Logist. Res. (2016) 9:6
DOI 10.1007/s12159-016-0133-z

ORIGINAL PAPER

A petrol station replenishment problem: new variant and formulation

Abdelaziz Benantar[1] · Rachid Ouafi[1] · Jaouad Boukachour[2]

Received: 20 October 2015 / Accepted: 19 March 2016 / Published online: 11 April 2016

Abstract One of the most important problems in the petroleum industry is the well-known petrol station replenishment problem with time windows, which calls for the determination of optimal routes by using a fleet of tank trucks to serve a set of petrol stations over a given planning horizon. In this paper, we introduce a model and solve a specific problem that originates from a real-life application arising in the fuel distribution where specific attention is paid to tank trucks with compartments and customers with different types of products and time windows. Literally, we call the resulting problem the multi-compartment vehicle routing problem with time windows (MCVRPTW). To solve the MCVRPTW, we begin by describing the problem, providing its mathematical formulation and discussing the sense of its constraints. As the problem is NP-hard, we propose an efficient tabu search algorithm for its solution. We introduce the Kolmogorov–Smirnov statistic into the framework of the tabu search to manage the neighbourhood size. We evaluate the performance of the algorithm on a set of vehicle routing problems with time windows instances as well as other realistic instances. Our results are compared to CPLEX, to the heuristics reported in the literature and also to those extracted from the company plans.

✉ Abdelaziz Benantar
a_benantar@yahoo.fr

Rachid Ouafi
rouafi@usthb.dz

Jaouad Boukachour
jaouad.boukachour@univ-lehavre.fr

[1] Department of Operational Research, USTHB University, P.O. Box 32, 16111 Bab-Ezzouar, Algeria

[2] LMAH, Normandie University, ULH, 76600, Le Havre, France

Keywords Petrol station replenishment · Vehicle routing · Compartments · Time windows · Tabu search

1 Introduction

The specific problem, which will be discussed in this paper, is a variant of the capacitated vehicle routing problem and occurs in the context of fuel distribution. More precisely, the paper deals with the daily replenishment-planning problem that the biggest Algerian petroleum company is facing. The company is faced with a particular problem which is demonstrated by the following procedures in operations. (a) The company has to deliver different fuel types ordered by a set of petrol stations during the next working day. (b) These stations order one or more fuel types each time and specify the quantity to be delivered for each product ordered. (c) The products are incompatible and must be transported in independent vehicle compartments. In addition, petrol stations specify time windows during which they must be served. The company delivers products from one or more depots. (d) Each depot is responsible for the demand of stations located in the same district as the depot. The company assigns a fleet of vehicles to each depot. (e) These vehicles are compartmentalized and do not have flow metres. This implies that the contents of a compartment cannot be used to replenish more than one underground reservoir. Consequently, each compartment of the delivery vehicle must be filled with one of the products to be delivered on its route. In this context, the company prepares a replenishment plan for their petrol stations for the next day. This plan requires a number of simultaneous and interrelated decisions to be made. To prepare such a plan, the company must determine the routes for the delivery of all the demanded products, assign

these routes to vehicles, determine the quantities to be delivered by each route, and load these quantities to the compartments. On this point, it should be noted that, in 2013, an Agreement was concluded between the company and the petrol station managers. According to this Agreement, the quantities loaded in the compartments can be adjusted up to a given threshold. This particular situation occurs quite frequently when the demands of petrol stations are high in winter. The objective of the replenishment plan is to determine a set of routes satisfying all customer orders at a minimal routing and service cost.

Our real-life application can be viewed as a combination of two variants of the vehicle routing problem (VRP): a VRP with multiple compartments (MCVRP) and VRP with time windows (VRPTW). For clarity of exposition, this problem will be called the multi-compartment vehicle routing problem with time windows (MCVRPTW). However, it is different from the problems previously discussed in the literature with respect to the following properties:

1. Besides the common constraints of heterogeneous fleets, multiple compartments and time windows, we incur a penalty cost in connection with the use of private vehicles. This constraint originates from the fact that the company managers would like to promote the use of their own vehicles.
2. Since restrictions on the accessibility of vehicles (see constraints 2 in the problem formulation) are imposed, some vehicles cannot visit some petrol stations, e.g., the petrol stations managed by the army cannot be delivered to by private vehicles.
3. The quantities loaded in the compartments can be adjusted according to an Agreement concluded between the company and the petrol station managers.

Consequently, not only do the vehicle routes have to be determined, but how to maximize the quantities to be delivered also has to be decided for each day.

The problem is NP-hard since it is a combination of the MCVRP and VRPTW, which are known to be NP-hard problems [34]. Furthermore, the MCVRPTW is more complicated than the MCVRP and VRPTW considering that it needs to tackle compartments and time windows constraints simultaneously. Because the practical large-scale MCVRPTW instances are difficult, if not impossible, to tackle efficiently within a reasonable amount of computing time, even by using the most powerful MIP solvers such as CPLEX (see Sect. 5), the purpose of this paper is to propose an effective metaheuristic for the MCVRPTW. To make the implementation simpler, we employ a tabu search as the algorithm framework. Different structural neighbourhood methods are used in the improving phase of the tabu search to broaden the exploration of the search space. Meanwhile, a Kolmogorov–Smirnov statistic (KSS) is incorporated into the framework of the tabu search to manage the neighbourhood size. The main idea of the KSS is that a neighbourhood size is determined according to a probability model that minimizes the distance criterion and decides whether two customers are neighbours or not.

2 Literature review

Several versions of the petrol station replenishment problem have attracted interest over the last three decades. In this section, we present a brief review of the published literature dealing with these versions of the petrol station replenishment problem in a chronological order.

One of the first articles was published by Brown and Graves [6], who developed an automated real-time dispatch system for the distribution of petroleum products for a major US oil company. Each order includes several gasoline products, jointly constituting a full truckload. Brown and Graves proposed a model to assign orders to trucks. The objective was to minimize the sum of travel costs and establish a penalty for trucks that exceed the allowable working hours per day, as well as those that had less than the required minimum working hours.

Brown et al. [7] developed a computerized assisted dispatch system for Mobil Oil Corporation in the United States. The dispatching procedure used by the system was an extension of the one presented by Brown and Graves [6] and allowed visiting more than one customer per trip.

Franz and Woodmanse [17] developed a rule-based semi-automated decision support system for a regional oil company to determine the daily schedule of the drivers and the dispatching of the tank trucks.

Ronen [27] studied the dispatching problem. The main concern was to set up a timely and economic delivery of petroleum products and/or liquid chemicals by a fleet of vehicles. He presented three models, the set partitioning model, the elastic set partitioning model and the set packing model, and showed how they could be used in a petrol product distribution system. Bausch et al. [2] proposed an elastic set partitioning technique to solve the same problem. The candidate schedules are obtained by generating trips using the sweep algorithm. Also, in 1995, Van der Bruggen et al. [36] solved the single period version of the problem as part of a broader study aimed at optimizing the distribution network of a large oil company operating in the Netherlands. They suggested some simple models to assign customers to depots, to determine the fleet size and composition and to restructure the depot network.

Nussbaum and Sepulveda [24] addressed the distribution problem for the biggest fuel company in Chile. The delivery plans are obtained using a heuristic approach and a planning, execution and control system is developed employing a

knowledge-based approach that utilizes a graphical user interface, which mimics the mental model of the user.

Several greedy heuristics followed by simple improvement procedures for the multi-period problem were proposed by Taqa allah et al. [33]. They proposed two heuristics for constructing petrol station replenishment plans for the case in which there is only one depot, an unlimited homogeneous tank truck fleet, and no time windows. In 2002, Ben Abdelaziz et al. [3] presented a real-life routing problem in which a variable neighbourhood search heuristic was applied to solve a single period petroleum products delivery problem using a heterogeneous fleet of compartmented tank trucks. Malépart et al. [21] generalized this problem by allowing delivery to more than one station in the same trip. The authors proposed four heuristics for constructing replenishment plans over a horizon of several working days. Their heuristics were tested using some real-life problems obtained from a transport company in Eastern Canada.

Avella et al. [1] studied a petrol replenishment network involving one depot, a heterogeneous tank truck fleet and no time windows. They proposed a heuristic and an exact algorithm based on a route generation scheme and a branch-and-price algorithm. To test the performance of their approach, they used real-world data consisting of 25 customers and six tank trucks of three different types.

Ng et al. [23] studied two small petrol distribution networks in Hong Kong: the Hong Kong Island network and the network for the Kowloon Peninsula and the New Territories. They proposed a model for simultaneously assigning trips to trucks and stations. For this case, station' inventories were managed by the vendor who decided when to replenish each station and what quantities to deliver.

Cornillier et al. [11] tackled the petrol station replenishment problem (PSRP) with one depot. They developed an exact algorithm to solve the single period case with unlimited heterogeneous truck fleet but without time windows. Cornillier et al. [12] extended the PSRP to a multi-period setting and developed a multi-phase heuristic with look-back and look-ahead procedures. Basically, the heuristic first determines the stations to be serviced in each period. Then, the problem reduces to solving multiple PSRPs where the exact algorithm of Cornillier et al. [11] is utilized. Cornillier et al. [13] addressed the PSRP with time windows. They developed two heuristics based on the mixed integer linear programming formulation of the problem. In a different setting, Day et al. [15] implemented a three-phase heuristic for the cyclical inventory routing problem encountered at a carbon dioxide gas distributer in Indiana. Their heuristic determines regular routes for each of three available delivery vehicles over a 12-day delivery horizon while improving delivery labour cost, stockouts, delivery regularity and driver-customer familiarity.

Cornillier et al. [14] published a paper, which was different from the previous ones as it dealt with the multi-depot version of the problem. In addition to the proposed formulation, the authors developed a heuristic, which requires generating trips, not routes as in the previous papers, and trips are generated in a different way. In fact, they suggested a restricted set of promising feasible trips to solve the trip selection model. They used a two-phase procedure in which they first constructed an initial set of trips and then selected a subset of this set to obtain the required set.

