Les Cahiers du GERAD ISSN: 0711–2440

Solving a real-world multi-depot multi-period petrol replenishment problem with complex loading constraints

A. Bani, I. El Hallaoui,
A.I. Corréa, A. Tahir

G–2019–79

October 2019

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
– Bibliothèque et Archives Canada, 2019

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2019
– Library and Archives Canada, 2019

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Solving a real-world multi-depot multi-period petrol replenishment problem with complex loading constraints

Abderrahman Bani [a]
Issmail El Hallaoui [a]
Ayoub Insa Corréa [b]
Adil Tahir [a]

[a] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada*

[b] *University of Thies, Senegal*

abderrahman.bani@gerad.ca
issmail.elhallaoui@gerad.ca
ayoub@univ-thies.sn
adil.tahir@gerad.ca

October 2019
Les Cahiers du GERAD
G–2019–79

Abstract: In this paper, we solve a rich real-word Multi-Depot Multi-Period Petrol Replenishment Problem with a heuristic based on Branch-and-Price heuristic. The network consists of five distinct depots, a group of five private carriers with heterogeneous fleets of compartmented tank-trucks and five types of gas to replenish three main groups of clients on a weekly basis. Due to the hazardous nature of the products carried, some complex handling rules apply and are addressed in the column generation sub-problem as an Elementary Shortest Path Problem with Resource Constraints. Acceleration strategies are discussed. Numerical results on some real-world data show the effectiveness and high potential of the proposed approach.

Keywords: Mathematical programming, branch-and-price, petrol replenishment, multi-compartment vehicle routing problem

1 Introduction

The petroleum sector is still a major component of most of modern societies even though other alternative sources of energy (gas, electricity, solar) are increasingly used. One of the key success factors in the petroleum industry is the efficiency of its distribution system. The sound management of its different activities is a real challenge. These activities include the procurement, the transportation of petroleum products from depots to sales points and the inventory management at both depots and sales points.

In this paper, we focus on the transportation of different types of gas from the depots to a network of distinct clients by a fleet of tank-trucks in a real-world application. Some complex handling rules apply due to the hazardous nature of the products carried. On a weekly basis, a Petroleum Company (PC) based in West Africa has to replenish with five types of gas a network of clients from five different depots using a heterogeneous fleet of compartmented tank-trucks. The tank-trucks are owned by five private transportation companies that are also doing business with some of PC competitors. That is, a pool of petroleum companies are contracting with the same group of private transportation companies. The rental of a specific truck T by a petroleum company from a transportation company is subject to its temporary availability since it may have been already rented by a competitor of the petroleum. This is one of the justification of the design of an efficient decision support system. The network of clients is comprised of three distinct subfamilies of clients: (i) gas stations mostly owned by the PC (ii) marine stations and (iii) private bakeries. The main attributes are then:

1. Rich Vehicle Routing Problem (VRP) with multiple depots and multiple products;
2. Limited number of a heterogeneous fleet of multi-compartment vehicles with up to 15 compartments which is absolutely huge;
3. Hazardous material (gas) transportation problem;
4. Cyclic network on a weekly horizon (multi-period) and many customers per route. Each can be visited more than once in a day/the horizon. From one planning period (horizon or instance) to another one the number and type of clients to be serviced may vary. It is not a rolling horizon.

These characteristics, often studied separately in the literature, make the problem complex and uncommon in the literature. The mid-term goals of this research project are : (i) provide near optimal routes and delivery schedules by considering the complex loading constraints (ii) rapidly re-optimize the solution in case of accidents, vehicle breakdowns or unexpected unavailability of drivers and (iii) pave way to the design of an effective Decision Support System (DSS). We address in this paper the first point. Our main contributions are the following:

- A Branch-and-Price based heuristic introducing a new branching rule;
- A pattern learning technique to accelerate the generation of routes/schedules respecting the very complex loading constraints, i.e., the solution of the column generation sub-problem.The problem is unsolvable without this technique even with a state-of-the-art solver;
- Two effective acceleration has been found;
- Extensive numerical results showing that our approach produces better quality solutions (in average 0.31 % of gap in less than 3.5 minutes) than a state-of-the-art Branch-and-Price method and the solutions provided by the PC.

This paper is organized as follows: After the introduction (Section 1), we review the most important publications related to Multi-Depot Multi-Period Petrol Replenishment Problem (MDMPPRP) in Section 2. In Section 3, we give a detailed description of the problem. The compact mathematical model is defined in Section 4, followed by a description of the proposed solution method (Section 5). In Section 6, we present and discuss the computational results obtained on real-world instances provided by PC before the conclusion and some suggestions for future research (Section 7).

2 Literature review

A variety of applications of the Petrol Replenishment Problem (PRP) have been reported in the literature. This type of problem is considered as the most studied variant of the Multi-Compartment VRP (MCVRP) [1] which is a generalization of the classical vehicle routing problem. The MCVRP has several aspects: (i) it considers demand for multiple heterogeneous products instead of a single product (ii) a vehicle has multiple compartments instead of a single one (iii) all goods delivered on a route must be assigned to compartments of a vehicle (iv) some products must not be loaded in the same vehicle and (v) presence of incompatibility between some products and compartments.

One of the first articles in the context of PRP was published in 1981 by [2]. They developed a method to solve a problem minimizing the transport costs with only one client per route. Each order corresponds to a full tank-truck load. The authors proposed, on the one hand, an exact approach using the set partitioning model. On the other hand, they proposed a heuristic based approach to assign orders to the tank-trucks by taking into account some operational constraints (product incompatibility, heterogeneous fleet, ...). Clients control their inventory level and place orders to the supplier, indicating their desired delivery time. [3] described a computer system they developed for Mobil Oil Corporation in the United States. The centerpiece of this system is a Computer Aided Dispatch (CAD) system where they focused on the possibility to add more than one client per route to their previous work [2]. [4] helped a large oil company in the Netherlands to redesign its distribution network. They suggested some simple models to assign clients to depots, to determine the fleet size and composition, and to restructure the depots network. [5] conducted a similar study and proposed three models to solve vehicle dispatching: set partitioning (SPP), elastic set partitioning (ESP) and set packing (SPK) models. The objective of the first two models is a transportation cost minimization. The ESP model involves a penalty in the case of constraint violation in the model, while the SPK model aims to maximize the overall profit (the cost saving between using the contracted tank-trucks against the owned tank-trucks).

The authors of [6] proposed two heuristics to solve the PRP with one depot, unlimited homogeneous tank-truck fleet and no time windows. In the first heuristic, each route visits only one client. The second one allows for multiple client routes. They mentioned that single client routes are common in distribution policy practice and showed that this practice is not efficient by applying these two heuristics to some test problems. [7] gives a more realistic constraint to the PRP by using a limited heterogeneous fleet of tank-trucks with three to nine compartments; They designed a set of least-cost tank-truck routes subject to more complex constraints than [6]. In addition, they only allow up to two stations in the same route. They adapted the Variable Neighborhood Search (VNS) [8] metaheuristic to reach a near optimal solution to this difficult problem due to the high number of constraints and variables. [9] proposed four heuristics for the Multi-Period PRP (MPPRP). The MPPRP's aim is to optimize the delivery of several petroleum products to a set of gas stations over a given planning horizon. A special feature of the problem studied was that while some clients manage their own inventories and send their orders to the vendor whenever they want, the inventory of other clients is managed by the transportation company that decides when to replenish these clients and what quantities to deliver. Their heuristics were tested using some real-world problems obtained from a transport company in eastern Quebec. These authors showed that their heuristics could lead to significant savings.

The authors of [10] proposed a Branch-and-Price algorithm based on a set partitioning model with one depot and a limited number of heterogeneous fleet of tank-trucks on a single period. They proposed a heuristic to provide an initial set of columns for the Branch-and-Price algorithm. They allow multiple clients per route. To test the performance of their approach, they used real-world data consisting of 25 customers and 6 tank trucks of 3 different types. [11] studied two small gas distribution networks in Hong Kong: the Hong Kong Island network and the network for the Kowloon Peninsula and the New Territories. They proposed a two-stage approach: heuristic clustering, and optimization assignment and routing. The clustering is based on the geographic limitation and expert considerations. The

optimal model for assignment and routing is a binary program with multiple objectives. For this case, clients inventories are managed by the vendor. The proposed model does not guarantee that no client will run out of stock.

The authors of [12] decompose the PRP using *a priori* column generation scheme. On a single period, they proposed an exact algorithm which decomposes the PRP into two sub-problems: the tank-truck loading problem (TTLP) and the Routing Problem (RP). The RP is used to select routes for all clients minimizing overall transportation costs. They reduced the complexity of the RP by allowing just two clients per route. The TTLP is used to assign orders to tank-truck compartments, maximizing the total quantity delivered and the profit. They start solving this problem by a heuristic assigning, if the heuristic fails, then they solve an exact MIP formulation of the problem. [13] proposed a heuristic algorithm to solve the MPRP. The objective is to maximize the total profit equal to the revenue minus the sum of routing costs and of regular and overtime costs. [14] proposes two heuristic approaches to solve the PRP with time windows (PRPTW).

The authors of [15, 16] conducted a study similar to [13] but proposed different approaches. [15] applied a heuristic approach to solve the MPPRP and used a simulation approach to analyze the results. [16] proposed a heuristic approach consisting of two steps "cluster first, route second". Both studies considered these problems at the inventory routing level since the supplier has full control of the client inventory, including determining order quantity as well as the delivery period. [17] conducted a further study to solve the Multi-Depot PRPTW (MDPRPTW). They proposed a new heuristic algorithm to solve the problem by using what they called trips (combination of routes and used vehicles). From this statement, it implies that multiple trips can have the same route. [18] proposed a stochastic VNS heuristic to solve MPPRP. The stochastic VNS heuristic is compared to a MIP model and a deterministic Compartment Transfer Heuristic (CTH).

A similar approach was proposed by [19], they solve their problem with a heuristic approach which combines a matheuristic component for the determination of an initial solution and a variable neighborhood descent algorithm to search further improvements. They also study the impact of fleet size costs on their solutions. [20] proves in real-world instances that considering multiple-periods gives a better outcome in comparison to a single period. They proposed two Multi-phase heuristics. They also allow clients to be visited more than once a day, which reflects the real-world situation where high demand clients could potentially be served several times per day. [21] proposed a Tabu Search algorithm where specific attention is paid to tank-trucks with compartments and clients with different types of ordered products and time windows.

Table 1 lists the studies described above and summarizes some of their characteristics to give a better understanding of them. Most of these studies proposed heuristic approaches to solve the MPPRP, due to the fact that the exact method might not be an appropriate approach to solve large scale problems [22]. Finally, to the best of our knowledge, none of these studies solve the integrated MPPRP (i.e., the TTLP and RP in one shot) using a Branch-and-Price algorithm with unlimited number of clients per route and multi-periods.

3 Problem statement

The MDMPPRP is defined over one-week horizon **D** (from Monday to Sunday), in which the PC must replenish a network of clients with five different types of gas **P** denoted (chemically) Marked Products and Unmarked Products. The marked products are *Diesel*, *Marine Zoom* and *Kerosene*. The unmarked products are *Super* and *Gasoil*.

The network of clients mainly consists of: (i) *gas stations* which are owned by PC (ii) *marine stations* located at some fishing docks and (iii) *bakeries*. However, from time to time, the PC has occasional clients (*private enterprises*). Customs escort fees apply for gas supplied to marine station. A discount on these fees is applied when two consecutive visits in different marine stations are performed.

Table 1: Main characteristics of past studies vs this paper.

