	25923	26430.8	26979	7.3	18.0
s40-v5-p2-c4-8000	32531.6	34962	35338.2	35898	7.0	33.1
s40-v5-p2-c4-16000	19740.2	20786	21175.9	21413	5.0	10.4
s40-v5-p3-c3-8000	51363.0	56067	56652.1	57118	8.4	134.5
s40-v5-p3-c3-16000	29233.9	32220	32573.9	32969	9.3	46.0

Continued on next page

Table A.7 – continued from previous page

instance	LB	ALNS				
		f_b	f_a	f_w	gap(%)	t(s)
s40-v5-p3-c4-8000	40220.4	42832	43505.7	44094	6.1	84.3
s40-v5-p3-c4-16000	23551.8	25719	26084.4	26450	8.4	29.4
s60-v15-p2-c3-8000	41780.3	46210	46638.9	47495	9.6	408.1
s60-v15-p2-c3-16000	23941.0	26339	26728.4	27268	9.1	120.1
s60-v15-p2-c4-8000	32678.6	36056	36521.4	37241	9.4	251.5
s60-v15-p2-c4-16000	19278.8	21463	21752.3	22078	10.2	86.7
s60-v15-p3-c3-8000	52705.7	59290	59725.3	60357	11.1	890.7
s60-v15-p3-c3-16000	29862.5	33034	33609.6	34542	9.6	367.8
s60-v15-p3-c4-8000	40697.6	45345	45897.1	46424	10.2	611.1
s60-v15-p3-c4-16000	23518.1	26332	26713.7	27155	10.7	247.7
s60-v20-p2-c3-8000	28885.1	31882	32295.4	32797	9.4	384.7
s60-v20-p2-c3-16000	16554.0	18411	18611.4	18885	10.1	137.5
s60-v20-p2-c4-8000	22593.3	25235	25444.7	25821	10.5	265.1
s60-v20-p2-c4-16000	13319.0	14960	15168.7	15487	11.0	94.6
s60-v20-p3-c3-8000	36437.0	40961	41531.1	41945	11.0	1250.5
s60-v20-p3-c3-16000	20636.7	22929	23362.9	24107	10.0	422.5
s60-v20-p3-c4-8000	28136.6	31103	31851.4	32211	9.5	956.7
s60-v20-p3-c4-16000	16289.6	18325	18707.8	19348	11.1	267.2