Popović et al. [25] developed a variable neighbourhood search (VNS) heuristic model for solving a multi-period multi-product IRP (Inventory Routing Problem) in fuel delivery with multi-compartment homogenous vehicles and deterministic consumption that varies with each petrol station and each fuel type. The stochastic VNS heuristic is compared to a mixed integer linear programming (MILP) model and the deterministic "compartment transfer" (CT) heuristic. For the same problem, Vidović et al. [38] presented two solution approaches: the MIP model and the heuristics approach. The MIP model is formulated as the problem of petrol station assignment to individual routes with consideration of the daily inventory costs. The proposed heuristic includes a relaxed MIP model for obtaining the initial solution, ideas for transferring deliveries over one or more time periods earlier, assignment of petrol stations to a vehicle in the same route (represented through the utilities calculation) and a variable neighbourhood descent (VND) search.

Our aim is to improve the fuel distribution operations of Naftal company using OR techniques. The problem seems similar to that of Cornillier et al. [13] since both attack a petrol station replenishment problem with time windows using multi-compartment vehicles. However, there are some key differences. Firstly, Cornillier et al. [13] consider a case where trucks can visit up to four stations per route, which is justified by the fact that most trucks have from four to six compartments, while stations generally require two or three products, one of which frequently requires two compartments. Moreover, the time windows for servicing stations are very wide. In addition, their aim is to maximize the total profit equal to the sales revenue, minus the sum of routing costs and of regular and overtime costs, whereas, in our case, since restrictions on the accessibility of vehicles (see constraints 2 in the problem formulation) are imposed, a vehicle cannot visit some petrol stations, e.g., the petrol stations managed by the army cannot be delivered to by the private vehicles.

The remainder of this paper is organized as follows. Section 3 describes and formulates the problem. Section 4 describes the details of the proposed approach. The instances used and the results obtained are discussed in Sect. 5. Section 6 presents the conclusion and some suggestions for future work.

3 Problem description and formulation

The MCVRPTW is defined on a complete undirected network graph $G = (V, E)$ where $V = \{0, 1, 2, \ldots, n\}$ is a vertex set, and $E = \{(i,j) \in V \times V; 0 \le i,j \le n, i \ne j\}$ is an edge set. Vertex 0 represents the depot while the remaining vertices $N = \{1, 2, \ldots, n\}$ correspond to the customers. The depot stores p types of products. There are two types of vehicles. The number of vehicles for each type is limited. Let P, K_1 and K_2 represent the sets of the p types of products and the two types of vehicles (National company and Private companies vehicles), respectively. Each vehicle k has capacity C_k with the set of Q_1 or Q_2 compartments. Each compartment q is dedicated to product p and has a known capacity c_k^q. Furthermore, a penalty cost f_2 is incurred for each use of vehicle $k \in K_2$ in the routes. The travelling cost of each edge $(i, j) \in E$ is c_{ij}. For each customer $i \in N$, we have a positive demand d_{ip} and a time window $[a_i, b_i]$. Each demand has to be delivered in total. However, this demand may be adjusted according to the Agreement that was concluded between the company and the petrol station managers. Hence, the resulting new demand may be up to λ_{ip} % less than the ordered demand d_{ip}, i.e., $d'_{\text{ip}} = (1 - \lambda_{\text{ip}}) \times d_{ip}$, where $\lambda_{\text{ip}} \in [0, 0.10]$. This particular problem, called the MCVRPTW-AD (MCVRPTW with adjustment), occurs quite frequently when the demands of petrol stations are high in winter.

The time window at the depot $[a_0, b_0]$ corresponds to the feasible scheduling horizon for each vehicle route. Associated with each arc $(i, j) \in E$, t_{ij} represents the travel time along that arc. Note that the service time at customer i must start within the associated time window and the vehicle must stop for a t_i time. To take into account the restrictions imposed on the accessibility of vehicles (see constraint 2 in the problem formulation), we define $\{0, 1\}$ matrix $A = (a_k^i)$, which equals 1 if and only if customer i can be served by vehicle k and which equals 0 otherwise.

The MCVRPTW requires the following three types of variables:

- $x_{ijk} = \begin{cases} 1 & \text{if } i \text{ precedes } j \text{ in the route of vehicle } k. \\ 0 & \text{otherwise.} \end{cases}$
- $y_{ipk} = \begin{cases} 1 & \text{if customer } i \text{ receives product } p \text{ from vehicle } k. \\ 0 & \text{otherwise.} \end{cases}$
- s_{ik} specifies the arrival time at i when serviced by vehicle k. In case of vehicle k does not service i, s_{ik} has no meaning and consequently its value is considered irrelevant.

Given all the parameters and variables defined above, the MCVRPTW can be formulated as follows:

$$Minimize \sum_{k\in K_1}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ijk} + \sum_{k\in K_2}\sum_{i\in V}\sum_{j\in V}(c_{ij}+f_2)x_{ijk} \quad (1)$$

Subject to

$$y_{ipk} \le a_k^i, \quad i \in N, \quad k \in \{K_1 \cup K_2\}, \quad p \in P \quad (2)$$

$$\sum_{i\in V} x_{ihk} - \sum_{j\in V} x_{hjk} = 0, \quad h \in N, k \in \{K_1 \cup K_2\} \quad (3)$$

$$x_{ijk} + x_{jik} \le 1, \quad i,j \in N, i \ne j, k \in \{K_1 \cup K_2\} \quad (4)$$

$$\sum_{j\in N} x_{0jk} \le 1, \quad k \in \{K_1 \cup K_2\} \quad (5)$$

$$\sum_{i\in N} x_{i0k} \le 1, \quad k \in \{K_1 \cup K_2\} \quad (6)$$

$$\sum_{i\in N} d_{\text{ip}} y_{ipk} \le c_k^q, \ k \in \{K_1 \cup K_2\}, \quad q \in \{Q_1 \cup Q_2\}, \ p \in P \quad (7)$$

$$\sum_{k\in\{K_1\cup K_2\}} y_{ipk} = 1, \quad i \in N, \quad p \in P \quad (8)$$

$$y_{ipk} \le \sum_{j\in V} x_{jik}, \quad i \in N, \quad k \in \{K_1 \cup K_2\}, \quad p \in P \quad (9)$$

$$s_{ik} + t_i + t_{ij} - M(1 - x_{ijk}) \le s_{jk}, \quad i,j \in V, \quad k \in \{K_1 \cup K_2\} \quad (10)$$

$$a_i \le s_{ik} \le b_i, \quad i \in V, \quad k \in \{K_1 \cup K_2\} \quad (11)$$

$$x_{ijk} \in \{0, 1\}, \quad i,j \in V, \quad k \in \{K_1 \cup K_2\} \quad (12)$$

$$y_{ipk} \in \{0, 1\}, \quad i \in N, \quad k \in \{K_1 \cup K_2\}, \quad p \in P \quad (13)$$

$$s_{ik} \ge 0, \quad i \in N, \quad k \in \{K_1 \cup K_2\} \quad (14)$$

Objective (1) is to minimize the total cost, which consists of the travelling costs and penalty costs of private company vehicles used for service. Constraint (2) considers the accessibility restrictions, where some vehicles cannot serve some customers. Constraints (3)–(4) guarantee that one vehicle arrives at each customer, leaves it and does not return to the previous customer. Constraints (5)–(6) impose that the start and end of the route for each vehicle must be the depot. Constraint (7) ensures that the compartment capacity is not exceeded. In constraint (8), each product ordered by the customer is brought by one single vehicle. Constraint (9) sets y_{ikp} to zero for each product p if customer i is not visited by vehicle k. Constraint (10) represents the relationship between the starting time of the service to one customer and the departure time from its predecessor. In constraint (10), parameter M is an arbitrarily large value. Constraint (11) ensures that the service takes place at each customer with respect to the time window. Constraints (12)–(14) define the decision variables, which are all binary except for variable s_{ik}.

4 Solution approach

To solve the problem modelled above, we describe our approach from its general structure to its main components.

4.1 General structure

The detailed steps of this approach are completely described as follows:

- *Step 1. Initialization*
  1. Generate one initial feasible solution by combining the loading aspect with the modified Solomon heuristic [28] as described in Sect. 4.2.
  2. Initialize parameters.
- *Step 2. Local search*
  2-opt*, relocate and swap operators are applied to explore the search space. In fact, the search process is restricted to a set of elite neighbouring solutions, and the criterion used to select them is based on the Kolmogorov–Smirnov statistic. At each iteration, one of these three operators is randomly chosen and applied to the current solution S.
- *Step 3. Penalty component*
  In order to enlarge the search space by visiting infeasible solutions, the capacity and time constraints are relaxed, i.e., the capacity and time violations are penalized by two coefficients, and the penalized term is added to the objective function.
- *Step 4. Tabu list*
  The tabu list is implemented as an upper triangular matrix where each element contains a set of attributes able to characterize the solution and records the iteration in which an arc is moved from one route to another. The size of the tabu list is updated according to the quality of the solutions obtained during the recent moves.
- *Step 5. Diversification and intensification strategies*
  Four mechanisms are used for diversifying and intensifying the search (see Sect. 4.7 for more details).
- *Step 6. Termination*
  The process can be stopped if the termination conditions are completed; otherwise go to step 2 and continue the search.