Paper	Limited Fleet	Hetero. Fleet	#Clients/route	Multi Depots	Time Windows	Multi-Periods	Multi Visits/day
Brown and Graves [2]	✓	✓	1	✓			
Brown et al. [3]	✓	✓	$+\infty$	✓			
Van der Bruggen et al.[4]	✓	✓	1	✓			
Ronen [5]	✓	✓	1				
Taqa allah et al. [6]			$1/+\infty$				
Malépart et al. [9]	✓	✓	$+\infty$				
Abdelaziz et al. [7]	✓	✓	≤ 2				
Avella et al. [10]	✓	✓	$+\infty$				
Ng et al. [11]	✓	✓	$+\infty$				
Cornillier et al. [12]		✓	≤ 2				
Cornillier et al. [13]	✓	✓	≤ 2			✓	
Cornillier et al. [14]	✓	✓	$+\infty$		✓		
Popovic et al. [15]	✓	✓	≤ 2			✓	
Hanczar [16]	✓	✓	$+\infty$			✓	
Cornillier et al. [17]	✓	✓	$+\infty$	✓	✓		
Popovic et al. [18]		✓	≤ 3			✓	
Vidovic et al. [19]		✓	≤ 4			✓	
Charusakwong and Lohatepanont [20]	✓	✓	≤ 2		✓	✓	✓
Benantar et al. [21]	✓	✓	$+\infty$		✓		✓
This paper	✓	✓	$+\infty$	✓		✓	✓

The distribution of the marked and unmarked products is carried out by a pool of heterogeneous fleet of tank-trucks $\mathbf{K}$ owned by four private transportation companies. We have two types of tank-trucks according to there capacities: (i) small tank-trucks with capacity < 20000 liters (ii) jumbo tank-trucks with capacity ≥ 20000 liters. Some tank-trucks labeled 'Fishing' are exclusively used to carry the marked products while some others cannot carry unmarked products and marked products at the same time. Note that a tank-truck can be both jumbo and labeled 'Fishing' or none of that. The tank-trucks may not be all available at the same time. For each tank-truck $k \in \mathbf{K}$, we define $\mathbf{D_k}$ the subset of availability days, $\mathbf{L_k}$ the set of compartments where each compartment $l \in \mathbf{L_k}$ has constant capacity C_{lk}.

The delivery consists in completely unloading one or more compartments using gravity because the tank-trucks are not equipped with a debit meter. The business contracts between the PC and each transportation company are distinct. Different penalties applied according to the four status of tank-trucks as (i) dedicated to marked products (ii) paint in the colors of the PC (iii) accepting to be partially paid with gas at one gas station of the PC (ix) regular tank-trucks. Before supplying the clients of the PC, the tank-trucks must pick-up some gas from the depots (precedence constraints between some deliveries and pick-ups). For the remainder of this article, we will assume that the total of gas available at the depots is always greater than the total demand for petroleum products of the clients.

An order placed by a client consists in a specific product denoted by *client-product*, its associated volume needed and its latest day of delivery. The set of client-products is denoted by $\mathbf{N}$. A gas station may order several types of gas while marine stations and bakeries can order just one distinct type of gas. The orders of petroleum products are placed before Friday of the week preceding the week of distribution. The loading time at each depot, the waiting time and the unloading time at each client are constant and do not depend on the quantities poured inside the client's tanks.

Some complex handling rules must be enforced (see Table 2). In addition, to ensure the balance of any loaded tank-truck during its route, a special attention is paid to the stowage of the petroleum

products in the compartments. Parts of the gas carried during the routes of tank-trucks turn into gas due to the friction caused by the liquid on the walls of compartments, thus changing the shape and the volumes of compartments. This phenomenon is called *slushing*. Therefore, the fact of having empty space in a tank-truck at the beginning of its route should be avoided as much as possible.

The direct billing is applied. That is, the billing of each customer supplied in gas is based on the distance between the client location and the depot(s) holding the petroleum product even the tank-truck visits other clients in the same route. The billing of a route to each customer also depends on the cost per kilometer per unit of volume defined for every type of gas and the volume of gas carried. Then, with this type of billing, the transportation costs could be considered as fixed costs therefore they do not impact the optimization.

The MDMPPRP consists in constructing a transportation plan over $\mathbf{D}$ in which, the type and volume of gas supplied, the tank-trucks and compartments used and the delivery time must be determined. In addition to the pick-up periods at the depot, the exact volumes of gas and the compartments loaded in every tank-truck. The objective function includes the following costs: (i) sum of the penalties incurred for fleet sizing purpose (ii) sum of the customs escort fees (iii) route length (ix) sum of the empty space of the tank-trucks.

Table 2: Main business rules.

Rule	Description
R-1	Exclusion constraint between marked products and unmarked products.
R-2	Some clients cannot accommodate 'jumbo' tank-trucks.
R-3	Tank-trucks labeled 'Fishing' may only carry marked products.
R-4	Demand satisfaction: each client-product must be served by a single tank-truck.
R-5	The quantities of products at the depots are limited.
R-6	When a compartment of a tank-truck is used, it must be completely filled.
R-7	One client-product per compartment.
R-8	We can only deliver what we collect from the depot.
R-9	The daily total travel time must not exceed the total working time in a day.
R-10	A truck can not visit more than two marine stations.

4 Mathematical model

We first introduce the network representation followed by some notations. Then, we give a mixed integer programming formulation with explanation of each constraint.

4.1 Network representation

Let $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ be a directed multi-graph, where $\mathbf{V}$ is the node set and $\mathbf{A}$ the arc set. The network $\mathbf{G}$ is partially illustrated in Figure 1, where for clarity reasons, some arcs are truncated or omitted. In addition to one source node (σ) and one destination node (δ), this network contains five node types: *depot*, *client*, *client-departure*, *client-arrival* and *client-product*. The client-product nodes $\mathbf{N}$ represent a product $p \in \mathbf{P}$ requested by a client, while $\mathbf{N_p}$ is the subset of client-products nodes of product $p \in \mathbf{P}$. There is a pair of departure and arrival nodes for each client. The subset of fictive nodes $\mathbf{V'}$ are the client-arrival which represent the entry to the client site and the client-departure nodes which represent the exit from each client site.

The network involves four arc types connecting the different nodes mentioned above: *travel arc*, *pickup arc*, *delivery arc* and *end of delivery arc*. The *travel arcs* link the client-departure node of each client to the client-arrival node of other clients, depot node to client-arrival node. The *pickup arcs* connect the source node with depots and the depots between them. *Delivery arcs* link the client-arrival node with the client-product nodes of the same client. The arcs connecting the client-product node of a client with the client-departure node are the *end of delivery arcs*. It should be noted that the graph above contains two disjoint sub-graphs: the sub-graph of unmarked products and the sub-graph of marked products.

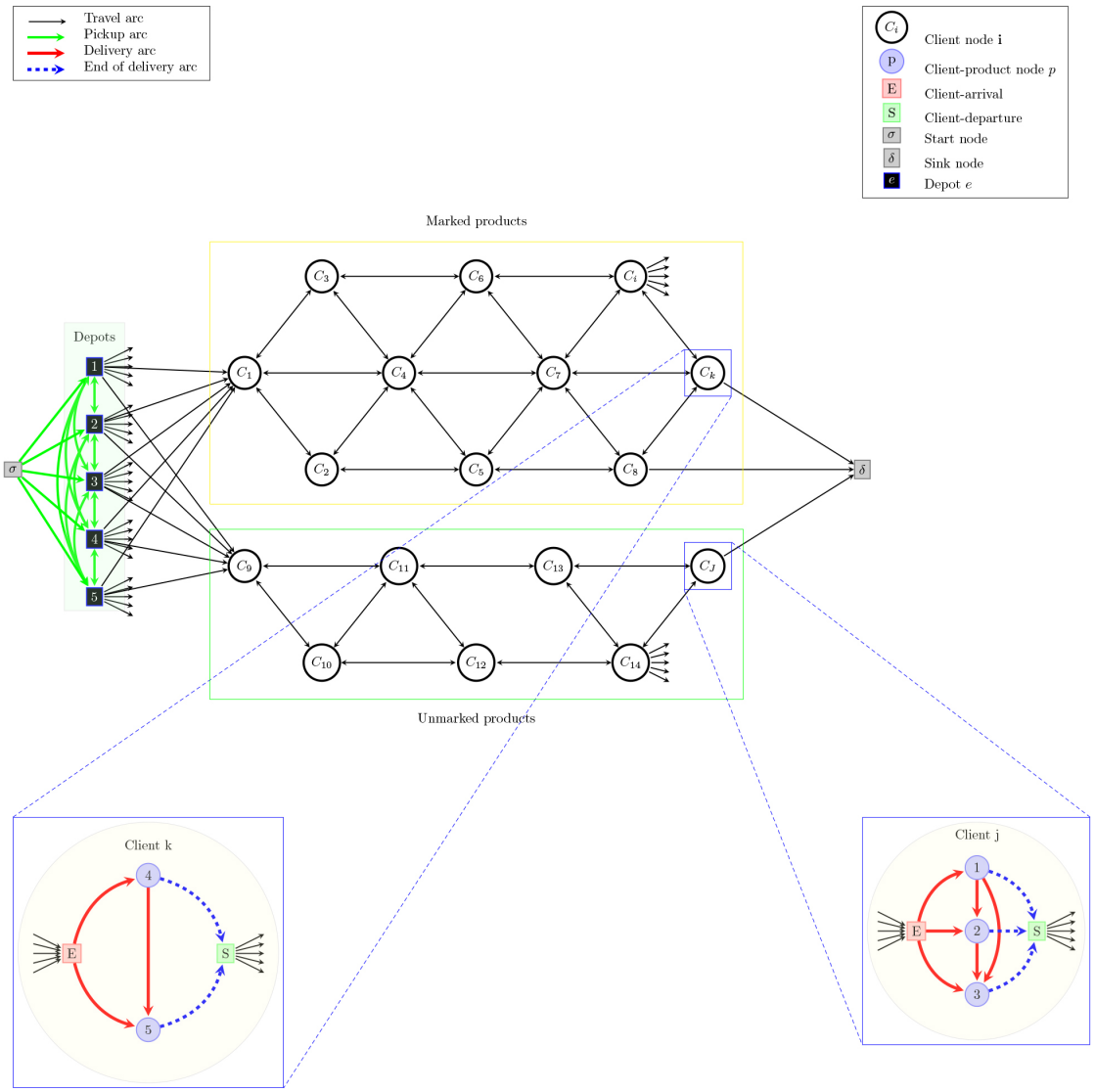

Figure 1: Example of a part of the network.

The problem can be modeled as a large-scale mixed integer program. Whereas this formulation is too large to be used in practice, it is helpful to understand the problem and the decomposition method that follows.

4.2 Sets and elements

- $\mathbf{P}$: Set of all products;
- $\mathbf{N}$: Set of all client-products nodes, each client-product node corresponds to an order of one product by a client;
- $\mathbf{N_p}$: Subset of client-products of product $p \in P$;
- $\mathbf{N_f}$: Set of client-products from marine stations;
- $\mathbf{E}$: Set of depots;
- $\mathbf{V'}$: Set of fictive nodes;
- $\mathbf{V} = \{\mathbf{N} \cup \mathbf{E} \cup \{\sigma, \delta\}\} \cup \mathbf{V'}$: Set of all nodes of the network including the source and destination nodes in addition to fictive nodes;

- **S**: Subset of vertices in the sub tour;
- **A**: Set of possible connections (arcs). Note that the Rule R-1 are respected in connection definition;
- **K**: Set of tank-trucks;
- **D**: Set of days in a week (numbered from 0 to 6) ;
- $\mathbf{D_k}$: Set of availability days for each tank-truck $k \in K$;
- $\mathbf{L_k}$: Set of compartments of each tank-truck $k \in K$;

4.3 Parameters and variables

The parameters and variables of the proposed model are defined as follows:

Parameters

- Q_{pe}: Quantity of product p available in depot e;
- C_{lk}: Capacity of compartment l in tank-truck k;
- c_{ij}: Cost of visiting node j after node i including the distances and custom escort fees;
- τ_{ij}: Time of visiting node j after node i, including the waiting time and the unloading times of the products;
- T_{max}: Length of a working day;
- o_n: Quantity of product ordered by the client-product $n \in N$;
- Ψ_k: Penalty value of using tank-truck k;
- Δ_k: Capacity of tank-truck $k \in K$, also equal to $\sum_{l \in L_k} C_{lk}$;

The decision variables

- $u_{kd} = 1$ if a tank-truck k is used on day d, 0 otherwise.
- $x_{ij}^{kd} = 1$ if a tank-truck k visit node j on day d after visiting the node i, 0 otherwise. Note that the rules R-2 and R-3 are implicitly considered in this variables definition.
- $y_{ln}^{kd} = 1$ if the product in client-product n is loaded in compartment l of tank-truck k on day d, 0 otherwise.
- $q_{pe}^{kd} \geq 0$ quantity of product p picked-up from depot e by tank-truck k on day d.

4.4 MDMPPRP compact formulation

Given this notation, the problem can be modeled as follows:

$$\min_{x,y,u,q} \quad \sum_{k \in K} \sum_{d \in D_k} \left((\Psi_k u_{kd}) + \left(\sum_{(i,j) \in A} c_{ij} x_{ij}^{kd} \right) + \left(\Delta_k u_{kd} - \sum_{l \in L_k} \sum_{n \in N} C_{lk} y_{ln}^{kd} \right) \right) \tag{1}$$

$$\text{s.t.:} \quad \sum_{(i,n) \in A} \sum_{k \in K} \sum_{d \in D_k} x_{in}^{kd} = 1 \qquad \forall n \in N \tag{2}$$

$$\sum_{k \in K} \sum_{d \in D_k} q_{pe}^{kd} \leq Q_{pe} \qquad \forall p \in P, \forall e \in E \tag{3}$$

$$q_{pe}^{kd} \leq Q_{pe} \sum_{i \in E \setminus \{e\}} x_{ie}^{kd} \qquad \forall p \in P, \forall e \in E, \forall k \in K, \forall d \in D_k \tag{4}$$

$$x_{ij}^{kd} \leq u_{kd} \qquad \forall (i,j) \in A, \forall k \in K, \forall d \in D_k \tag{5}$$

$$\sum_{l \in L_k} C_{lk} y_{ln}^{kd} = o_n \sum_{(i,n) \in A} x_{in}^{kd} \qquad \forall n \in N, \forall k \in K, \forall d \in D_k \tag{6}$$

$$\sum_{n\in N} y_{ln}^{kd} \leq 1 \qquad \forall k \in K, \forall d \in D_k, \forall l \in L_k \tag{7}$$

$$\sum_{e\in E} q_{pe}^{kd} = \sum_{n\in N_p} \sum_{l\in L_k} C_{lk} y_{ln}^{kd} \qquad \forall p \in P, \forall k \in K, \forall d \in D_k \tag{8}$$

$$\sum_{(i,j)\in A} \tau_{ij} x_{ij}^{kd} \leq T_{max} \qquad \forall k \in K, \forall d \in D_k \tag{9}$$

$$\sum_{(i,j)\in\{N\times N_f\}} x_{ij}^{kd} \leq 2 \qquad \forall k \in K, \forall d \in D_k \tag{10}$$

$$\sum_{i\in V} x_{ij}^{kd} - \sum_{i\in V} x_{ji}^{kd} = \begin{cases} -1 & \text{if } j = \sigma \\ 0 & \text{if } j \in V \setminus \{\sigma, \delta\} \\ 1 & \text{if } j = \delta \end{cases} \qquad \forall k \in K, \forall d \in D_k \tag{11}$$

$$\sum_{i,j\in S} x_{ij}^{kd} \leq |S| - 1 \qquad \forall k \in K, \forall d \in D_k, (S \subset V, 2 \leq |S| \leq |V| - 2) \tag{12}$$

$$x_{ij}^{kd} \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in K, \forall d \in D_k, \forall p \in P \tag{13}$$

$$u_{kd} \in \{0,1\} \qquad \forall k \in K, \forall d \in D_k \tag{14}$$

$$y_{ln}^{kd} \in \{0,1\} \qquad \forall n \in N, \forall k \in K, \forall d \in D_k, \forall l \in L_k \tag{15}$$

$$q_{pe}^{kd} \geq 0 \qquad \forall k \in K, \forall d \in D_k, \forall p \in P, \forall e \in E \tag{16}$$

The objective function (1) aims at minimizing the total cost, including the penalty costs of using tank-trucks, total distance, and customs escort fees. The total cost includes also the empty space in tank-trucks used. Constraints (2) indicate that each client-product must be visited exactly once as stipulated by Rule R-4. The Rule R-5 on quantities available at the depot is ensured by constraints (3). Constraints (4) ensure that tank-truck can only load products from the depot after visiting it. Constraints (5) ensure that a tank-truck can be used only if available on the day d. Constraints (6) indicate that the sum of the capacities of the compartments in which we want to put the order of client-product n must correspond exactly to the quantity demanded as introduced in Rule R-6. Constraints (7) impose that for each tank-truck k, at most one client-product is allowed in the same compartment on a given day d (R-7). Constraints (8) define that the quantities of each product delivered to clients must be loaded from depots as mentioned in Rule R-8. Constraints (9) make sure that the daily total travel time must fall within the length of a working day for each driver tank-truck requested by Rule R-9. Constraints (10) state that a tank-truck cannot visit more than two marine stations as imposed by Rule R-10. Constraints (11) and (12) are the flow conservation constraints and sub-tour elimination constraints respectively. Finally, the domain of the decision variables are restricted by (13)–(16).

5 Solution method

We propose a new Branch-and-Price Heuristic to solve this hard problem where each node in the branch and bound tree is evaluated using Column Generation (CG). CG is closely connected to Dantzig–Wolfe decomposition which is introduced by [23]. It involves reformulating the problem as a Restricted Master Problem (RMP) and one or more Column Generation Sub-Problems (CGSPs). To perform CG, we reformulate the MDMPPRP compact formulation described in Section 4.4 as a set partitioning Master Problem (MP) in Section 5.1 and define the CG Sub-Problems (CGSPs) in Section 5.2. CG is an iterative approach where in every iteration, we consider only a subset of all possible columns (or routes) in the MP, that is why it is known as restricted MP (RMP). The RMP guides the CGSPs to generate new promising columns (routes) by providing them at each iteration with a dual solution. The CG procedure continues until no negative reduced cost columns (using this dual solution) are generated. To solve the CGSPs, we have used a dynamic labeling algorithm given in Section 5.3. Finally, the Section 5.4 outlines how we obtain integer solutions.

5.1 The master problem

We reformulate the MDMPPRP as a set-partitioning model using Dantzig–Wolfe Decomposition [23]. For this, let $\Omega = \{1, \ldots, |\Omega|\}$ be the set of feasible routes (columns). For each route $\omega \in \Omega$, a binary variable ρ_ω is associated. ρ_ω gets the value 1 if route ω with cost $c_\omega = (c^e_\omega + c^p_\omega + c^d_\omega + c^g_\omega)$ is selected or 0 otherwise where c^e_ω is the sum of customs escort fees, c^p_ω is the penalty of using the tank-truck of the ω route, c^d_ω is the sum of the distances traveled by the ω route tank-truck, and c^g_ω is the penalty for the empty space in ω route.

The coefficient $s_{n\omega}$ is set to 1 if the tank-truck running the route $\omega \in \Omega$ serves the client-product $n \in N$, 0 otherwise. The coefficient $u_{kd\omega}$ is equal to 1 if the route ω is served by a tank-truck k in day d, 0 otherwise. The coefficient $q_{pe\omega}$ is the quantity of the product p loaded from the depot e to satisfy the order for the product p on the route ω. The mathematical formulation of our master problem is as follows:

$$\min_{\rho} \quad \sum_{\omega \in \Omega} c_\omega \rho_\omega \tag{17}$$

$$\text{s.t.:} \quad \sum_{\omega \in \Omega} s_{n\omega} \rho_\omega = 1 \qquad \forall n \in N \tag{18}$$

$$\sum_{\omega \in \Omega} u_{kd\omega} \rho_\omega \leq 1 \qquad \forall k \in K, \forall d \in D_k \tag{19}$$

$$\sum_{\omega \in \Omega} q_{pe\omega} \rho_\omega \leq Q_{pe} \qquad \forall p \in P, \forall e \in E \tag{20}$$

$$\rho_\omega \in \{0, 1\} \qquad \forall \omega \in \Omega \tag{21}$$

The objective function (17) is to minimize the total cost. The partitioning constraints (18) indicate that each client-product must be visited exactly once, note that we perform multiple visits per day if we visit two client-products of the same client in the same day. The additional constraints (19) ensure that we can use the tank-truck k only once per period d. The additional constraints (20) ensure that the quantities loaded from each depot do not exceed what is available. Finally, integrality constraints (21) are imposed on the variables ρ_ω. These latter are relaxed when the CG is used to find the lower bound in a branching node.

In summary, in the master problem, we manage two constraints of the original formulation: the demand satisfaction requirement (2) and the availability of products at the depots (3). So, the other constraints (4)–(16) are managed in the CGSPs.

5.2 The column generation sub-problem

The constraints (4)–(16) are decomposed by tank-truck k and day d. So, we modeled our pricing problem CGSP as an ESPPRC. It consists in a connected graph $G^{kd}(V^{kd}, A^{kd})$ where $V^{kd} \subseteq V$ is the subset of nodes possible to be covered by tank-truck k on day $d \in D_k$; this subset includes the nodes σ and δ, N^{kd}, N^{kd}_p, N^{kd}_f and depot nodes. $A^{kd} \subseteq A$ is the subset of arcs induced by V^{kd}. The ESPPRC finds a least negative reduced cost route among all the routes from σ to δ that satisfy the resource constraints imposed by the set of rules (R-5)–(R-10).

Every resource-feasible route from σ to δ in the ESPPRC corresponds to a feasible route $\omega \in \Omega$. The cost of this route is the sum of the costs of its arcs A^ω and is equal to the corresponding route cost c_ω. However, in a CGSPs, the arc costs need to be modified because the role of a CGSP is to find negative reduced cost routes, if at least one exists. Hence, the cost of a route should be changed to the reduced cost of the corresponding route variable. Let π_n, γ_{dk}, θ_{pe}, be the dual variables associated with the master problem constraints (18), (19), (20), respectively. To ensure that, we must define the appropriate arc reduced cost $\bar{c}_{ij}$ for each arc $(i, j) \in A$. For that, we define four sets of arcs:

- A^{ω}: Set of arcs of the route ω.
- A^{u}_{n}: Set of unloading arcs is corresponding to serve client-product n.
- A^{l}_{pe}: Set of loading arcs is corresponding to pickup a quantity q_{pe} of product p from depot e.
- A^{o}: Set of other arcs.

The reduced cost $\overline{c}_{\omega}$ of a variable ρ_{ω} is then given by:

$$\overline{c}_{\omega} = \quad c_{\omega} - (\sum_{n\in N} s_{n\omega}\pi_{n}) - (\sum_{k\in K}\sum_{d\in D_k} u_{kd\omega}\gamma_{dk}) - (\sum_{p\in P}\sum_{e\in E} q_{pe\omega}\theta_{pe}) = \sum_{(i,j)\in A^{\omega}} \overline{c}_{ij} \tag{22}$$

In this case, the (reduced) cost $\overline{c}_{ij}$ of arc $(i,j) \in A^{\omega}$ is given by:

$$\overline{c}_{ij} = \begin{cases} c_{ij} - \pi_{n} & \text{if } (i,j) \in A^{u}_{n} \\ c_{ij} - q_{pe}\theta_{pe} & \text{if } (i,j) \in A^{l}_{pe} \\ c_{ij} & \text{if } (i,j) \in A^{o} \end{cases} \tag{23}$$

The CGSP^{kd} is formulated as follows:

$$\min_{x,y} \quad (\Psi_{k} - \gamma_{dk}) + \left(\sum_{(i,j)\in A} \overline{c}_{ij} x^{kd}_{ij}\right) + \left(\Delta_{k} - \sum_{l\in L_k}\sum_{n\in N^{kd}} C_{lk} y^{kd}_{ln}\right) \tag{24}$$

$$\text{s.t.:} \quad q^{kd}_{pe} \leq Q_{pe} \sum_{i\in E\setminus\{e\}\cup s} x^{kd}_{ie} \qquad \forall p \in P, \forall e \in E \tag{25}$$

$$\sum_{l\in I_k} C_{lk} y^{kd}_{ln} = o_{n} \sum_{(i,n)\in A^{kd}} x^{kd}_{in} \qquad \forall n \in N, \forall l \in L_k \tag{26}$$

$$\sum_{n\in N} y^{kd}_{ln} \leq 1 \qquad \forall l \in L_k \tag{27}$$

$$\sum_{e\in E} q^{kd}_{pe} = \sum_{n\in N^{kd}_{p}}\sum_{l\in L_k} C_{lk} y^{kd}_{ln} \qquad \forall p \in P \tag{28}$$

$$\sum_{(i,j)\in A^{kd}} \tau_{ij} x^{kd}_{ij} \leq T_{max} \tag{29}$$

$$\sum_{(i,j)\in\{N^{kd}\times N^{kd}_{f}\}} x^{kd}_{ij} \leq 2 \tag{30}$$

$$\sum_{i\in V^{kd}} x^{kd}_{ij} - \sum_{i\in V^{kd}} x^{kd}_{ji} = \begin{cases} -1 & \text{if } j = \sigma \\ 0 & \text{if } j \in V^{kd}\setminus\{\sigma,\delta\} \\ 1 & \text{if } j = \delta \end{cases} \tag{31}$$

$$\sum_{i,j\in S} x^{kd}_{ij} \leq |S| - 1 \qquad \forall k \in K, \forall d \in D_k \quad (S \subset V^{kd}, 2 \leq |S| \leq |V^{kd}| - 2) \tag{32}$$

$$y^{kd}_{ln} \in \{0,1\} \qquad \forall n \in N^{kd}, \forall l \in L_k \tag{33}$$

$$x^{kd}_{ij} \in \{0,1\} \qquad \forall (i,j) \in A^{kd} \tag{34}$$

The objective function (24) minimizes the reduced cost. The constraints (25)–(34) are a restriction of constraints (4)–(16) to tank-truck k and day d respectively.