4.2 Route construction heuristic

Usually, the methods used to generate an initial solution are simple and fast to compute because it is assumed that the task of producing a good solution is mainly handled by the tabu search algorithm [28]. In our work, the search process for the initial solution includes two main phases. In the first phase, called loading, an initial loading solution is obtained. This solution is then used to generate the initial routes in the routing phase. The details of these two phases are explained as follows:

1. In the loading phase, our aim is to determine the subsets of orders that will be loaded into the same vehicle. More precisely, one must determine which of the products is assigned to a certain compartment. The compartments of each vehicle are loaded so that none of the routing constraints is considered. This can be done by solving the following compartment-loading problem (CLP):

$$Minimize \sum_{k\in\{K_1\cup K_2\}} \sum_{q\in\{Q_1\cup Q_2\}} \sum_{p\in P} z_{pqk} \tag{15}$$

subject to

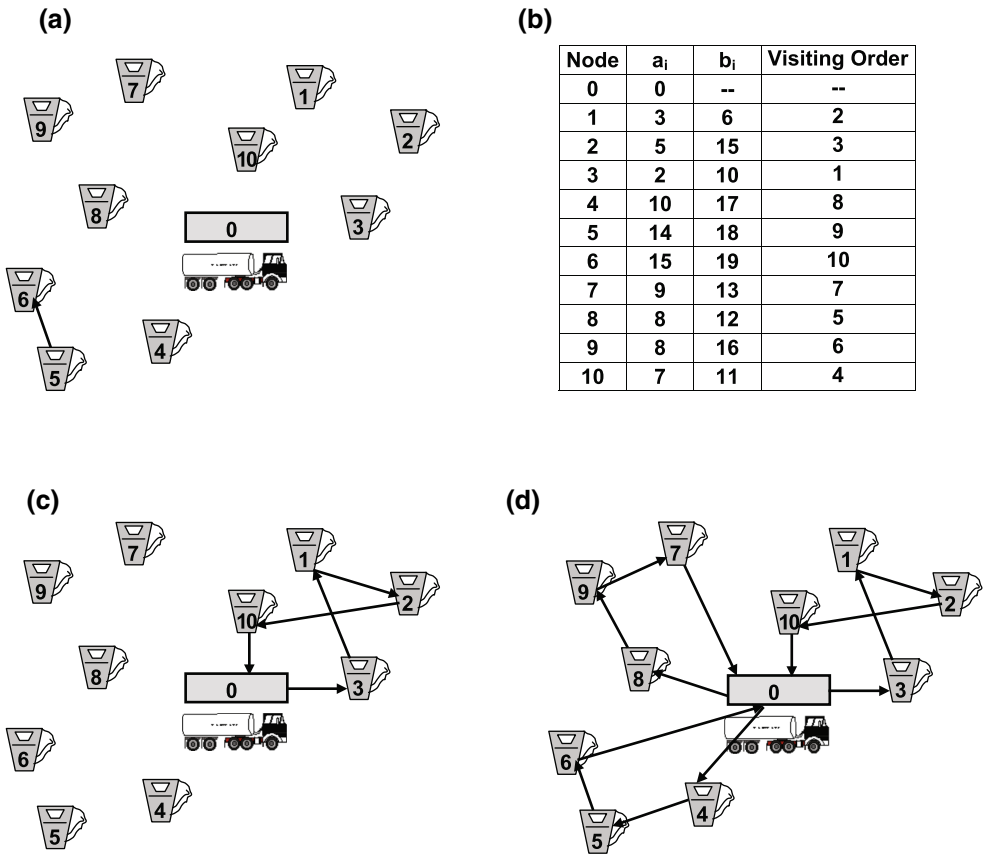
$$d_{ip} \le \sum_{q\in\{Q_1\cup Q_2\}} c_k^q z_{pqk}, \quad k \in \{K_1 \cup K_2\}, \quad i \in N, \quad p \in P \tag{16}$$

$$\sum_{p\in P} z_{pqk} \le 1, \quad k \in \{K_1 \cup K_2\}, \quad q \in \{Q_1 \cup Q_2\} \tag{17}$$

   Binary variable z_{pqk} indicates whether product p is assigned to compartment q in vehicle k. The objective function in (15) expresses the fact that we wish to minimize the number of loaded compartments. Constraint (16) states that the quantities to load for one product must not exceed the sum of the capacity of compartments assigned to that product. Constraint (17) imposes that each compartment is dedicated at most to one product per route.
2. In the above initial loading solution, each vehicle has a list of customers to visit. For each list, we apply the modified nearest neighbour heuristic proposed by Solomon [28] to generate our initial routes. As graphically described in Fig. 1, our heuristic constructs the routes by first visiting the customer closest to the depot, where only temporal closeness is taken into account. At each iteration, a vehicle with its subset of customers is selected. The route starts with the customer who has the earliest time a_i. The next customer to be visited in the route will be the one that is (1) not yet visited and (2) closest to the last customer in the current route. This process is repeated until there is no customer to visit in the current route. When this happens, the whole process is repeated until all the customers are visited.

4.3 Neighbourhood structure

In the tabu search algorithm, it will take a long time to compute the values of all moves that allow one to pass



Node	a_i	b_i	Visiting Order
0	0	--	--
1	3	6	2
2	5	15	3
3	2	10	1
4	10	17	8
5	14	18	9
6	15	19	10
7	9	13	7
8	8	12	5
9	8	16	6
10	7	11	4

Fig. 1 Route construction heuristic: **a** Spatial location of the depot and customers. **b** Customers' time windows and visiting order. **c** Customers visited by one vehicle. **d** Initial solution

from one solution to a neighbouring one. One of the reasons for these large computing-time requirements is that the mechanisms of generating candidate solutions normally need to perform several thousand iterations to obtain high-quality solutions [34]. Therefore, the strategy of selecting the nearest neighbours was adopted to improve the convergence speed in this paper. Within the general structure of the neighbourhood, this strategy may be seen as an implementation of a candidate list. In fact, the search is restricted to a set of elite neighbouring solutions, and the criterion used to select them is fixed in advance. The size of the candidate list will be determined in the following sections by applying the Kolmogorov–Smirnov statistic. The introduction of this statistic is an advance over existing work in which most of the relevant algorithms tend to use classical moves. To improve the clarity of exposition, in Sect. 4.4, we describe in detail the Kolmogorov–Smirnov statistic by applying it to a given instance of Solomon. As is shown in Fig. 4, with 100 customers, the size of the candidate list is restricted to a value of 21. Our approach employs three move types; each explores a restricted search space by embedding these lists in the search process. At each iteration, the algorithm randomly chooses one operator from these three operators and applies it to the current solution S. The details of these operators are listed and described below.

a. *2-opt**: Two customers, $i \in R_k$ and $j \in R_{k'}$, are chosen. Then, the edges connecting i to $i+1$ and j to $j+1$ are removed. Two new edges are added adjoining i with $j+1$ and j with $i+1$. See Fig. 2a.
b. *Relocate*: Two customers, $i \in R_k$ and $j \in R_{k'}$, are selected and i is removed from its original route R_k and inserted following j in the second route $R_{k'}$. The relocate operator may reduce the number of vehicle routes. See Fig. 2b.
c. *Swap*: Two customers, $i \in R_k$ and $j \in R_{k'}$, are chosen and exchanged between two routes. See Fig. 2c.

4.4 Neighbourhood size

The tabu search can be very time consuming due to the large size to the neighbourhood N(S) and also to the cost functions that must be constantly reevaluated. Thus, we propose a neighbour reduction strategy designed to reduce the computing time. Within the general settings of tabu

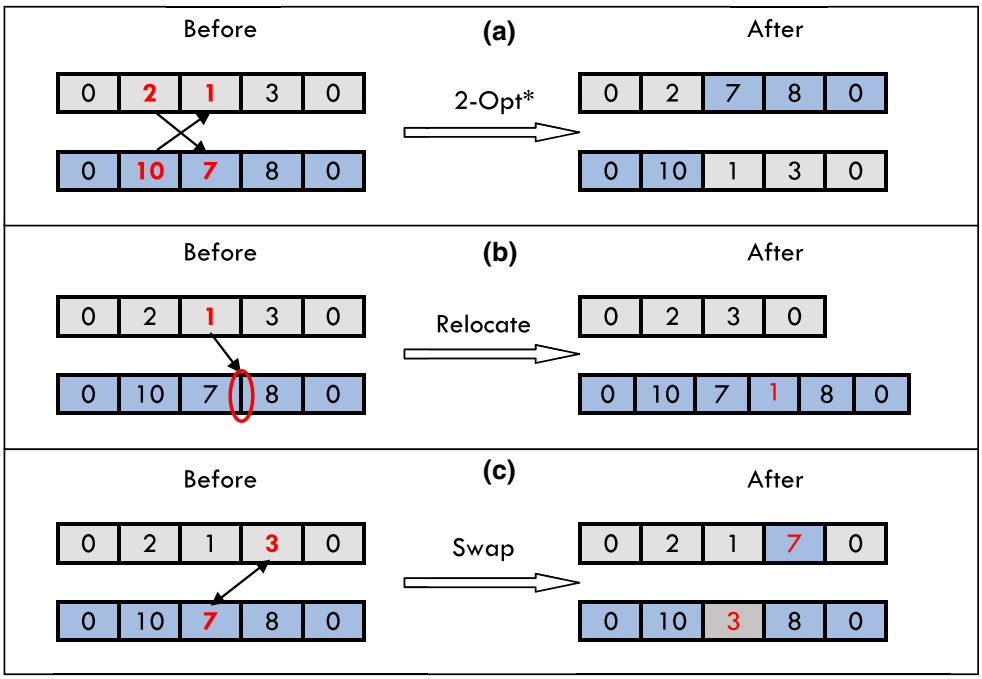


Fig. 2 Neighbourhood structure

search, this strategy may be seen as an implementation of candidate list [19]. In fact, the search is restricted to a set of elite neighbouring solutions, and the criterion used to select them is fixed in advance. Similar ideas have been used by various authors to speed up local-search algorithms [35]. The most widely used strategy, called Random, involves randomly selecting some neighbours from N(S) for evaluation, thus allowing only a predefined proportion of the total neighbourhood to be considered. By forcing a decrease in the number of neighbours, the strategy is able to decrease the time needed to find a solution. However, because the strategy does not necessarily select the best neighbour, it could lead to a different, and maybe worse, solution. Another frequently used strategy, called Distance, relies on taking the problem configuration into account [26]. The idea is simple: a given neighbour is evaluated only if the customer under consideration is within a fixed distance from the nearest customer on the route into which it will be inserted. For example, in [26], the fixed distance is predefined as being equal to a fraction of the longest distance on the geographical map. Apart from the mentioned strategies, the candidate list is also restricted to a fraction of the total number of customers [8].