5.2.1 Resource constraint reformulation.

We replace the constraints (29)–(30) with resource constraint reformulation suitable for the algorithm presented in Section 5.3. So, let $R = \{0,1,2,3\}$ be the set of resources where resource 0 represents the time and resource 1 represents the number of marine stations in the route while resources 2 and 3 correspond to tank-truck resources.

Time and marine resources. The $[a_0, w_0] = [0, T_{max}]$ are the lower and upper bounds on the time resource, while $[a_1, w_1] = [0, 2]$ are the lower and upper bounds on the number of marine stations resource. The resource consumption for the resource 0 is $r^0_{ij} = \tau_{ij}, \forall (i,j) \in A^{kd}$, while the resource consumption for the resource 1 is $r^1_{ij} = 1$ if the type of the j node is a marine station, 0 otherwise.

Tank-truck resources. We also use two additional resources to check feasibility of a route such as capacity and number of full compartments denoted by resource 2 and 3 respectively. For resource 2, $[a_2, w_2] = [0, \Delta_k]$ are the lower and upper bounds and the resource consumption is $r^2_{ij} = o_j$ if node $j \in N_{kd}$, 0 otherwise. In the meanwhile, the lower and upper bounds of resource 3 are $[a_3, w_3] = [0, |L_k|]$ and the resource consumption is given by $r^3_{ij} = \{$the minimum number of compartments required to load the quantity $o_j\}$ if node $j \in N_{kd}$, 0 otherwise.

We denote by R^m_i the consumption of resource $m \in R$ over all the arcs composing of a partial route Φ_i from σ to i. The new formulation is as follows :

$$(R^m_i + r^m_{ij} - R^m_j) x^{kd}_{ij} \leq 0 \qquad \forall m \in R, \forall (i,j) \in A^{kd} \tag{35}$$

$$a^m_i \leq R^m_i \leq w^m_i \qquad \forall m \in R, \forall i \in V^{kd} \tag{36}$$

In this formulation, constraints (35) model the resource consumption along arc (i, j) whenever it is part of the route, while constraints (36) require the resource consumption along the σ-i partial route to be within the corresponding resource interval. The following section describes the Sub-problem Labeling Algorithm to determine the possible optimal partial paths and eliminate labels using some dominance rules.

5.3 Sub-problem Labeling Algorithm

Many articles covering the basics of solving ESPPRC by using dynamic programming based labeling algorithms already exist. In this section, we give the details of all modifications to classical ESPPRC to solve our sub-problem. The ESPPRC is NP-hard in the strong sense, see [24]. A relaxed version of the ESPPRC, the shortest path problem with resource constraints (SPPRC), is commonly used instead. The SPPRC can be solved with pseudo-polynomial algorithms, see [25]. However, this relaxation provides a lower bound of poor quality than those of the ESPPRC.

The basic dynamic programming algorithm was devised by [26] as an extension of the well-known Ford-Bellman algorithm. The central part of the algorithm is the use of labels to store information about resource values for partial-routes rooted at source node σ. We use the *improved definition of label* introduced by [27]. Hence, each label ζ has a set of attributes: (i) a unique id (ii) current vertex $\overline{v}(\zeta)$ (iii) accumulated reduced cost $\overline{C}(\zeta)$ (iv) the cumulative travel cost (v) the accumulated consumption of each resource given by $\overline{R}(\zeta) = (r^1, r^2, ..., r^{|R|})$ (vi) an ordered set of last $i-1$ visited nodes $vis(\zeta) \subseteq V \setminus \{\overline{v}\}$ (vii) a set of client-products served $N(\zeta)$ (viii) set of unreachable nodes $U(\zeta)$ by the tank-truck k on day d (ix) the number of unreachable nodes $s(\zeta)$ (x) the parent label id $p(\zeta)$ (xi) the tank-truck associated $k(\zeta)$ (xii) the corresponding TTLP denoted by $TTLP(\zeta)$. To summarize, each label is represented by $\zeta = \{\overline{C}, \overline{R}, \overline{v}, vis, N, U, k, p, TTLP\}$.

The concept of labeling algorithm as presented in Algorithm 1 is to iteratively extend labels according to resource constraints in the following way, until there are no more labels left. When a label has been extended, it is considered "processed". Loading constraints are managed in the Extension Function (EF).

We initialize the first label from the source node σ which is then inserted into $\mho$ in line 1. We loop as long as there are unprocessed labels left in lines 2–9. In line 3, a dominance function is used to remove dominated labels as described in Section 5.3.1. It is clear that without removing dominated labels in line 3, these results in a complete enumeration of all feasible paths. We pick out a label in line 4 which is then extended using the EF in line 9. In this function, client-products are assigned to the compartments according to the rules specified in Table 2. We use the TTLP-heuristic (defined

Algorithm 1: Labeling(G, σ, δ)

```
1  Ʊ ← {ζ_σ = {0, [0, 0, ...., 0], σ, ∅, ∅, ∅, k, ∅, ∅}} // Set of unprocessed labels
2  while Ʊ ≠ ∅ do
3      Remove-Dominated(Ʊ)
4      ζ ← pick a ζ ∈ Ʊ and Ʊ ← Ʊ \ ζ
5      foreach Node j ∈ Extendables(ζ) // Nodes to which ζ can be extended to: j ∉ U(ζ)
6       do
7          ζ_j = Extend-Label(ζ, j) // Defined in Algorithm 2
8          if v̄(ζ_j) = δ then Store-Solution(ζ_j, Sol) // List of Labels at destination.
9          else Ʊ ← Ʊ ∪ {ζ_j}
10 return Sol
```

in Section 5.3.3) as long as it finds a solution to the TTLP. If the TTLP-heuristic fails, the EF uses a MIP to solve the TTLP. The MIP formulation of the TTLP denoted by TTLP-MIP and extension function are described in the Section 5.3.1. Feasible routes that reach the destination node δ are saved in line 8, otherwise, the labels are marked as unprocessed.

5.3.1 Extension function

Algorithm 2: Extend-Label(ζ, j)

```
ζ_j ← create-Label(ζ) // create new label from a parent label on vertex j
Check the feasability of TTLP:
TTLP(ζ_j) ← Build TTLP from (N(ζ_j), k(ζ_j))
h-code ← HF(TTLP(ζ_j))
if h-code is in HM then
    if HM[h-code] is not feasible then
        return ∅
else
    if TTLP-Heuristic(ζ_j) finds a feasible solution then
        HM[h-code] ← result of TTLP-Heuristic(ζ_j)
    else
        HM[h-code] ← Solve TTLP-MIP(ζ_j)
        if TTLP-MIP(ζ_j) is not feasible then
            return ∅
Update the list of unreachable vertices:
foreach j' ∈ SUCCESSORS of j do
    if ζ_j can not be extended to j' due to
    resource constraints or the corresponding TTLP is not feasible then
        U(ζ_j) ← U(ζ_j) ∪ j'
return ζ_j
```

Calling function Extend-Label extends the existing labels at node $i \in V$ to its successor nodes $\{j \in V | (i, j) \in A\}$, checks the feasibility of the new labels, and discards infeasible ones. A label ζ_j is feasible if it corresponds to a partial route that respects the resource constraints and the corresponding $TTLP(\zeta_j)$ is feasible. The function first updates the consumption of resources. If the resource constraints are satisfied, it updates the set of client-product $N(\zeta_j)$ and solves the $TTLP(\zeta_j)$. In this case, we do not need to solve the same problem more than once. For that, we give a method to store the solved problem in a hash-map HM using a hash function (HF); the HF is used to determine if a TTLP problem is already solved or not. The HF assign a unique key (*h-code*) for each $TTLP(\zeta_j)$, we start by transforming the sorted list of $N(\zeta_j)$ into a string of characters. We concatenate this with the id of the vehicle k separated by a separator (";"). If the $TTLP(\zeta_j)$ is feasible, it explores the set of outgoing arcs of node j to update the vector of unreachable nodes $U(\zeta_j)$ and the number of unreachable nodes $s(\zeta_j)$.

We use an adaptation of the heuristic proposed in the literature by [12] to our context. In our case, we do not have lower and upper bounds on the quantity of product requested by a client-product.

5.3.2 TTLP-MIP

As we mention in Section 3, tank-trucks are not equipped with follow meter. In this case, to remove any ambiguity we consider the following rules: (i) The delivery of client-product is to completely unload one or more compartments by gravity, a compartment is either filled at 100% or empty. (ii) We can only assign one client-product per compartment. (iii) We can only ship a maximum of $|\mathbf{L_{k(\zeta_j)}}|$ client-products. (iv) A client-product can however be assigned to multiple compartments.

The mathematical formulation of the loading problem of the client-products in the tank-truck denoted by TTLP-MIP(ζ_j) takes the form:

$$\min_{y} \quad \sum_{n\in N(\zeta_j)} \sum_{l\in L_{k(\zeta_j)}} y_{ln}^{kd} \tag{37}$$

$$\text{s.t.:} \quad \sum_{n\in N(\zeta_j)} y_{ln}^{kd} \leq 1 \qquad \forall l \in L_{k(\zeta_j)} \tag{38}$$

$$\sum_{l\in L_{k(\zeta_j)}} C_{lk} y_{ln}^{kd} = o_n \qquad \forall n \in N(\zeta_j) \tag{39}$$

$$y_{ln}^{kd} \in \{0,1\} \qquad \forall l \in L_{k(\zeta_j)}, \forall n \in N(\zeta_j) \tag{40}$$

The objective function (37) minimizes the number of filled compartments. The constraints (38) ensure that at most one client-product per compartment while the constraints (39) match the compartments assigned to each client-product with the requested quantity.

5.3.3 TTLP-Heuristic

Here we describe how to solve the $TTLP(\zeta_j)$ using TTLP-Heuristic. Let Γ be the number of client-products in the list $N(\zeta_j)$ and Γ^s be the number of client-products in the list $N(\zeta_j)$ that are greater than the largest compartment. In Step 1 of Algorithm 3, we start by quick identification of sufficient conditions of unfeasibility, this step is performed by the tank-truck resource constraints introduced in Section 5.2.1. In Step 2, we start by sorting the $L_{k(\zeta_j)}$ in descending order, then we loop as much as $\Gamma > 0$. We sort the list $N(\zeta_j)$ in descending order. We pick the client-product $\hat{n}$ with the highest value from $N(\zeta_j)$. If the $\hat{n}$ does not fit to the largest compartment $\hat{l}$ from the list $L_{k(\zeta_j)}$, then we split $\hat{n}$ to $d^{''}$ and $C_{\hat{l}k(\zeta_j)}$, we insert the result $d^{''}$ in $N(\zeta_j)$. If a client-product cannot be assigned to a compartment, then we return false.