By analysing these strategies, some drawbacks can be underlined. In fact, their way of fixing the number of candidates does not take into account the instance's density and consequently no guarantee is provided regarding the compromise between solution quality and the time to find it. This means that if the customer under consideration is located in a remote location, the algorithm can evaluate some unpromising moves; or if it is in a cluster, it might still result in too many moves to be evaluated. In order to overcome these drawbacks, our paper attends to additional aspects which, according to the best of our knowledge, have not been considered in the literature before. These aspects include (1) the analysis of the instance configuration, (2) the introduction of the Kolmogorov–Smirnov distance criterion, and (3) the implementation of candidate list for each customer. The second aspect defines a new feature with which we suggest a heuristic (Algorithm 1) for the neighbourhood size.

The purpose of introducing the Kolmogorov–Smirnov statistic (KSS) is to find the preferable customers to visit (when moving from one solution to a neighbouring one), taking into to account their geographical positions in the instance. The idea is simple: (1) select the model that minimizes the distance criterion, (2) deduce its parameters and (3) decide whether two customers are neighbours or not. The customers are considered neighbours of a given customer i, if and only if the distance between them is less or equal to a given parameter A_i. This parameter is calculated by using the pth quantile formula, which depends on the probability distribution of its distance sample L_i. The use of such a probability aspect in computing A_i is justified by the aim of checking whether one of the known distributions can be suited to each distance sample. In fact, the KSS fits the distance sample by selecting a list of candidate distributions, estimating their parameters and giving their ranking. Once the best statistical model is identified, the parameter A_i is defined as follows: $F(A_i) = \Pr(L_i \leq A_i) = p \Rightarrow A_i = F^{-1}(p)$.

The whole procedure of fit is briefly explained in Algorithm 1.

Algorithm 1 Procedure of fit.

Step 1.
for each customer $i = 1, ..., n$ **do**
(a) Calculate $l_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, j \in N$
(b) Build the set $L_i = \{l_{ij}, j \in N\}$
end for
Step 2.
Over the candidate distributions do:
(a) Fit the distributions to the sample (L_i).
(b) Calculate the Kolmogorov-Smirnov statistics, give their ranking and parameters.
(c) Choose the best distribution that minimizes the distance criterion.
(d) Use the parameters of the best distribution to determine $A_i = F^{-1}(p)$.
Step 3.
Repeat steps 2 for each sample L_i.

To improve the clarity of exposition, we give an illustrative application in which we show how to apply the Kolmogorov–Smirnov statistic on a given instance. Let $R101$, $i = 26$ and L_{26} represent the instance of Solomon (Fig. 4), the selected customer and its distance sample, respectively. With this sample design, we test the following set of distributions: Normal, Exponential, Weibull, GEV, Gamma, Fréchet and Pareto. Table 1 summarizes the goodness of fit for this example and shows that the GEV model is the best. This result is confirmed when the GEV quantile-quantile plot of the L_{26} variable is displayed as in Fig. 3. As can be seen in this figure, our distribution is very closely related to the empirical cumulative distribution function. Hence, the L_{26} variable seems to follow a GEV distribution which is defined as follows:

$$\begin{cases} P(L_i \leq A_i) = F(A_i, \mu_i, \sigma_i, \xi_i) \\ P(L_i \leq A_i) = \exp\left\{ -\left[1 + \xi_i \left(\dfrac{A_i - \mu_i}{\sigma_i} \right) \right]^{-1/\xi_i} \right\} \end{cases} \quad (18)$$

where F is the Cumulative Distribution Function (CDF), $\mu_i \in \mathbb{R}$ is the location parameter, $\sigma_i > 0$ is the scale parameter and $\xi_i \in \mathbb{R}^*$ is the shape parameter. By inverting (18), we compute the parameter GEV A_i as follows:

Table 1 Goodness of fit-synthesis

Distribution	Kolmogorov–Smirnov		Corresponding parameters
	Statistic	Rank	
GEV	0.0372	1	$\mu = 22.697,\ \sigma = 10.923,\ \xi = -0.1901$
Weibull	0.0530	2	$\alpha = 2.6739, \beta = 30.999$
Gamma	0.0567	3	$\alpha = 5.4814, \beta = 5.0205$
Normal	0.0602	4	$\mu = 27.244, \sigma = 11.471$
Fréchet	0.1327	5	$\alpha = 1.9407, \beta = 19.648$
Exponential	0.2949	6	$\lambda = 0.0367$
Pareto II	0.3141	7	$\alpha = 197.26, \beta = 5048.2$
Pareto I	No fit	8	No fit

$$F(A_i, \mu_i, \sigma_i, \xi_i) = p \Rightarrow A_i = \left[\left(-\frac{1}{\ln p} \right)^{\xi_i} - 1 \right] \frac{\sigma_i}{\xi_i} + \mu_i. \quad (19)$$

With $(\mu_{26}, \sigma_{26}, \xi_{26}) = (22.697, 10.923, -0.1901)$ and $p = 25$ % for example, we get $A_{26} = 19.016$. Hence, the set of nearest neighbours of the 26th customer will be noted by $V(26)$ and Fig. 4 offers its representation. $V(26) = \{j \in N | l(26, j) \leq 19.016\}$. This set, considered as the restricted neighbourhood of the current solution, will be used by the three operators to evaluate the neighbourhood $N(26)$.

4.5 Penalty component

To develop the approach for the MCVRPTW, possible violations of both capacity and time window constraints need to be addressed. Let S be an infeasible solution that violates capacity constraints and/or time window constraints. The penalized cost function $F'(S)$ of solution S is defined in Eq. (20). It consists of the total travel distance $F(S)$ and the penalty terms $C(S)$ and $T(S)$ for the violations of the capacity and time window constraints multiplied by the penalty coefficients α and β respectively. Initially set equal to 1, these coefficients are periodically divided by $1 + \rho$ ($\rho \in]0, 1[$) after each block of ϕ feasible solutions and multiplied by $1 + \rho$ after each block of ϕ infeasible solutions. This way of proceeding produces a mix of feasible and infeasible solutions, which acts as a diversification strategy.

$$F'(S) = F(S) + \alpha C(S) + \beta T(S). \quad (20)$$

$C(S)$ is straightforwardly defined in Eq. (21) as the sum of the total demand excess in all routes [35].

$$C(S) = \sum_{k \in \{K_1 \cup K_2\}} \max \left\{ \sum_{p \in P} \sum_{i \in N} d_{ip} y_{ipk} - C_k, 0 \right\} \quad (21)$$

As for the T(S) penalty term, variants of the time window penalty for the VRP with soft time windows are

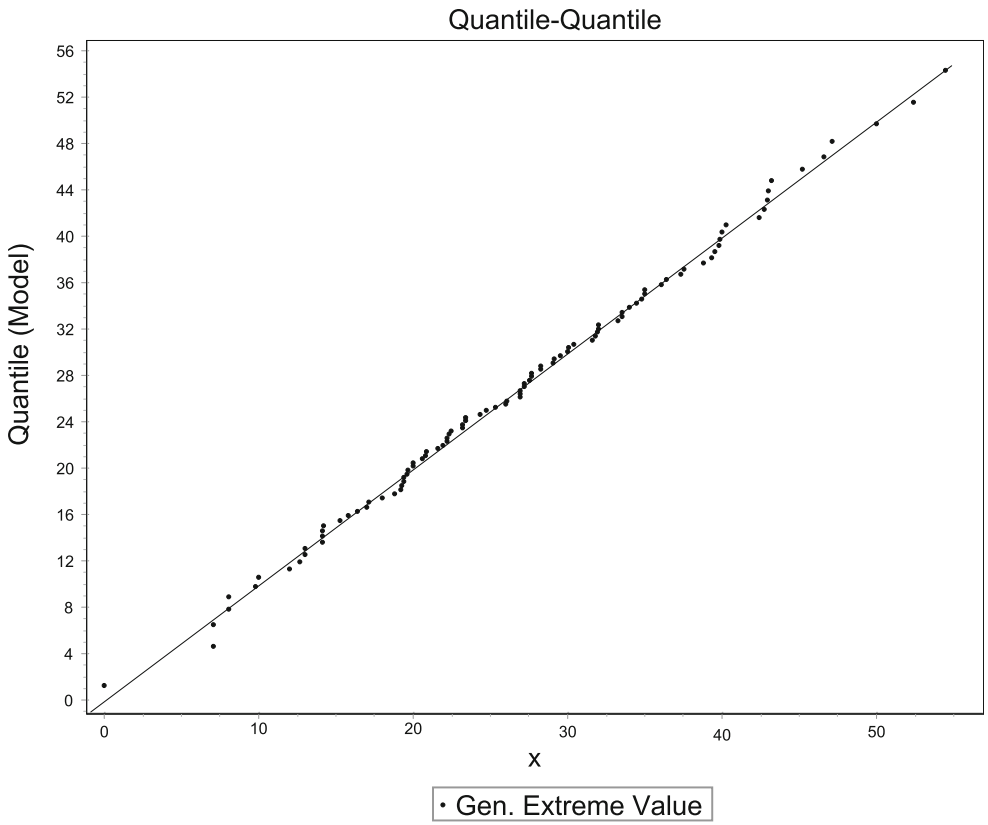
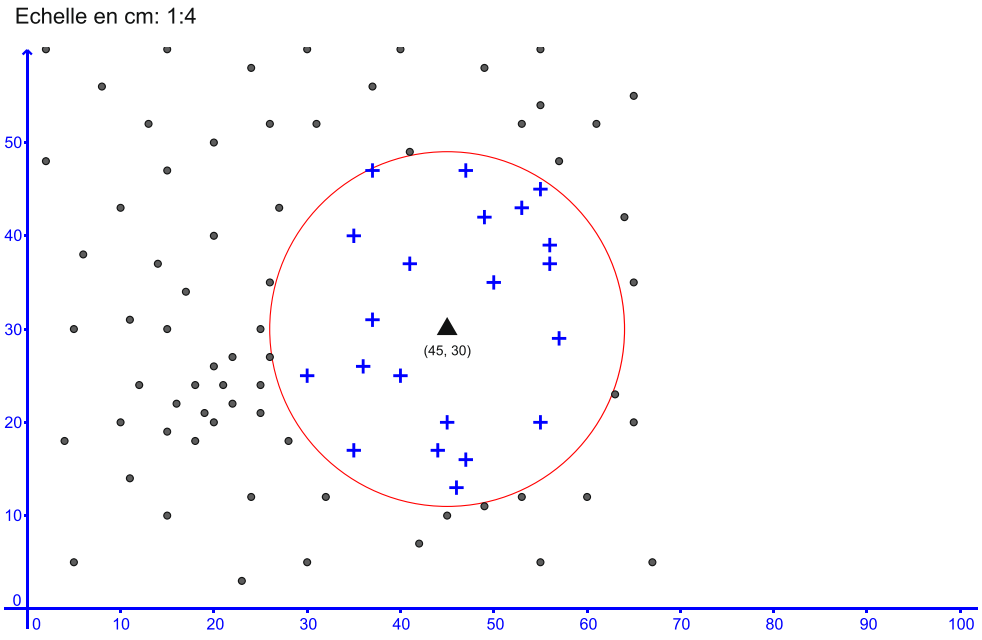


Fig. 3 GEV Quantile–Quantile plot

Fig. 4 An example of candidate list



employed in Berger and Barkaoui [4]. We suggest the definition of Nagata et al. [22] for the time window penalty structure defined in Eq. (23) whose change can be computed in O(1) time for most traditional neighbourhood operators. Moreover, this penalty measures the amount of the time window violation more appropriately as described below.

Given a solution S, the extended earliest departure time at a depot $\widetilde{s}_{0k}$, the extended earliest start time of service at a customer i, $\widetilde{s}_{ik}$ are defined recursively in Eq. (22) and the

suggested time window penalty of the solution, denoted as T(S), is defined in Eq. (23).

$$\begin{cases} \widetilde{s'}_{0k} = \widetilde{s}_{0k} = a_0 \\ \widetilde{s'}_{ik} = \widetilde{s}_{i-1,k} + t_{ik} + t_{i-1,i} \\ \widetilde{s}_{ik} = \max(\widetilde{s'}_{ik}, a_i) \quad \text{if} \quad \widetilde{s'}_{ik} \le b_i \\ \widetilde{s}_{ik} = b_i \quad \text{if} \quad \widetilde{s'}_{ik} > b_i \end{cases} \tag{22}$$

$$T(S) = \sum_{k \in K} \sum_{i \in V} \max(\widetilde{s'}_{ik} - b_i, 0\} \tag{23}$$

Note that $\widetilde{s}_{ik}$ is equal to s_{ik} if the route of vehicle k is feasible with respect to the time window constraint. $\widetilde{s'}_{ik}$ refers to the extended earliest arrival time of vehicle k at customer i, and the time window constraint is violated at customer i if $b_i < \widetilde{s'}_{ik}$. In this case, we assume that the vehicle can travel back in time to b_i to start the service of customer i without delay, but at the expense of paying a penalty $(\widetilde{s'}_{ik} - b_i)$. Therefore, in this case, $\widetilde{s}_{ik}$ is set to b_i. The total time window violation in solution S, $T(S)$, is defined by the sum of the penalties that the vehicles must pay in all the routes to service all customers and to arrive at the depot without delay.

4.6 Tabu list

The tabu list is implemented as an upper triangular matrix L of $K \times K$ dimensions where each element $L(k, k'), (k, k' = 1, ..., K, k < k')$ is associated with the pair of routes R_k and $R_{k'}$. Each element $L(k, k')$ contains a set of attributes able to characterize the solution and also records the iteration in which the arc (i, j) has been moved from route R_k to route $R_{k'}$. The element $L(k, k')$ has the structure: $(R_k, R_{k'}, i, position\, i, j, position\, j, F(S), t)$.

where R_k and $R_{k'}$ are the two routes under operation, i is a customer from R_k and *position i* is the service order of i in R_k. The case is likewise for j and *position j*. $F(S)$ is the solution value, and t is the iteration in which the arc (i, j) has been moved from route R_k to route $R_{k'}$. An arc moved at iteration t cannot be reinserted in the solution before iteration $t + \theta$.

This notion provides a guideline to avoid making similar moves in the near future. Such representation does not uniquely describe a move, because a full description is very complicated and its use increases the computation tremendously. Therefore, when an exchange between routes R_k and $R_{k'}$ is accepted, we just update the information corresponding to line k and column k'. Thus, we avoid calculating information above the other pairs of routes, which do not contain either R_k or $R_{k'}$. The size of tabu list θ takes its values in $[\theta_{\min}, \theta_{\max}]$ starting from θ_{init}. Parameter θ is updated according to the quality of the solutions obtained during the recent moves. After each improvement of the current objective function, θ is updated as $\theta = \max(\theta - 1, \theta_{\min})$ with the aim of intensifying the search around this solution. Otherwise, after ϕ_{LT} consecutive moves deteriorating the value of the visited solutions, the size of the tabu list is updated as $\theta = \min(\theta + 1, \theta_{\max})$.

4.7 Diversification and intensification

There are two diversification strategies for the proposed algorithm to guide the search into less explored regions. (1) A neighbourhood is randomly selected from the moves described in Sect. 4.3. (2) The idea of passing through infeasible regions is introduced and defined in Sect. 4.5.

The intensification is mainly achieved by using the following two strategies. (1) The search is accentuated around the best-known solution by increasing or decreasing the size of the tabu list as explained in Sect. 4.6. (2) The full search proceeds starting from the most promising solution. In fact, the tabu search restarts the exploration of the solution space from the best feasible solution evaluated, but not visited $\overline{S}$. To update $\overline{S}$ at each iteration, the algorithm generates two solutions, S' and S'', from the neighbourhood $N(S)$ of the incumbent solution S. S' represents the best non-tabu solution in $N(S)$, and it is used to continue the search process, while S'' represents the best non-tabu feasible solution obtained in $N(S) \backslash S'$. Note that S' can be infeasible, since the solutions visited may violate capacity or time constraints. After $\gamma_{\max}$ iterations without improving the best feasible solution found so far or after $\nu_{\max}$ iterations since the last restart, the search process "jumps" to $\overline{S}$ and restarts with an empty tabu list. The maximal number of restarts is fixed as $\eta_{\max}$, but this process can be stopped if after $\overline{\eta_{\max}}$ restarts the best solution is not improved.

4.8 Algorithm overview

Algorithm 2 gives the skeleton of the proposed tabu search. Before describing its general structure, we first give some notations as follows:

- η: Number of restarts.
- $\overline{\eta}$: Number of restarts without improvement.
- γ: Number of iterations without improvement.
- γ_S: Number of iterations without improvement the incumbent solution.
- ν: Total number of iterations.
- χ: Number of consecutive feasible solutions.
- $\overline{\chi}$: Number of consecutive infeasible solutions. $\overline{\chi} = \overline{\chi_c} + \overline{\chi_t}$.
- $\overline{\chi_c}$: Number of consecutive infeasible solutions that violate capacity constraint.
- $\overline{\chi_t}$: Number of consecutive infeasible solutions that violate time window constraint.

Algorithm 2 Procedure Search

(1) Generate one initial solution S_0 by combining the loading aspect with the modified nearest neighbor heuristic as described in Section 4.2.
(2) Set $\overline{S} := S_0, S^* := S_0$
(3) initialize the parameters: $\theta = \theta_{init}, \eta = \overline{\eta} = \gamma = \nu = \chi = \overline{\chi} = \gamma_S = 0$.
repeat
  $\eta = \eta + 1$
  while $\nu \leq \nu_{max}$ and $\gamma \leq \gamma_{max}$ **do**
    Randomly choose an operator from the set of three operators and apply it to the current solution S. Two solutions $S^{'}$ and $S^{''}$ are generated from $N(S)$:
    $S^{'}$: the best non-tabu solution in $N(S)$.
    $S^{''}$: the best feasible non-tabu solution in $N(S) \setminus S^{'}$.
    Set $\overline{S} = argmin(F(S^{'}), F(S^{''}))$
    if $F(S^{'}) < F(S)$ **then**
      Update the size of the tabu list. Set $\theta := max(\theta - 1, \theta_{min})$
      Set $\gamma_S := 0$
    else
      Update $\gamma_S := \gamma_S + 1$
    end if
    if $\gamma_S = \phi_{LT}$ **then**
      Update the size of the tabu list. Set $\theta := min(\theta + 1, \theta_{max})$
      Set $\gamma_S := 0$
    end if
    if $F(S^{'}) < F(S^*)$ and $S^{'}$ is feasible **then**
      Set $S^* := S^{'}$, $\gamma := 0$, $\chi := \chi + 1$.
    else
      $\gamma := \gamma + 1$
    end if
    $S := S^{'}$
    Update the tabu list
    if S is infeasible **then**
      $\overline{\chi} := \overline{\chi} + 1$. Update $\overline{\chi_c}, \overline{\chi_t}$.
    else
      $\overline{\chi} := 0$. Update $\overline{\chi_c}, \overline{\chi_t}$.
    end if
    check the variables corresponding to the number of consecutive feasible and non feasible solutions and update the penalization parameters.
    if $\chi = \phi$ **then**
      Set $\alpha := \alpha/(1 + \rho)$, $\beta := \beta/(1 + \rho)$.
    end if
    if $\overline{\chi} = \phi$ **then**
      Set $\rho_c := \overline{\chi_c}/\phi, \rho_t := \overline{\chi_t}/\phi$.
      Update $\alpha := \alpha * (1 + \rho_c)$, $\beta := \beta * (1 + \rho_t)$.
    end if
  end while
  if S^* is updated **then**
    $\overline{\eta} := 0$
  else
    $\overline{\eta} := \overline{\eta} + 1$
  end if
  $S = \overline{S}$
until $(\eta = \eta_{max})$ or $(\overline{\eta} = \overline{\eta_{max}})$
Return S^*.