5.3.4 Dominance rules

To keep the number of labels as small as possible, it is decisive to perform a dominance step for eliminating unneeded labels. A label ζ_1 dominates a label ζ_2 if each resides at the same vertex and if, for every possible extension of ζ_2, there is a feasible extension of ζ_1 wherever the value of every cardinally scaled resource is a smaller than or up to the value of the resource within the extension of ζ_2, and wherever the value of every nominally scaled resource is up to the worth of the resource within the extension of ζ_2. Dominated labels need not be extended. A label that is not dominated by the other label is called undominated or Pareto-optimal. In this paper, we use two phases of dominance, each one has different rules to follow:

Phase 1 is a heuristic dominance formulation. ζ_1 dominates a label ζ_2 if and only if :

$$\overline{C}(\zeta_1) \leq \overline{C}(\zeta_2) \quad (41)$$
$$r^m(\zeta_1) \leq r^m(\zeta_2) \quad \forall m \in R \quad (42)$$
$$s(\zeta_1) \leq s(\zeta_2) \quad (43)$$

Phase 2 is an exact dominance formulation. ζ_1 dominates a label ζ_2 if and only if :

$$\overline{C}(\zeta_1) \leq \overline{C}(\zeta_2) \quad (44)$$
$$r^m(\zeta_1) \leq r^m(\zeta_2) \quad \forall m \in R \quad (45)$$
$$U(\zeta_1) \subseteq U(\zeta_2) \quad (46)$$

Algorithm 3: TTLP-Heuristic(ζ_j)

$\Gamma \leftarrow |N(\zeta_j)|$, $\Gamma^s \leftarrow \{n \in N(\zeta_j) \mid o_n > \max_{l \in L_{k(\zeta_j)}} C_{lk}\}$

Step 1:

if $\Gamma^s > |L_{k(\zeta_j)}|$ ***or*** $\sum_{n \in N(\zeta_j)} o_n > \Delta_{k(\zeta_j)}$ **then** No feasible solution exists: **return** *False*

Step 2:

Sort $L_{k(\zeta_j)}$ in descending order of their capacities.
while $\Gamma > 0$ **do**
  Sort $N(\zeta)$ in descending order.
  $\hat{n} \leftarrow$ first element in $N(\zeta)$ **and** $N(\zeta) \leftarrow N(\zeta) \setminus \hat{n}$
  $\hat{l} \leftarrow$ first element in $L_{k(\zeta_j)}$ **and** $L_{k(\zeta_j)} \leftarrow L_{k(\zeta_j)} \setminus \hat{l}$
  if $o_{\hat{n}} > C_{\hat{l}k(\zeta_j)}$ **then**
    Split the client-product $\hat{n}$: $d'' \leftarrow (o_{\hat{n}} - C_{\hat{l}k(\zeta_j)})$
    Insert the new client-product $N(\zeta) \leftarrow N(\zeta) \cup d''$.
    $\Gamma \leftarrow \Gamma + 1$
  else
    if *$\hat{n}$ can not be assigned to the compartment $\hat{l}$* **then**
      return *False*
    else
      Assign the client-product $\hat{n}$ to $\hat{l}$
      $\Gamma \leftarrow \Gamma - 1$
return *True*

5.4 Integer solutions by primal diving heuristic.

As we mentioned earlier, to derive integer solutions from the RMP, we used a method denoted by Primal Diving Heuristic (PDH). Diving Heuristic (DH), as defined in [28], is a heuristic of the Branch-and-Price Heuristic method which consists in deeply branching until finding an integer solution by branching on the variable with the highest fractional value. DHs are generic ways of repairing infeasibilities. The RMP that remains after a rounding operation must be cleaned up before re-optimization, deleting all columns that could not be part of an integer solution to the residual problem (and hence would lead to infeasibility if selected). Such preprocessing is a key feature in diving heuristics. It helps to avoid the primal heuristic dead-ending with an unfeasible solution. In this context, one should generate so-called proper columns, i.e., columns that could take in a non-zero integer value in an optimal solution to the RMP.

The DH method has no information about any integer solution. The idea of the PDH is that we reinforce the DH method by adding a new type of branching rule. This branching rule is related to the family of local search methods (see [28]). We use the information provided by the current integer solution to choose the variable on which to branch. This variable gives the minimum difference between its value in current integer solution and fractional solution from the set of support of the integer solution variables equal to 1 in the current integer solution. Algorithm 5 gives the pseudo-code of the method.

We start by finding an initial fractional and integer solution, denoted by x^{LP} and x^{IP} respectively. If the gap between x^{LP} and x^{IP} is smaller than a given value $Lgap$, we stop. Otherwise we attempt to find a better integer solution by performing the branching method. A description of the generic algorithm of Branch-and-Price is presented by [29]. Algorithm 4 shows the solution of a branch and bound node with the CG.

Algorithm 4: SolveNode$(\eta, \mathbf{\Omega})$

```
// η is the node number
// Ω is a set of columns
c̄ ← −1;
while c̄ < 0 do
    (x, α) ← Solve RMP // x is the solution, α is the vector of duals
    foreach k ∈ K and d ∈ D do
        // Ω_kd : is the set of columns generated by CGSP^kd
        // c̄_kd : is the minimum reduced cost of columns in Ω_kd
        (c̄_kd, Ω_kd) ← Solve CGSP^kd(α)
        Ω ← Ω ∪ Ω_kd
        c̄ ← min{0, c̄_kd}
return (x)
```

Initial solutions. The Restricted Master Heuristic suggested by [30] searches for an integer feasible master solution by restricting the formulation again to a subset of promising variables, hence regarding a problem which is of considerably smaller size than the current master formulation. We use this approach as described in Algorithm 5 to find an initial fractional and integer solutions. We solve the root node to optimality to obtain the fractional initial solution $x^{LP} = \text{SolveNode}(0, \Omega)$. Then we solve the RMP as a MIP using only the columns generated during the solution of the root node of the Branch-and-Price method to get an initial integer solution x^{IP}.

Algorithm 5: PDH

```
1  Initial solutions:
2  Initialize Ω with initial set of columns
3  x^LP ← SolveNode(0, Ω) // Defined in Algorithm 4.
4  x^IP ←Solve RMP as a MIP
5  Branching:
6  η ← 1, zNotI ← 0
7  while GAP(x^LP, x^IP) ≤ Lgap or zIpNotI ≤ maxZIpNotI do
8      nbVars ← 0
9      while nbVars ≤ nbMaxFVars do
10         i' ← arg min_{i|x_i^IP=1} (x_i^IP − x_i^LP) and set value x_i' = 1
11         Cleanup and preprocessing
12         x^LP ← SolveNode (η, Ω)
13         if x^LP is Integer and Z(x^LP) < Z(x^IP) then
14             x^IP ← x^LP, zIpNotI ← 0
15         else if x^LP is Integer and Z(x^LP) ≥ Z(x^IP) then
16             zIpNotI ← zIpNotI + 1
17         η ← η + 1, nbVars ← nbVars + 1
18     x_M^IP ←Solve RMP as a MIP
19     if Z(x_M^IP) < Z(x^IP) then x^IP ← x_M^IP, zIpNotI ← 0
20     else zIpNotI ← zIpNotI + 1
21 return (x_IP)
```

Branching. In line 6, we initialize a node counter $\eta = 1$ and the number of the integer objective value not improved $zIpNotI = 0$, then we loop (line 7) as long as the gap$\leq Lgap$ or $zIpNotI \leq maxZIpNotI$ (where $maxZIpNotI$ is a parameter that gives the maximum of not improved objective value allowed). In this loop we initialize in line 8 the number of fixed variables ($nbVars = 0$) before solving the RMP as a MIP. We loop in line 9 as long as the number of fixed variables do not exceed $nbMaxFVars$ which is the maximum number of variables to be fixed before the call to MIP. We choose the branching variable and fix his value to 1 (line 10). We create a new branch and bound node (η). In line 11, we perform the cleaning and the preprocessing, then we solve the new node in line 12. If the solution is an integer (line 13), then we compare the objective value Z of the new solution with the integer current solution and update the counter $zIpNotI$ in line 14. If the solution is better, we update the current integer solution x^{IP} (line 14). We increase the node and $nbVars$ counter and $nbVars$ in line 17. After fixing a $nbMaxFVars$ we solve in line 18 the RMP as a MIP and consequently we update the current solution in line 19 if the objective is better than the current objective. Else, we increase $zIpNotI$ in line 20.

5.5 Acceleration strategies

We apply many acceleration strategies to reduce the runtime of the PDH in the CGSPs. We start by defining the network reduction scheme. Then we present the activation and deactivation of arcs in the CGSPs.

Network Reduction. Based on the following elements we define an *acyclic network* G^a of the clients :

- Country divided into four geographic regions;
- All the depots are located in the western region;
- Marine stations are located in the western seafront;
- Movements of tank-trucks : West →East, West → North, West → Center-West and West → South;
- Configuration of the actual road network;
- Pruning of arcs between and inside some regions: Logical pruning is based on the actual road network and the predefined directions of movement of tank-trucks.
- Aggregation of some bakeries nodes geographically close into super nodes. Each bakery order is around 1000 liters while the orders from gas stations are tens times higher;

Activation and deactivation of arcs We start with the reduced network G^a. For each iteration, we activate a maximum n^a of the first best arcs (from the original network G) with negative reduced cost and we deactivate a maximum n^a arcs with positive reduced cost. We stop when no arcs are activated or deactivated. We modify the Algorithm 4 to consider the arcs activation and deactivation strategies given in Algorithm 6.

Algorithm 6: SolveNodeArsActivation$(\eta, \boldsymbol{\Omega})$

```
c̄ ← −1;
if η = 0 then
    Active all arcs of G^a
while c̄ < 0 do
    (x, α) ← Solve RMP
    Sort arcs by reduced cost
    Select a maximum of n^a of non activated arcs with negative reduced cost and active them.
    Select a maximum of n^a of activated arcs with positive reduced cost and deactive them.
    foreach k ∈ K and d ∈ D do
        (c̄_kd, Ω_kd) ← Solve CGSP^kd(α)
        Ω ← Ω ∪ Ω_kd
        c̄ ← min{0, c̄_kd}
return (x)
```

6 Computational results

In this section, we present computational experiments on a benchmark of 29 real-world data-sets provided by PC from Senegal. At the beginning, we go through characteristics of the instances featuring our test benchmark. Then, we report and discuss the computational results obtained from these experiments. Finally, we compare our results with those provided by the current solution approach used by PC.

All algorithms have been implemented using the C ++ language. The optimization runs were performed on a machine equipped with an Intel (R) Core (TM) i7-6700 CPU at 3.40 GHz using a Linux operating system. All Linear Programs and Mixed Integer Programs are solved by the IBM CPLEX Commercial Solver (version 12.9.0.0). We use the OsiSolverInterface (version 0.107.9) framework to communicate with CPLEX solver via C++ language. All the CGSPs are ESPPRCs, solved by dynamic programming using the Boost library (version 1.57.0) [31]. The computational times are in seconds. For the test settings, we use the following parameters: $Lgap = 0.5\%$, $nbMaxFVars = 3$ and $maxZIpNotI = 3$ and $n^a = 5000$.

6.1 Instances characteristics

The 29 real-world instances available correspond to 29 planning weeks chosen from two years (2015-2016) of data shared gratefully by the PC. The latter covers a network of 78 distinct locations. The locations consist in 5 depots, 13 gas stations, 11 marine stations, 7 enterprises, 42 bakeries. First, in Section 6.1.1, we describe the planning weeks. Next, in Section 6.1.2, we give the characteristics of the tank-trucks used.

6.1.1 Planning weeks

Table 3 shows the instances and their characteristics, namely (from left to right), instance type, instance name (Inst.), the week from which it was generated, the number of client-products (#CP), the total number of stations (#Stat.), the number of gas stations it contains (#Gas), the number of marine stations (#Mar), the number of bakeries (#Bak), the number of enterprises (#Entr). We also give the volume of unmarked products (UnPr) and the volume of marked products (MaPr). Note that the volume of products is in cubic meter. The number of client-products is between 38 and 68 while the number of stations varies from 31 to 54.

Based on the number of feasible routes (Figure 2), the instances are divided into the following three groups: *small* size instances with less than 2 millions of feasible routes, *medium* size instances between 2 millions and 7 millions and *large* size instances with more than 7 millions of possible routes. Note that for the five large instances (W24, W25, W26, W27, W29), we are not able to generate all feasible routes, we give the value 5×10^7 just to represent the maximum that we can generate using the memory available.