5 Computational results

5.1 Overview

Because the problem under study is a new problem in the literature, there is still a lack of benchmark instances to test the behaviour of the algorithm. As a result and with the aim of avoiding indirect or limited comparisons, we performed our computational experiments on the well-known data sets given by Solomon [28]. These instances (http://www.neo.lcc.uma.es/vrp/vrp-instances/), developed for the classical VRPTW, are transformed to fit our problem. There are, in total, 56 different instances, which can be classified into six categories. In sets R1 and R2, the customer locations are randomly generated. Problem sets C1 and C2 have clustered distributions of customers. Sets RC1 and RC2 are semi-clustered with a mix of randomly distributed and

clustered customers. Sets R1, C1 and RC1 have a shorter route horizon compared with those of sets R2, C2 and RC2, which have longer scheduling horizons. In order to accommodate these instances to the MCVRPTW with two products and two compartments, we adopted the idea proposed by El Fallahi et al. [16] to derive the MCVRP instances from the VRP data sets. More precisely, we split each customer demand into two parts. The demands of customer i for the two types of products are $d_{i1} = [d_i/\omega]$ and $d_{i2} = d_i - d_{i1}$ respectively where ω is a random integer in [2, 3], $[x]$ stands for the integer part of x and d_i indicates the demand of customer i in Solomon instances. The capacity of compartments is set as $c_{q1} = (C \times \overline{d_1})/(\overline{d_1} + \overline{d_2}); c_{q2} = (C \times \overline{d_2})/(\overline{d_1} + \overline{d_2})$ where $\overline{d_1}$, $\overline{d_2}$ and C are the average demand for the first product, the average demand for the second product and the vehicle capacity in the original VRPTW, respectively.

Although our implementation handles more compartments, we took the case of two compartments for testing because a) the problem is already difficult enough with two compartments, and b) the demands obtained by splitting the original demands into three or more demands would be too small.

To further assess the performance of the algorithm, we solve a real case where the company managers provided us with daily data for a period of 15 days.

All instances can be downloaded from the VRP-REP website (http://www.vrp-rep.org/).

The algorithm proposed in our solution procedure is coded in C++, and the mathematical model is solved by using IBM ILOG CPLEX 12.6 for a 7200s CPU. All experiments are conducted on a laptop computer with an Intel Core i7 2.9 GHz processor with 8 GB RAM and operating with Windows® 7 Professional edition.

5.2 Parameters

After some preliminary experiments, the parameter configurations for the tabu search algorithm have been set to the values reported in Table 2. α, β and ρ are three penalty factors. ϕ and ϕ_{LT} have been set to the same value and, respectively, count the number of consecutive moves updating or deteriorating the value of the visited solutions. The size of the tabu list is updated in the interval $[\theta_{\min}, \theta_{max}]$ starting from θ_{init}. The maximum number of moves without improving the best feasible solution in each restart is $\gamma_{\max}$. $\nu_{\max}$ is the number of iterations of each restart. $\eta_{\max}$ fixes the maximal number of restarts while $\overline{\eta_{max}}$ is the maximal number of restarts without improvement. λ_{ip} is the threshold used as an adjustment parameter in Eq. (24). p is the pth GEV quantile used to find the neighbourhood size.

Table 2 Parameters used in tabu search algorithm

Parameter	Value
α	1
β	1
ρ	0.3
$\phi = \phi_{LT}$	15
θ_{init}	10
$\theta_{\min}$	5
$\theta_{\max}$	15
$\gamma_{\max}$	700
$\nu_{\max}$	2500
$\eta_{\max}$	100
$\overline{\eta_{\max}}$	5
λ_{ip}	0.04
p	0.25

5.3 Results

5.3.1 Results for the MCVRPTW instances

We started by comparing the MCVRPTW with what we call the MCVRPTW-AD (MCVRPTW with adjustment) in which the quantities loaded in the compartments can be adjusted up to a given threshold (see Sect. 3). To handle this MCVRPTW-AD, we simply modified the tabu search algorithm to get two versions in which constraint (7) is replaced by constraint (24):

$$\sum_{i \in N} (d_{\text{ip}} - \lambda_{\text{ip}}) y_{ipk} \leq c_k^q, \quad k \in \{K_1 \cup K_2\}, \quad q \in \{Q_1 \cup Q_2\}, \quad p \in P \tag{24}$$

Table 3 provides the solutions obtained by the tabu search algorithm for each instance. In Table 3, we report the total distance (TD), the number of vehicles used for the service (NV), the CPU times in seconds (CPU) and the final gain in percentage (*Gain* %) when the adjustment option is permitted. Solutions in this table indicate that, in all problem instances, the adjustment option proves to be favourable, both in terms of the total distance (7 %) and the number of vehicles used (10 %). Another interesting thing we have found is that the tabu search converges much faster for the MCVRPTW than for the MCVRPTW-AD. This may be explained by the fact that, in the MCVRPTW-AD, we have the possibility of moving customers among the routes without violating any capacity constraint. Conversely, when excluding the adjustment option, the set of feasible solutions for the MCVRPTW becomes very restricted, resulting in a rapid search.

We then tried to solve the formulation (1)–(14) directly within CPLEX 12.6. CPLEX failed to solve any of the MCVRPTW instances to optimality and produced an out of

Table 3 Best solutions on the MCVRPTW instances

Data	MCVRPTW			MCVRPTW-AD			Gain (%)	
	NV	TD	CPU (s)	NV	TD	CPU (s)	NV	TD
C1	13	1131.39	432	12	1030.74	541	9	10
C2	7	1056.80	359	6	986.31	491	13	7
R1	17	1496.71	543	16	1422.73	731	7	5
R2	6	1163.03	459	5	1102.62	601	11	5
RC1	16	1728.05	364	15	1588.31	541	9	9
RC2	7	1356.80	295	7	1302.41	441	9	4
Avg.	11	1322.67	825	10	1240.97	1069	10	7

memory error after about 16,000s of computation time. Because of this, we compared the solutions obtained for the small-sized instances by our algorithm to those obtained by CPLEX 12.6. We conducted a set of experiments by randomly selecting 15 customers from each MCVRPTW instance. These instances are denoted as in the following example. C101-15 corresponds to the instance of class "C1" where only 15 customers are considered. In our solution, each of the generated instances is resolved in the same way, as for the instances with 100 customers considered above. For CPLEX, a maximum CPU of 7200s is imposed on the solution time. Table 4 shows the solutions obtained by our algorithm and the optimal solutions obtained by CPLEX. For completeness, the final optimality gap in percentage (Gap_f (%)) is also provided.

Results in this table indicate that our algorithm provides the optimal solutions to C102-15, R101-15 and RC102-15 instances with a substantially lower computation time. Furthermore, CPLEX cannot provide a feasible solution (within the time limit) to six problem instances (C104-15, R103-15, R104-15, R107-15, R108-15 and RC107-15). These results confirm that the solutions obtained by our algorithm are better than those obtained by CPLEX. Moreover, the average computation time required by our solution procedure to solve these instances is much less than that required by CPLEX.

5.3.2 Results for the VRPTW instances

For a more meaningful comparison of results, we interpreted the classical VRPTW instances as MCVRPTW instances with one product and one compartment. To this end, we compared our results with nine other metaheuristic approaches proposed by Taillard et al. [29] (TBGGP), Chiang and Russell [9] (CR), Gambardella et al. [18] (GTA), Tan et al. [30, 31] (TLO), Cordeau et al. [10] (CLM), Braysy and Gendreau [5] (BG), Lau et al. [20] (LST), Tan et al. [32] (TCL) and Vidal et al. [37] (VCGP). The comparison of the results of each approach is shown in Table 5. The first row gives the name of the authors of the study. Rows C1, C2, R1, R2, RC1 and RC2 present the average number of vehicles (NV) and average total distance (TD) with respect to the six groups of problem instances, respectively. The performance of our algorithm