6.1.2 Tank-trucks description

Table 3 presents the heterogeneous tank-trucks database and their characteristics, namely (from left to right), the id of tank-truck (Id), the total capacity (Ca.), the configuration of the compartments in order (Config.) where each digit represents a compartment multiplier (the capacity of the compartment is 1000l $\times$ this multiplier), the number of compartments it contains (Co.), the last two columns indicate if the tank-truck is labeled 'Fishing' or not (F.) or jumbo (J.). To summarize, the database contains 36 tank-trucks with their capacities varying between 13000 and 40000 liters. Each tank-truck contains from 3 to 15 compartments of capacities variying between 1000 and 10000 liters. We have 11 tank-trucks labeled 'Fishing' and 21 Jumbo.

Table 3: Overview of the set of instances.

Type	Inst.	Week	#CP	#Stat.	#Gas	#Mar	#Bak	#Entr	UnPr	MaPr
Small	**W1**	14-12-2015	41	33	16	2	12	3	18233	8029
	W2	04-01-2016	49	36	17	8	10	1	18250	8098
	W3	01-02-2016	46	33	16	6	8	3	18263	8100
	W4	22-06-2015	47	39	20	5	13	1	18285	8065
	W5	19-10-2015	48	31	18	2	10	1	18272	8040
	W6	18-01-2016	49	37	16	6	12	3	18293	8092
	W7	18-01-2015	50	38	16	6	13	3	18303	8092
	W8	02-11-2015	47	37	22	3	6	6	18310	8054
	W9	24-11-2015	51	34	21	2	9	2	18325	8034
	W10	16-11-2015	58	44	20	7	15	2	18299	8136
	W11	02-05-2016	56	44	20	4	17	3	18312	8075
	W12	14-03-2016	43	32	15	5	9	3	18246	8069
	W13	18-04-2016	68	54	18	7	24	5	18275	8124
	W14	05-04-2016	45	33	16	8	6	3	18247	8099
Medium	**W15**	16-05-2016	44	36	13	6	14	3	18250	8089
	W16	21-12-2015	51	39	18	6	15	0	18253	8083
	W17	09-11-2016	56	44	18	6	19	1	18287	8094
	W18	09-05-2016	53	42	18	8	14	2	18303	8117
	W19	29-03-2016	50	39	14	6	17	2	18228	8080
	W20	15-02-2016	46	35	14	7	12	2	18222	8090
	W21	12-10-2015	49	40	16	3	20	1	18222	8073
	W22	25-01-2016	52	39	16	5	18	0	18267	8083
	W23	08-02-2016	63	47	19	8	17	3	18306	8121
Large	**W24**	11-04-2016	67	49	23	7	17	2	18326	8100
	W25	11-01-2016	43	34	11	6	14	3	18149	8070
	W26	21-03-2016	64	50	18	9	23	0	18288	8143
	W27	11-06-2016	44	34	11	6	14	3	18150	8070
	W28	28-12-2015	52	39	17	3	17	2	18245	8062
	W29	30-11-2015	38	31	8	3	18	2	18113	8057

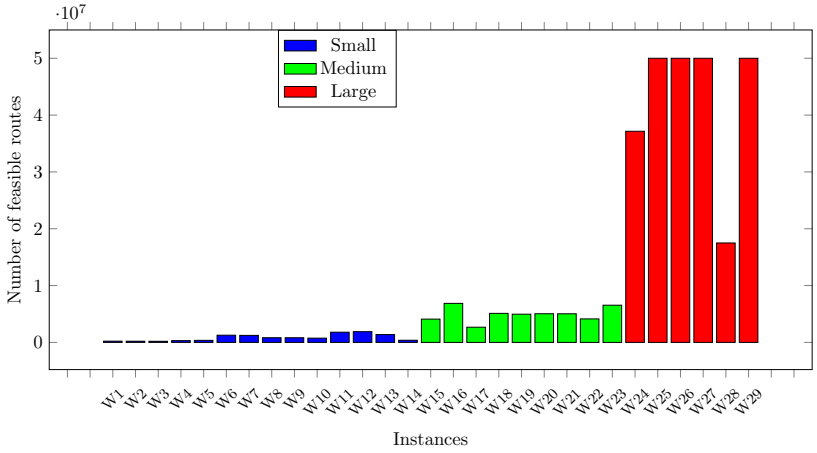

Figure 2: Distribution of feasible routes on the instances.

6.2 TTLP solving

TTLP Results. To better understand the TTLP, we report in Table 5, the number of distinct TTLP-MIPs solved (#MIP); this number include those solved in the preprocessing phase (in which we solved all the combination of 1 to 2 client-products for each tank-truck to determine if we will add the corresponding arc to the graph or not), the number of call to already solved TTLP-MIP (#CAS), the maximum time to solve one TTLP-MIP (MaxT(s)), the total time spent in solving TTLP-MIPs (T(s)) and the percentage of solving TTLP in total time (P(%)). From this table, we observe that the percentage of time consumed in solution of TTLP decreases when the size of instances increases (32%

Table 4: Tank-trucks properties.

Id	Ca.	Config.	Co.	F.	J.	Id	Ca.	Config.	Co.	F.	J.
1	13	3-2-2-3-3	5	☐	☐	**19**	21	2-3-1-1-2-3-5-4	8	☐	☑
2	13	3-1-1-1-1-2-2-2	8	☐	☐	**20**	22	2-2-1-3-2-1-2-3-2-2-2	11	☐	☑
3	13	1-2-4-6	4	☑	☐	**21**	30	7-7-2-7-7	5	☐	☑
4	13	2-1-4-6	4	☑	☐	**22**	33	5-4-3-2-1-1-2-2-3-4-6	11	☐	☑
5	14	2-1-1-4-4-2	6	☐	☐	**23**	33	4-5-4-3-2-3-2-5-5	9	☐	☑
6	14	4-2-1-1-4-2	6	☐	☐	**24**	33	5-4-3-2-1-1-2-2-3-4-6	11	☐	☑
7	14	2-1-1-1-1-2-3-3	8	☑	☐	**25**	33	5-3-4-1,5-1,5-1-1-2-3-2-4-3-2	13	☑	☑
8	14	4-2-3-2-3	5	☑	☐	**26**	35	6-4-3-1-1-2-5-6-7	9	☑	☑
9	14	4-2-3-2-3	5	☐	☐	**27**	35	6-4-5-2-2-2-3-8-3	9	☑	☑
10	14	4-2-3-2-3	5	☐	☐	**28**	35	4-6-2-4-3-4-3-6-3	9	☐	☑
11	14	4-3-7	**3**	☐	☐	**29**	36	4-2-6-2-4-2-3-6-3	9	☐	☑
12	18	5-2-2-2-3-4	6	☐	☐	**30**	37	10-3-4-2-5-6-7	7	☐	☑
13	18	3-2-2-3-2-6	6	☐	☐	**31**	37	6-3-6-4-6-6-6	7	☑	☑
14	18	2-2-2-2-1-1-1-1-3-3	10	☐	☐	**32**	37	5-2-5-2-2-1-1-2-2-2-2-5-2-2-2	**15**	☑	☑
15	18	2-1,5-1-1-1,5-4-4-3	8	☐	☐	**33**	38	8-4-3-2-2-1-1-2-5-4-6	11	☑	☑
16	19	5-1-1-5-5-2	6	☐	☐	**34**	38	6-4-6-2-3-5-4-2-6	9	☐	☑
17	20	6-5-5-2-2	5	☐	☑	**35**	38	6-4-1-1-4-2-4-6-4-3-5	11	☑	☑
18	20	2-4-3-4-5-1-1	7	☐	☑	**36**	40	6-4-1-1-4-2-4-6-4-3-5	11	☐	☑

on average for the small instances, 19% for the medium ones and 6% for the largest). This is justified by the number of TTLP-MIPs solved in the first iteration. We solve 99% of the TTLP-MIPs in the first iteration. In the next iterations, we just use the output of the already solved TTLP-MIPs in the majority of cases. We also observe that #CAS increases with the size of instances (3.8E+06, 1.0E+07, 7.3E+07, for the small, medium and large instances, respectively).

In the EF (see Section 5.3.1), for some cases we solve a TTLP-MIP to check if a label is feasible or not which takes a maximum of 0.11 seconds on average on the medium instances. The time needed to solve all the TTLP-MIPs is on average ranging from 2 seconds (reported in the small instances) to 34 seconds (observed in the large ones). The number of TLLP-MIPs solved compared to the value of #CAS is insignificant. It represents less than 0.50%. on average, we call one distinct TTLP-MIP 640 times for the small instances, 2254 and 13308 times, for the medium and large ones, respectively. In the following paragraph, we explain the impact of the pattern learning technique.

Impact of TTLP Hashing. To evaluate the impact of TTLP Hashing we ran the PDH method with and without TTLP hashing on all instances with a time limit of 1 hour. As mentioned in Section 5.3.1, without the hashing TTLP, the same TTLP problem is solved many times. The results obtained are as follows: (i) PDH with TTLP hashing solves all instances in less than 417 seconds, (ii) PDH without TTLP hashing cannot solve any instance in less than one hour (there are instances that are not solved even after 6 hours). Figure 3 gives an overview of the progress of the two methods on instance **W3**. First, we report in Figure 3a, the evolution of the number of TTLPs solved at each iteration without using TTLP hashing. Second, in Figure 3b we report the evolution of the number of TTLPs solved at each iteration using TTLP hashing.

We observe that the number of TTLPs solved in the first case is very big, jumping from 123418 in the first iteration to 919285 in the last iteration with an augmentation of more than 100000 at each iteration. For the second case, the number of TTLPs solved is 75 times smaller and stable (varying between 1636 and 1638). Note that in the two cases we obtain the same solution. In the first case, the time needed is more than one hour (3702 seconds) while it is 16 seconds in the second case. We observe that using TTLP hashing is 231 times faster than not using TTLP hashing. Therefore, we use TTLP hashing in the following sections.

Impact of TTLP-Heuristic. Table 5 reports, for each instance, the number of calls to the heuristic (#calls) and the percentage of success (Success(%)). Note that the number of TTLP-MIPs solved is not necessary less than the value #Calls because we include the TTLP-MIPs that we solve in the preprocessing phase. We observe that TTLP-heuristic performs well in the same way for the small

Table 5: TTLP statistics of PDH.

		TTLP					TTLP-Heuristic	
Type	**Inst.**	**#MIP**	**#CAS**	**MaxT(s)**	**T(s)**	**P(%)**	**#Calls**	**Success(%)**
Small	**W1**	2 904	1,1E+06	0,05	5	34	2391	36
	W2	3 508	8,3E+05	0,04	5	33	3174	42
	W3	4 265	8,6E+05	0,05	7	42	4211	39
	W4	2 967	2,1E+06	0,04	2	13	1711	47
	W5	5 828	1,5E+06	0,05	11	43	6313	41
	W6	4 241	4,3E+06	0,04	6	22	3419	38
	W7	4 101	4,0E+06	0,04	5	18	3062	42
	W8	6 847	4,5E+06	0,05	18	41	7223	29
	W9	7 989	4,1E+06	0,05	19	42	8732	33
	W10	7 517	5,2E+06	0,04	15	33	7831	38
	W11	8 598	4,3E+06	0,06	22	48	8366	25
	W12	8 407	8,4E+06	0,06	17	32	9326	34
	W13	6 513	9,4E+06	0,05	11	21	7001	40
	W14	5 858	2,3E+06	0,06	9	31	5110	38
Avg		**5 682**	**3,8E+06**	**0,05**	**11**	**32**	**5562**	**37**
Medium	**W15**	2 340	9,5E+06	0,03	2	5	1316	49
	W16	5 652	5,4E+06	0,04	8	25	5309	39
	W17	6 122	6,4E+06	0,08	11	27	6240	36
	W18	8 301	7,4E+06	0,05	14	31	8282	32
	W19	3 898	1,1E+07	0,04	5	12	3562	40
	W20	6 641	9,5E+06	0,05	13	23	7227	39
	W21	3 596	1,6E+07	0,63	6	11	2798	37
	W22	5 567	1,3E+07	0,04	8	14	4361	32
	W23	6 746	1,6E+07	0,05	15	22	7150	31
Avg		**5 429**	**1,0E+07**	**0,11**	**9**	**19**	**5138**	**37**
Large	**W24**	7 376	4,8E+07	0,07	15	8	8004	37
	W25	4 973	4,8E+07	0,04	10	6	6430	44
	W26	6 445	6,5E+07	0,06	9	4	5386	37
	W27	5 878	8,1E+07	0,05	12	4	7343	40
	W28	12 786	7,8E+07	0,05	34	11	15740	32
	W29	3 441	1,2E+08	0,05	6	1	3608	41
Avg		**6 817**	**7,3E+07**	**0,05**	**14**	**6**	**7752**	**38**