Table 4 Comparison between our solutions and optimal solutions obtained by CPLEX

Data	Our solutions			CPLEX solutions			Gap_f(%)
	NV	TD	CPU	NV	TD	CPU	
C101-15	3	222.61	13	3	220.92	7200	70.34
C102-15	**2**	**207.84**	**24**	**2**	**207.84**	**2593**	
C103-15	3	227.68	32	3	233.68	7200	84.04
C104-15	3	220.62	48		***NO SOL***		
C105-15	3	229.88	15	3	229.88	7200	51.32
C106-15	3	239.02	19	3	239.02	7200	45.35
C107-15	2	198.08	23	2	194.95	7200	86.79
C108-15	3	270.96	22	3	273.78	7200	84.16
C109-15	3	233.44	31	3	243.21	7200	87.33
R101-15	**4**	**359.66**	**39**	**4**	**359.66**	**6025**	
R102-15	4	338.06	47	4	353.11	7200	78.12
R103-15	3	278.88	90		***NO SOL***		
R104-15	3	280.54	43		***NO SOL***		
R105-15	3	298.92	32	4	295.77	7200	72.86
R106-15	3	305.41	37	3	322.19	7200	78.37
R107-15	3	297.50	57		***NO SOL***		
R108-15	3	284.22	45		***NO SOL***		
R109-15	3	299.71	36	4	318.44	7200	69.95
R110-15	3	271.52	41	3	311.28	7200	88.15
R111-15	3	300.46	42	4	329.57	7200	75.43
R112-15	3	270.25	42	3	428.92	7200	82.03
RC101-15	4	370.26	26	4	370.26	7200	80.10
RC102-15	**3**	**367.34**	**26**	**3**	**367.34**	**791**	
RC103-15	3	319.78	34	3	324.55	7200	89.76
RC104-15	3	326.87	53	3	398.20	7200	79.92
RC105-15	3	393.84	41	3	400.41	7200	88.17
RC106-15	3	319.56	33	3	320.80	7200	90.12
RC107-15	3	339.49	26		***NO SOL***		
RC108-15	3	322.23	38	3	323.92	7200	89.77
Avg.	3	289	36	3	307	6670	78.60

Bold indicates optimal solutions

Table 5 Comparison among different heuristics

Data set		TBGGP [29]	CR [9]	GTA [18]	TLO [30, 31]	CLM [10]
C1	NV	10	10	10	10	10
	TD	828.45	828.38	828.38	828.74	828.38
C2	NV	3	3	3	3	3
	TD	590.30	591.42	589.86	590.69	589.86
R1	NV	12.25	12.17	12	12.92	12.08
	TD	1216.70	1204.19	1217.73	1187.35	1210.14
R2	NV	3	2.73	2.73	3.51	2.73
	TD	995.38	986.32	967.75	960.83	969.58
RC1	NV	11.88	11.88	11.63	12.74	11.50
	TD	1367.51	1397.44	1382.42	1355.37	1389.78
RC2	NV	3.38	3.25	3.25	4.25	3.25
	TD	1165.62	1229.54	1129.19	1068.26	1134.51

		BG [5]	LST [20]	TCL [32]	VCGP [37]	Our solution
C1	NV	10	10	10	10	10
	TD	828.38	832.13	828.38	828.38	828.38
C2	NV	3	3	3	3	3
	TD	589.86	589.86	589.86	589.86	589.86
R1	NV	11.92	12.16	12	11.92	12.08
	TD	1222.12	1211.55	1217.73	1210.69	1197.97
R2	NV	2.73	3	2.73	2.73	2.73
	TD	975.12	1001.12	967.75	951.51	952.17
RC1	NV	11.50	12.25	11.63	11.50	11.50
	TD	1389.58	1418.77	1382.42	1384.17	1355.97
RC2	NV	3.25	3.37	3.25	3.25	3.25
	TD	1128.38	1170.93	1129.19	1119.24	1116.38

appears satisfactory, considering that all approaches in the literature were tailored for the VRPTW, most of them actually aiming first to reduce the travelled distance.

5.3.3 Results for the real-life instances

To further prove the feasibility and effectiveness of the presented algorithm under real situations, we collected and investigated the real data of the NAFTAL petroleum company. This company is responsible for delivering various kinds of fuel (gasoline, premium gasoline, aviation gasoline, kerosene and diesel fuel) from 70 depots, serving more than 3500 petrol stations and using around 1500 tank trucks. In this experiment, we used instances that we obtained from the regional depot of petroleum products, Caroubier depot, in the city of Algiers in Algeria. For this depot, the company provided us with daily data for a period of 15 days.

The data consist of the cities where the customers are located and the associated distance matrix, the order quantities with their time windows and tank truck related information, such as the number of tanks and their capacities. The fleet dedicated to the replenishment of petrol stations consists of 20 tank trucks, five of which are owned by private companies. The travel time in minutes between each pair of petrol stations is calculated by multiplying the travel distance in kilometres by a definite constant 1.200 based on the average travel speed, 50 km per hour or 0.833 km per minute. The travel cost between each pair of petrol stations is calculated by multiplying the travel distance by a definite constant 0.685 given by the company. As mentioned earlier, the private trucks are hired with a penalty cost. This cost is calculated by multiplying the travel distance between each pair of petrol stations by a definite constant $f_2 = 0.497$ based on the average rental price of private vehicles. In Table 6, the instances are denoted by a name that allows one to identify their customers per day. In particular, the names have the form $D - n$ where D is the working day and n is the number of customers. For example the instance "1–41" denotes the 41 customers to be delivered to on the first working day.

In the first place, we tried to solve these 15 instances using CPLEX 12.6. Unfortunately, it failed to find the optimal solution for most of these trials and sometimes produced an out of memory error. Because of this, we completed our assessment by comparing our solutions with those provided by the company. For CPLEX, a maximum of a 7200s CPU is imposed on the solution time, and the final optimality gap in percentage (Gap_f (%)) is provided.

Table 6 shows the solutions obtained by our algorithm, the optimal solutions obtained by CPLEX and the solutions extracted from the company plans over a testing period of 15 days. In this table, we report the number of vehicles used for the service (NV), the total distance (TD), the cost (u), the CPU times in seconds (CPU), and the final optimality gap in percentage (Gap_f (%)) when CPLEX 12.6 is used. Results in this table indicate that our algorithm provides the optimal solutions to instances 3–25, 6–32 and 8–27 with a substantially lower computation time. Furthermore, CPLEX cannot provide a feasible solution (within the time limit of 7200 seconds) to seven instances (4–62, 5–83, 7–89, 10–79, 12–71, 14–61 and 15–83).

As a comparison with the solutions extracted by the company, we may conclude that our algorithm has a better performance on every measure. The main reason that the company system does not calculate the CPU times is because it develops the daily delivery plans manually using MS Excel.

5.4 Sensitivity analysis of the adjustment option on a real case

We have noted earlier that threshold parameter λ_{ip} is an important and integral component affecting the

Table 6 Daily results using the real data of Caroubier depot

Day	Method	NV	DT	$Cost\ (\times 10^2)$	CPU (s)	$Gap_f\ (\%)$
1–41	Cplex	4	536	47,570	7200	82.17
	Company	6	610	56,390	/	
	Our solution	4	527	47,292	315.43	
2–46	Cplex	6	881	121,766	7200	86.24
	Company	8	991	151,494	/	
	Our solution	6	881	121,766	243.12	
3–25	Cplex	3	379	23,694	6578	
	Company	4	443	36,512	/	
	Our solution	**3**	**379**	**23,694**	**317.82**	
4–62	Cplex	***NO SOL***				
	Company	9	1014	107,938	/	
	Our solution	7	918	90,747	403.82	
5–83	Cplex	***NO SOL***				
	Company	11	1475	11,382		
	Our solution	9	1323	101,745	515.39	
6–32	Cplex	3	488	29,178	4915	
	Company	4	510	40,129	/	
	Our solution	**3**	**488**	**29,178**	**182.25**	
7–89	Cplex	***NO SOL***				
	Company	15	1577	151,436	/	
	Our solution	12	1479	129,305	398.76	
8–27	Cplex	3	283	25,886	3755	
	Company	5	356	31,500	/	
	Our solution	**3**	**283**	**25,886**	**143.29**	
9–40	Cplex	6	853	55,350	7200	79.12
	Company	8	1030	77,892	/	
	Our solution	6	841	52,773	204.74	
10–79	Cplex	***NO SOL***				
	Company	11	1692	213,915	/	
	Our solution	9	1537	174,676	479.38	
11–28	Cplex	6	911	79,000	7200	89.12
	Company	7	1112	91,503	/	
	Our solution	6	913	79,023	207.19	
12–71	Cplex	***NO SOL***				
	Company	9	1233	147,419	/	
	Our solution	8	1129	130,771	210.16	
13–33	Cplex	7	1014	95,310	7200	69.23
	Company	9	1084	99,459	/	
	Our solution	7	1003	95,163	309.80	
14–61	Cplex	***NO SOL***				
	Company	10	1305	90,417	/	
	Our solution	8	1091	71,460	92.01	
15–83	Cplex	***NO SOL***				
	Company	13	1567	128,979	/	
	Our solution	10	1387	110,388	429.52	

Bold indicates optimal solutions

Table 7 Impact of the adjustment option on the daily costs

λ_{ip}	Total	Quantity (m^3)			NV		
(%)	*Cost* ($\times 10^2$)	Requested	Delivered	Gap (%)	Company	Private	Total
0	129,305	975	975	0.00	9	3	12
1	129,191	975	972.25	−0.28	9	3	12
2	128,609	975	969.75	−0.54	9	2	11
3	128,005	975	962.5	−1.28	9	2	11
4	127,319	975	955	−2.05	8	2	10
5	126,903	975	951.25	−2.44	8	2	10
6	126,180	975	944.75	−3.10	8	2	10
7	125,813	975	939.25	−3.67	8	2	10
8	124,298	975	918	−5.85	8	1	9
9	123,677	975	898.25	−7.87	8	1	9
10	121,963	975	890.5	−8.67	8	0	8

performance of the adjustment option in the MCVRPTW. To observe its role in the solution's quality, we perform a sensitivity analysis by solving the problem on the daily data for varying values of λ_{ip} between 0 and 10 %. Note that we did not perform the algorithm to better observe the effect of λ_{ip} values. With these values, we implemented the plan of the *seventh* day only and resolved the problem when excluding and including the adjustment option. In Table 7, we report the daily results obtained and we show the effect of threshold parameter λ_{ip} on (a) the total cost figures, (b) the quantities delivered and (c) the vehicles used. The results show that the solution quality is very sensitive to the threshold parameter. This is indeed an expected result since the adjustment option attempts to assign the demands taking more account of the capacity constraints. From this table, the following three points can be observed. (1) All the total costs decrease when customer quantities are adjusted, and this decrease is inversely proportional to the increase of the adjustment rate. (2) The total delivered quantity and the vehicles used decrease when λ_{ip} values become increasingly large. (3) Overall, the solution quality varies in different ratios, indicating that the solution quality is very sensitive to travelled distance, quantities delivered and even to the type of vehicle used.