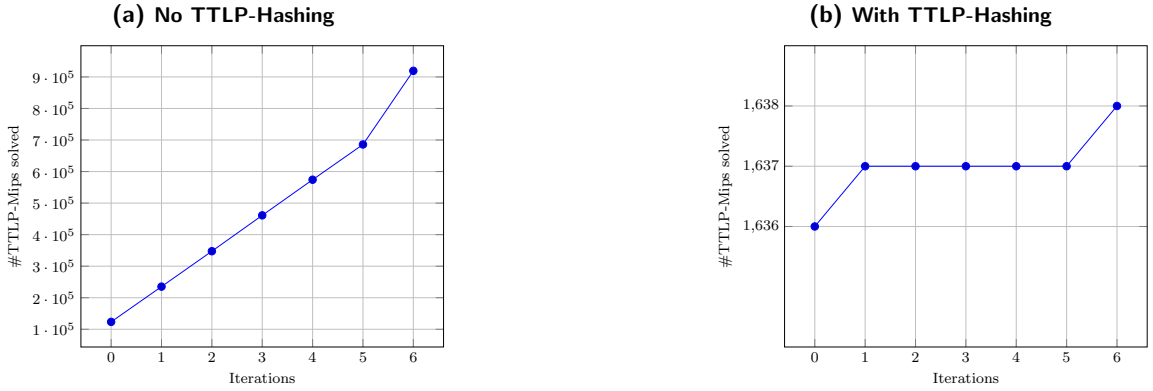

Figure 3: Impact of TTLP-Hashing on instance W3.

and the medium instances. It succeeds to solve the TTLP in 37% of cases on average. It gives better performance for the large instances (+1% on average). The TTLP-Heuristic is a polynomial algorithm while solving the TTLP-MIP is not polynomial. For that, this percentage of success helps the EF to be faster than solving the TTLP-MIP all to time.

6.3 PDH and DH performance testing

The goal of this section is to demonstrate that PDH finds excellent solutions in very short times. We provide a comparison between PDH and DH and give performance analysis. In Table 6, we report the optimal value of the linear relaxation (LP) obtained at the root node followed by the root node time (RT). These values are the same for the two methods (PDH and DH). For each method, we report the integrality gap (GAP) in percentage and the total runtime (T). In addition, we report three types of information: the number of generated columns (#cols.), the number of iterations (It.) and the number of branch and bound nodes (No.). We can see that all the instances are solved by PDH in smaller computing time. The average time for the large instances is 274 seconds and the largest one is 417. The DH fails to find a feasible solution for two instances W22 and W28. We observe that PDH outperforms the DH in all the instances.

The PDH time is 5 times faster (we compare T-RT) on average for the small instances, 8 times for the medium and 38 times for the large instances. This is justified by the number of branching nodes that needs. It needs 8 times less nodes on average for the small instances, 9 and 7 for the medium and large ones. It needs 3 times less iterations on average for all instances. We stop after finding the initial integer solution in all instances except for W28 and W29 for each we solve 9 and 10 nodes to find a better solution than the initial integer solution.

Table 6: Runtime and solution quality.

		Root Node		*PDH*					*DH*				
Type	**#Inst**	**LP**	**RT(s)**	**GAP(%)**	**T(s)**	**#Cols**	**It.**	**No.**	**GAP(%)**	**T(s)**	**#Cols**	**It.**	**No.**
Small	**W1**	57033	9	0,37	14	4543	9	1	0,93	16	4558	13	2
	W2	168111	8	0,03	14	5469	8	1	0,38	20	5494	23	6
	W3	173060	11	0,06	16	6099	7	1	0,58	27	6198	34	8
	W4	142765	9	0,01	17	4727	13	1	0,03	23	4745	25	4
	W5	57934	18	0,09	26	7331	7	1	0,12	35	7362	18	5
	W6	173114	20	0,49	27	6739	12	1	1,05	80	7030	70	14
	W7	174047	18	0,44	26	6141	12	1	0,48	64	6238	38	8
	W8	77341	36	0,12	43	7052	9	1	1,74	107	7476	56	11
	W9	57546	36	0,41	44	8747	9	1	0,79	112	8994	48	12
	W10	197748	35	0,02	45	9131	11	1	0,02	73	9215	30	6
	W11	82706	36	0,19	45	6723	10	1	0,6	89	6836	39	9
	W12	141953	45	0,04	53	7161	13	1	0,14	97	7304	34	5
	W13	148299	46	0,14	55	9966	18	1	0,75	93	10124	44	7
	W14	153104	18	0,05	27	8842	9	1	0,54	52	9078	34	8
		Avg	**25**	**0,18**	**32**	**7048**	**11**	**1**	**0,58**	**63**	**7189**	**36**	**8**
Medium	**W15**	122726	30	0,12	36	4352	17	1	0,65	66	4356	30	7
	W16	153645	23	0,16	32	6475	14	1	0,56	57	6581	43	8
	W17	138419	32	0,25	40	8482	12	1	0,71	85	8620	42	9
	W18	189546	36	0,07	46	7842	13	1	0,29	89	7946	49	10
	W19	163038	40	0,06	46	7410	16	1	0,56	115	7793	57	8
	W20	162834	48	0,34	57	7720	14	1	1,16	128	7802	47	11
	W21	92952	50	0,13	57	6169	23	1	0,13	57	6169	23	1
	W22	132679	47	0,29	57	6188	16	1	-	150	6282	51	11
	W23	199063	64	0,14	71	8281	15	1	0,75	187	8922	64	12
		Avg	**41**	**0,17**	**49**	**6991**	**16**	**1**	**0,60**	**104**	**7163**	**45**	**9**
Large	**W24**	178067	180	0,07	190	12454	17	1	0,7	756	12910	74	14
	W25	157705	161	0,1	167	7511	26	1	0,65	559	7785	84	11
	W26	203309	248	0,27	259	9548	15	1	0,35	991	9674	51	8
	W27	157712	301	0,11	308	7663	32	1	0,45	554	7887	72	7
	W28	76940	149	0,71	303	10130	35	9	-	329	10415	60	8
	W29	71564	293	0,59	417	6230	44	10	1,76	391	6369	54	5
		Avg	**222**	**0,31**	**274**	**8923**	**28**	**4**	**0,78**	**597**	**9173**	**66**	**9**

The DH generates more columns compared to the PDH (from 140 to 250 on average for each type of instances) as a result of solving more branch and bound nodes. The solution quality measured

by the integrality gap of the PDH is 5 times better than DH. When we analyze the solutions of the two methods, we find that this is to the DH solutions that have more empty space on average (see Section 6.5).

PDH detailed performance statistics. We give the detailed performance statistics of the PDH method. Table 7 shows from left to right, the instance type, the instance name (Inst.), the total number of labels (#Lbs), the total number of calls to dominance function (#DC), the time of CGSPs (CGSPt(s)) and the time of MP (MPt(s)). The first thing we can observe that the time of MP solution is very small compared to the time consumed to solve the CGSPs for all instances, it represents at most 3,57 seconds while the CGSP take 416,03 seconds for the instance W28. It is due to the complexity of the CGSPs related to the complex loading constraints and the elementary routes. We can observe an important number of created labels (ranging 2,1E+06 on average for the small instance, 3,5E+06 for the medium instances and 1,2E+07 for the large instance).

Table 7: Performance statistics of PDH.

Type	Inst.	#Lbs	#DC	CGSPt(s)	MPt(s)
Small	**W1**	9,3E+05	7,2E+05	13,94	0,06
	W2	8,4E+05	6,1E+05	13,77	0,23
	W3	7,6E+05	5,8E+05	15,79	0,21
	W4	1,5E+06	1,1E+06	16,93	0,07
	W5	1,3E+06	9,9E+05	25,84	0,16
	W6	2,1E+06	1,7E+06	26,46	0,54
	W7	2,1E+06	1,7E+06	25,56	0,44
	W8	2,5E+06	2,1E+06	42,63	0,37
	W9	2,4E+06	2,0E+06	43,35	0,65
	W10	2,9E+06	2,4E+06	44,60	0,4
	W11	1,9E+06	1,6E+06	44,53	0,47
	W12	3,5E+06	3,0E+06	52,90	0,1
	W13	5,2E+06	4,2E+06	54,57	0,43
	W14	1,8E+06	1,4E+06	26,44	0,56
	Avg	**2,1E+06**	**1,7E+06**	**31,95**	**0,34**
Medium	**W15**	2,7E+06	2,3E+06	35,91	0,09
	W16	2,1E+06	1,6E+06	31,70	0,3
	W17	2,9E+06	2,4E+06	39,73	0,27
	W18	2,5E+06	2,0E+06	45,61	0,39
	W19	4,1E+06	3,4E+06	45,72	0,28
	W20	3,7E+06	3,1E+06	56,01	0,99
	W21	4,2E+06	3,5E+06	56,94	0,06
	W22	4,4E+06	3,7E+06	56,52	0,48
	W23	5,3E+06	4,5E+06	70,57	0,43
	Avg	**3,5E+06**	**3,0E+06**	**48,75**	**0,37**
Large	**W24**	1,2E+07	1,1E+07	188,78	1,22
	W25	1,1E+07	9,8E+06	166,35	0,65
	W26	1,3E+07	1,2E+07	258,60	0,4
	W27	1,8E+07	1,6E+07	307,12	0,88
	W28	2,2E+07	2,0E+07	299,43	3,57
	W29	2,4E+07	2,2E+07	416,03	0,97
	Avg	**1,7E+07**	**1,5E+07**	**272,72**	**1,28**

6.4 PDH acceleration strategies

In this section, we analyze the impact of acceleration strategies on the PDH method, denoted by PDHAA. In the Table 8, we report similar information given in Table 6 but this time for the PDHAA. In addition, we report the percentage of reduction by adding the prefix (r.) obtained for PDHAA vs PDH. From these results, we can see that we conserve the same solution quality, we have a small augmentation of the gap and the number of nodes (an increase of 29% on the small instances but this represents a 0.01% of the gap). PDHAA needs more iterations (24% to 208 iterations on average).

This is justified by the additional iterations needed by the activation and deactivation of arcs. These additional iterations do not augment the runtime because the network is reduced. The impact of acceleration strategies is shown on the runtime and especially for the large instances (we save 38% of time on average).

Table 8: PDH acceleration strategies results.

		PDHAA				*PDHAA vs PDH (%)*			
Type	**#Inst**	**GAP(%)**	**T(s)**	**Iter.**	**#Nds**	**r.GAP(%)**	**r.T(s)**	**r. It.**	**r.#Nds**
Small	**W1**	0,48	14	18	1	30	-7	100	0
	W2	0,05	14	13	1	67	-7	63	0
	W3	0,07	17	17	1	17	6	143	0
	W4	0,01	15	30	1	0	-17	131	0
	W5	0,03	28	18	1	-67	8	157	0
	W6	0,54	34	37	10	10	17	208	900
	W7	0,43	23	21	1	-2	-15	75	0
	W8	0,12	40	16	1	0	-7	78	0
	W9	0,38	48	19	1	-7	9	111	0
	W10	0,09	46	22	1	350	2	100	0
	W11	0,19	46	21	1	0	-4	110	0
	W12	0,03	54	25	1	-25	0	92	0
	W13	0,18	51	28	1	29	-7	56	0
	W14	0,05	29	14	1	0	-3	56	0
	Avg	**0,19**	**32,79**	**21,36**	**1,64**	**29**	**-2**	**106**	**64**
Medium	**W15**	0,1	25	21	1	-17	-31	24	0
	W16	0,13	25	26	1	-19	-29	86	0
	W17	0,29	34	18	1	16	-19	50	0
	W18	0,07	38	26	1	0	-21	100	0
	W19	0,06	34	24	1	0	-23	50	0
	W20	0,33	49	23	1	-3	-9	64	0
	W21	0,13	31	34	1	0	-48	48	0
	W22	0,3	67	24	1	3	14	50	0
	W23	0,13	39	18	1	-7	-51	20	0
	Avg	**0,17**	**38,00**	**23,78**	**1,00**	**-3**	**-24**	**55**	**0**
Large	**W24**	0,08	176	32	1	14	-1	88	0
	W25	0,07	31	19	1	-30	-82	-27	0
	W26	0,28	242	35	1	4	-5	133	0
	W27	0,12	64	28	1	9	-78	-13	0
	W28	0,78	446	86	13	10	26	146	44
	W29	0,71	55	30	7	20	-87	-32	-30
	Avg	**0,34**	**169,00**	**38,33**	**4,00**	**5**	**-38**	**49**	**2**

6.5 Functional results

In this section, we give an overview of functional results of PDHAA, PDH and DH. For each method, we report in Table 9 the number of routes (#Ro) followed by the number of tank-trucks used (#Ta). In addition, we show the value of escort fees (Ef) and the percentage of escort fees saved (Sef(%)) for each instance. Another important characteristic of the solutions is related to the percentage of empty space (Es(%)) in tank-trucks.

As we can see, the value of escort fees saved is the same for all methods. We save an average 8% of escort fees for the small instances. For the medium and large, we save 11% on average. We note that for all instances where the value of escort fees saved is equal to zero, we cannot save escort fees, because we cannot link two marine stations, either because there is no reduction for each pair or the two stations cannot be served by the same tank-truck due to capacity or the compartments configuration. We also observe that the PDH gives the best match between the number of vehicles, the number of routes used in the solution and the empty space for most instances. PDHAA provides a solution close to the PDH giving the same number of routes with a similar number of tank-trucks

used. The DH uses only one more route on average for the small and medium instances. For the large instances, it uses on average the same number of routes.

The DH shows highest values of the empty space, which explains why it finds a solution with a gap (reported in the Table 6) with a higher value than the other methods. The empty space impacts directly the objective value of integer solutions (consequently, the value of the gap increases).

Table 9: Number of routes and number of vehicles.

				PDHAA			***PDH***			***DH***		
Type	**Inst.**	**Ef**	**Sef(%)**	**#Ro**	**#Ta**	**Es(%)**	**#Ro**	**#Ta**	**Es(%)**	**#Ro**	**#Ta**	**Es(%)**
Small	**W1**	55000	0	13	10	3	13	10	3	14	12	4
	W2	185000	11	15	12	2	16	11	2	16	12	4
	W3	180000	6	17	11	1	17	10	1	18	10	5
	W4	140000	0	17	12	1	17	13	1	17	13	1
	W5	55000	0	14	12	2	14	12	2	15	11	2
	W6	195000	13	18	13	6	17	12	5	19	13	8
	W7	195000	13	19	14	11	19	12	8	18	13	9
	W8	90000	17	16	11	2	17	12	2	17	10	6
	W9	55000	0	15	10	2	16	11	2	16	11	2
	W10	195000	0	20	11	1	19	12	0	19	14	0
	W11	90000	11	16	13	3	16	14	3	17	14	4
	W12	140000	0	13	11	0	13	11	0	13	11	0
	W13	165000	12	19	13	3	19	13	3	20	13	6
	W14	185000	19	16	12	3	16	12	3	17	10	5
Avg		**137500**	**7**	**16**	**12**	**3**	**16**	**12**	**3**	**17**	**12**	**4**
Medium	**W15**	120000	0	15	11	1	16	10	1	18	12	4
	W16	175000	14	15	14	4	15	12	4	18	13	6
	W17	160000	16	17	12	4	17	12	4	21	13	6
	W18	220000	16	18	13	6	18	13	6	19	12	7
	W19	175000	9	15	9	2	14	10	2	15	10	5
	W20	170000	6	15	12	4	16	10	3	17	11	9
	W21	105000	14	14	11	5	14	10	5	14	10	5
	W22	145000	10	16	12	3	16	12	3	18	13	9
	W23	230000	15	19	15	3	19	14	3	20	14	7
Avg		**166667**	**11**	**16**	**12**	**4**	**16**	**11**	**3**	**18**	**12**	**6**
Large	**W24**	190000	8	21	12	0	21	12	0	20	12	4
	W25	180000	14	12	10	5	12	10	5	14	10	10
	W26	235000	15	20	14	3	20	14	3	20	16	4
	W27	180000	14	12	10	5	12	11	5	12	11	8
	W28	90000	17	14	12	4	15	10	4	13	10	1
	W29	70000	0	10	7	4	10	7	4	12	7	11
Avg		**157500**	**11**	**15**	**11**	**4**	**15**	**11**	**4**	**15**	**11**	**11**

6.6 Comparison with PC solutions

In this section, we compare the solutions of the PDH to those of PC. The manual solutions (MS) provided by PC are obtained manually by an expert who spends two working days to find a solution for each week of scheduling. Note that for the MS we do not have the information about the order of visiting clients. So, we compare the functional aspect only. For that, we report in Table 10 the same information in Table 9 (RE,#Ro, #Ta, Es(%), Sef(%)) for the MS. In addition, we report for PDH the reduction by adding prefix (r.) obtained for MS vs PDH.

From these results, we observe a significant reduction on the escort fees representing more than 200% on the small instances and 300% on the medium and large size instances. For the number of routes, we observe a reduction of 15% routes on average for all types of instances. Looking at the numbers of vehicles used, one can notice that for the small instances, we have a small variation (±1%) on #Ve. For the empty space, no significant reduction for the medium instances (1% on

average). A significant reduction on the small and large instances are observed, 20% and 28% on average, respectively. Finally, the reduction of runtime to find a good solution to the problem is very significant, it represents 99% (a max of 424 seconds compared to 14 hours of work needed by the PC expert).

Table 10: Comparison MS and PDH solutions.

		MS				***PDH vs MS***			
Type	**Inst.**	**#Ro**	**#Ta**	**Es(%)**	**Sef(%)**	**r.#Ro**	**r.#Ta**	**r.Es(%)**	**r.Sef(%)**
Small	**W1**	17	8	11	0	-4	2	-8	0
	W2	21	10	1	0	-5	1	1	-11
	W3	21	13	2	0	-4	-3	0	-6
	W4	18	8	3	0	-1	5	-2	0
	W5	17	11	2	0	-3	1	0	0
	W6	20	14	4	8	-3	-2	1	-5
	W7	21	14	6	8	-2	-2	2	-5
	W8	19	12	2	0	-2	0	0	-17
	W9	19	12	2	0	-3	-1	-1	0
	W10	24	13	8	0	-5	-1	-8	0
	W11	20	13	3	0	-4	1	0	-11
	W12	18	13	1	0	-5	-2	0	0
	W13	22	13	4	6	-3	0	-1	-6
	W14	20	13	2	5	-4	-1	0	-14
Avg		**20**	**12**	**4**	**2**	**-3**	**0**	**-1**	**-5**
Medium	**W15**	18	10	2	0	-2	0	-1	0
	W16	20	10	1	0	-5	2	3	-14
	W17	19	12	5	9	-2	0	0	-6
	W18	21	12	1	11	-3	1	5	-5
	W19	17	11	0	0	-3	-1	2	-9
	W20	18	12	2	0	-2	-2	2	-6
	W21	19	9	7	0	-5	1	-2	-14
	W22	21	10	7	0	-5	2	-5	-10
	W23	24	12	6	0	-5	2	-3	-15
Avg		**20**	**11**	**3**	**2**	**-4**	**1**	**0**	**-9**
Large	**W24**	24	12	0	0	-3	0	0	-8
	W25	15	9	6	6	-3	1	-1	-8
	W26	24	13	3	4	-4	1	0	-11
	W27	15	9	6	6	-3	2	-1	-8
	W28	18	10	4	0	-3	0	0	-17
	W29	12	7	8	0	-2	0	-3	0
Avg		**18**	**10**	**4**	**3**	**-3**	**1**	**-1**	**-9**

7 Conclusion

In this paper, we present and discuss a real-world of the Multi-Depot Multi-Period Petrol Replenishment Problem of a Petroleum Company located in Senegal. This problem is challenging for its specific complexity related to loading constraints (from 3 to 15 compartments), Multi-Product, the Multi-Depot, the Multi-trip and the clients that can order a small quantity of gas this increases the number of clients included in a single route. Furthermore, some clients may be visited by tank-trucks more than once.

We propose a Branch-and-Price Heuristic to find a near optimal solution to this problem. In this method, we solve the integrated problem of tank-truck loading problem and the routing problem. We use a pattern learning technique to accelerate the generation of routes. The extensive numerical results on a 29 real-world instances show that our approach produces better quality solutions (with 0.31% on the gap on average) than a state-of-the-art Branch-and-Price method and the solutions provided by the PC. These results show the effectiveness and high potential of the proposed approach.

Future developments could address the generalization of the pattern learning technique to the general MC-VRP. We believe that this method will be useful to accelerate the solution of CGSPs with dynamic programming approach. We plan to test our approach on bigger real-world network. With more realistic constraints such as integrating safe stowage, the inventory management with deterministic and stochastic demand and the case with the total supply of gas is less than the total demand.

References

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

[2] G. G. Brown, G. W. Graves, Real-time dispatch of petroleum tank trucks, Management science 27 (1) (1981) 19–32.

[3] G. G. Brown, C. J. Ellis, G. W. Graves, D. Ronen, Real-time, wide area dispatch of mobil tank trucks, Interfaces 17 (1) (1987) 107–120.

[4] L. van der Bruggen, R. Gruson, M. Salomon, Reconsidering the distribution structure of gasoline products for a large oil company, European Journal of Operational Research 81 (3) (1995) 460–473.

[5] D. Ronen, Dispatching petroleum products, Operations Research 43 (3) (1995) 379–387.

[6] D. T. Allah, J. Renaud, F. Boctor, Le problème d'approvisionnement des stations d'essence, Journal européen des systemes automatisés 34 (1) (2000) 11–33.

[7] F. B. Abdelaziz, C. Roucairol, C. Bacha, 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, 2002, pp. 478–483.

[8] P. Hansen, N. Mladenović, Variable neighborhood search: Principles and applications, European journal of operational research 130 (3) (2001) 449–467.

[9] M. Malepart, F. Boctor, J. Renaud, S. Labillois, Nouvelles approches pour l'approvisionnement des stations d'essence, Revue Française de Gestion Industrielle (2) (2003) 15–32.

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

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

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

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

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

[15] D. Popović, N. Bjelić, G. Radivojević, Simulation approach to analyse deterministic irp solution of the stochastic fuel delivery problem, Procedia-Social and Behavioral Sciences 20 (2011) 273–282.

[16] P. Hanczar, A fuel distribution problem–application of new multi-item inventory routing formulation, Procedia-Social and Behavioral Sciences 54 (2012) 726–735.

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

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

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

[20] N. Charusakwong, M. Lohatepanont, Optimization approach for multi-period fuel replenishment, Engineering Journal 20 (5) (2016) 239–261.

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

[22] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, European journal of operational research 59 (3) (1992) 345–358.

[23] G. B. Dantzig, P. Wolfe, Decomposition principle for linear programs, Operations research 8 (1) (1960) 101–111.

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

[25] S. Irnich, G. Desaulniers, Shortest path problems with resource constraints, in: Column generation, Springer, 2005, pp. 33–65.

[26] M. Desrochers, F. Soumis, A generalized permanent labelling algorithm for the shortest path problem with time windows, INFOR: Information Systems and Operational Research 26 (3) (1988) 191–212.

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

[28] C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, F. Vanderbeck, Column generation based primal heuristics, Electronic Notes in Discrete Mathematics 36 (2010) 695–702.

[29] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, P. H. Vance, Branch-and-price: Column generation for solving huge integer programs, Operations research 46 (3) (1998) 316–329.

[30] C. Joncour, S. Michel, R. Sadykov, D. Sverdlov, F. Vanderbeck, Column generation based primal heuristics, Electronic Notes in Discrete Mathematics 36 (2010) 695–702.

[31] Boost, Boost C++ Libraries, http://www.boost.org/, last accessed 2019-03-15 (2019).