6 Conclusion

The specific problem we tackled in this paper, called the multi-compartment vehicle routing problem with time windows (MCVRPTW), originated from a real-life application concerning the distribution of fuel. In this problem, we focused on the daily replenishment-planning problem that the biggest Algerian petroleum company is facing. In particular, the vehicles have multiple compartments and customers require to be served during a given time window. Because of the loading aspect, an additional question arises concerning the assignment of product types to vehicles.

The main contributions of this paper include (1) a description and formulation of the problem inspired by a real-life application, and (2) the development of an effective heuristic solution procedure, which combines the loading and the routing problems. This is an advance over existing work, in which most researchers take a two-stage framework where the routing problem acts as the main problem and iteratively calls for specific procedures to deal with the loading sub-problem. (3) The introduction of a Kolmogorov–Smirnov statistic in order to explore the solutions space is used, unlike most of the relevant approaches, which tend to use the classical moves. (4) The conduct of a series of numerical experiments on benchmark instances and an analysis of a real case in fuel distribution to demonstrate the effectiveness of the proposed approach are adopted.

Concerning the experiments on benchmark instances, Solomon's 56 VRPTW 100-customer instances have been modified in a way that reflects real-life situations. For this purpose, a comparison is made between the real MCVRPTW and what we call the MCVRPTW-AD (MCVRPTW with adjustment) in which the quantities loaded in the compartments can be adjusted up to a given threshold. This particular problem occurs quite frequently when the demands of petrol stations are high in winter. Under this scenario, we conducted experiments on how the algorithm performs when excluding and including the adjustment option. We demonstrated how the number of vehicles and the total distance can be reduced when the adjustment option is allowed and how this reduction depends on the fixed threshold.

Our solutions are also compared to CPLEX and to the heuristics reported in the literature. The obtained results

show that our approach is competitive for the VRPTW and highly effective for the MCVRPTW instances. As for the realistic instances, we solved a real case where petrol stations need to be replenished over a planning horizon of 15 days. The comparative analysis shows that our results are better than those produced by the dispatcher on every measure in terms of total travel distance and number of vehicles.

As for prospects, we envision producing more efficient solutions by adapting known metaheuristics, such as Particle Swarm Optimization (PSO), a Genetic Algorithm (GA) or Simulated Annealing (SA), to the problem and by adjusting the parameters of the algorithm because, often, in a metaheuristic, the selection of good parameter values significantly affects the quality of solutions. A hybridization of clever heuristics with complex search methods and an examination of penalty functions are also on the agenda.

Acknowledgments We gratefully acknowledge the anonymous referees for carefully reading the paper and for their helpful comments and suggestions.

References

1. Avella P, Boccia M, Sforza A (2004) Solving a fuel delivery problem by heuristic and exact approaches. Eur J Oper Res 152(1):170–179. doi:10.1016/S0377-2217(02)00676-8
2. Bausch DO, Brown G, Ronen D (1995) Consolidating and dispatching truck shipments of Mobil heavy petroleum products. Interfaces 25:117. doi:10.1287/inte.25.2.1
3. Ben Abdelaziz F, Roucairol C, Bacha C (2002) Deliveries of liquid fuels to SNDP gas stations using vehicles with multiple compartments. In: International conference on systems man and cybernetics 2002 IEEE, Hammamet, Tunisia. doi:10.1109/ICSMC.2002.1168021
4. Berger J, Barkaoui M (2004) A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Comput Oper Res 31(12):2037–2053. doi:10.1016/S0305-0548(03)00163-1
5. Braysy O, Gendreau M (2002) Tabu search heuristics for the vehicle routing problem with time windows. TOP 10(2):211–237. doi:10.1007/BF02579017
6. Brown GG, Graves WG (1981) Real-time dispatch of petroleum tank trucks. Manag Sci 27(1):19–31. doi:10.1287/mnsc.27.1.19
7. Brown GG, Ellis CJ, Graves WG, Ronen D (1987) Real-time, wide area dispatch of mobil tank trucks. Interfaces 17(1):107–120. doi:10.1287/inte.17.1.107
8. Bullnheimer B, Hartl RF, Strauss C (1999) An improved ant system algorithm for the vehicle routing problem. Ann Oper Res 89:319–328
9. Chiang WC, Russell RA (1997) A reactive tabu search metaheuristics for the vehicle routing problem with time windows. INFORMS J Comput 9(4):417–430. doi:10.1287/ijoc.9.4.417
10. Cordeau JF, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. J Oper Res Soc 52(8):928–936. doi:10.1057/palgrave.jors.2601163
11. Cornillier F, Boctor FF, Laporte G, Renaud J (2008a) An exact algorithm for the petrol station replenishment problem. J Oper Res Soc 59(5):607–615. doi:10.1057/palgrave.jors.2602374
12. Cornillier F, Boctor FF, Laporte G, Renaud J (2008b) A heuristic for the multi-period petrol station replenishment problem. Eur J Oper Res 191(2):295–305. doi:10.1016/j.ejor.2007.08.016
13. Cornillier F, Boctor FF, Laporte G, Renaud J (2009) The petrol station replenishment problem with time windows. Comput Oper Res 36(3):919–935. doi:10.1016/j.cor.2007.11.007
14. Cornillier F, Boctor FF, Renaud J (2012) Heuristics for the multi-depot petrol station replenishment problem with time windows. Eur J Oper Res 220:361–369. doi:10.1016/j.ejor.2012.02.007
15. Day JM, Wright PD, Schoenherr T, Venkataramanan M, Gaudette K (2009) Improving routing and scheduling decisions at a distributor of industrial gasses. Omega 37(1):227–237. doi:10.1016/j.omega.2006.11.007
16. El Fallahi A, Prins C, Wolfler Calvo R (2008) A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Comput Oper Res 35(5):1725–1741. doi:10.1016/j.cor.2006.10.006
17. Franz LS, Woodmanse J (1990) Computer-aided truck dispatching under conditions of product price variance with limited supply. J Bus Logist 11(1):127–139
18. Gambardella LM, Taillard E, Agazzi G (1999) MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: New ideas in optimization, pp 63-76
19. Glover F, Laguna M (1997) Tabu search. Kluwer, Boston
20. Lau HC, Sim M, Teo KM (2003) Vehicle routing problem with time windows and a limited number of vehicles. Eur J Oper Res 148(3):559–569. doi:10.1016/S0377-2217(02)00363-6
21. Malépart V, Boctor FF, Renaud J, Labilois S (2003) Nouvelles approches pour l'approvisionnement des stations d'essence. Revue Franaise de Gestion Industrielle 22:15–31
22. Nagata Y, Braysy O, Dullaert W (2010) A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput Oper Res 37(4):724–737. doi:10.1016/j.cor.2009.06.022
23. Ng WL, Leung S, Lam J, Pan S (2008) Petrol delivery tanker assignment and routing: a case study in Hong Kong. J Oper Res Soc 59:1191–1200. doi:10.1057/palgrave.jors.2602464
24. Nussbaum M, Sepulveda M (1997) A fuel distribution knowledge-based decision support system. Omega 25(2):225–234. doi:10.1016/S0305-0483(96)00059-X
25. Popović D, Vidović M, Radivojević G (2012) Variable neighbourhood search heuristic for the inventory routing problem in fuel delivery. Expert Syst Appl 39(18):13390–13398. doi:10.1016/j.eswa.2012.05.064
26. Renaud J, Bolduc MC, Laporte G, Boctor F (2010) A tabu search heuristic for the split delivery vehicle routing problem with production and demand calendars. Eur J Oper Res 202:122–130
27. Ronen D (1995) Dispatching petroleum products. Oper Res 43(3):379–387. doi:10.1287/opre.43.3.379
28. Solomon M (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265. doi:10.1287/opre.35.2.254
29. Taillard E, Badeau P, Gendreau M, Geurtin F, Potvin JY (1997) A tabu search heuristic for the vehicle routing problem with time windows. Transp Sci 31(2):170–186. doi:10.1287/trsc.31.2.170
30. Tan KC, Lee LH, Ou K (2001a) Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. Eng Appl Artif Intell 14(6):825–837. doi:10.1016/S0952-1976(02)00011-8

31. Tan KC, Lee LH, Ou K (2001) A messy genetic algorithm for the vehicle routing problem with time window constraints. In: IEEE congress on evolutionary computation, vol. 1, pp 679–686. doi:10.1109/CEC.2001.934457
32. Tan KC, Chew YH, Lee LH (2006) A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. Comput Optim Appl 34(1):115–151. doi:10.1007/s10589-005-3070-3
33. Taqa allah D, Renaud J, Boctor FF (2000) Le problme d'approvisionnement des stations d'essence. J Eur des Syst Autom 34:11–33
34. Toth P, Vigo D (2014) Vehicle routing: problems, methods, and application, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia
35. Toth P, Vigo D (2003) The granular tabu search and its application to the vehicle routing problem. INFORMS J Comput 15(4):333–348. doi:10.1287/ijoc.15.4.333.24890
36. Van der Bruggen L, Gruson R, Salomon M (1995) Reconsidering the distribution of gasoline products for a large oil company. Eur J Oper Res 81(3):460–473. doi:10.1016/0377-2217(94)00189-J
37. Vidal T, Crainic TG, Gendreau M, Prins C (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time windows. Comput Oper Res 40(1):475–489. doi:10.1016/j.cor.2012.07.018
38. Vidović M, Popović D, Ratković B (2014) Mixed integer and heuristics model for the inventory routing problem in fuel delivery. Int J Prod Econ 147:593–604. doi:10.1016/j.ijpe.2013.04.034