Computers & Operations Research 65 (2016) 64–75

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A 2-stage method for a field service routing problem with stochastic travel and service times

S. Binart [c,d,*], P. Dejax [b], M. Gendreau [c], F. Semet [a]

[a] CRIStAL UMR 9189, École Centrale de Lille, Cité Scientifique, 59650 Villeneuve d'Ascq, France
[b] LUNAM, École des Mines de Nantes, IRCCyN, 4 rue Alfred Kastler, 44307 Nantes, France
[c] CIRRELT et MAGI, École Polytechnique de Montréal, 2900 Boulevard Edouard-Montpetit, Canada, Montréal QC H3T 1J4
[d] CRIStAL UMR 9189, Université de Lille 1, Cité Scientifique, 59650 Villeneuve d'Ascq, France

ARTICLE INFO

Available online 10 July 2015

Keywords:
Field service routing
Stochastic travel times
Stochastic service times

ABSTRACT

In this paper, we consider a specific variant of the field service routing problem. It consists in determining vehicle routes in a single period to serve two types of customers: mandatory and optional. Mandatory customers have to be served within a specified time window whereas optional customers may be served (or not) within the planning horizon. For more realism, we assume that service as well as travel times are stochastic and also that there are multiple depots. The objective is to visit as many optional customers as possible while minimizing the total travel time. To tackle this problem, we propose a 2-stage solution method: the planning stage and the execution stage. We decompose the planning stage into two phases: the design of a skeleton of mandatory customers and the insertion of optional customers in this skeleton. In the execution stage, we proceed to a real-time modification of the planned routes to face stochastic travel and service times and to enable time windows to be respected.

1. Introduction

The field service routing problem consists, given a limited number of technicians, in determining a set of optimal technician routes to serve customer requests, while ensuring that each technician has the required skills for his tasks. The problem we are dealing with in this paper is a variant of the single-period field service routing problem without technician skills. In this variant, we suppose that all customers are known a priori and we distinguish between two types of customers: mandatory and optional customers. Optional customers have an associated time window corresponding to the time horizon and may be postponed at any time. Mandatory customers have an associated hard time window (they must be served within this time window). We associate to each technician a vehicle and we suppose that each vehicle has unlimited capacity, its own origin and destination depots and must return to this destination depot by the end of the period (hard time window). We consider that travel and service times for all customers (mandatory and optional) are stochastic. The objective is to visit as many optional customers as possible while minimizing the total travel time. This problem can also be seen as a variant of the vehicle routing problem with time windows (VRPTW). Indeed, the VRPTW consists in determining a set of optimal vehicle routes serving all customers to meet customer demands within specified time windows, each of these routes starting and ending at a given depot. The objective in the VRPTW is to minimize the total traveled distance and also, sometimes, the number of vehicles. Compared to this classical VRPTW, our variant present some differences as we consider multiple depots, uncapacitated vehicles, priority within customers as well as stochastic travel and service times. In the remainder of this paper, we will call this variant the Multi-Depot Vehicle Routing Problem with Time Windows, Stochastic Service and Travel Times and with priority (MDVRPTWSSTT with priority).

A generic application of the MDVRPTWSSTT with priority we consider is the design of routes for technicians for repair and maintenance operations. Mandatory customers are requiring repair operations whereas optional customers are requiring a service (control, maintenance, meter-reading…). In this application, as it is about service routing, vehicles are used only for carrying material and personnel. Thus we can suppose that the vehicle capacity is unlimited. Moreover, as vehicles do not carry goods, they can have their own origin and destination depots (typically technician homes). Last, as the service provided to the customer may be a repair operation, we understand the need to consider stochastic service times. The particular application that we are considering corresponds to a real problem coming from a leading international company in the area of domestic water provision and treatment. In this company, technicians have to perform maintenance operations (corresponding to optional customers) and repair operations (corresponding to mandatory customers). It is described in Tricoire et al. [1,2] and Binart et al. [3] as well as Bostel et al. [4]. In this paper, we consider uncertainties

* Corresponding author.
E-mail address: sixtine.binart@ed.univ-lille1.fr (S. Binart).

http://dx.doi.org/10.1016/j.cor.2015.07.001
0305-0548/

in both travel times between clients and duration of service at customers' premises. Uncertainties are indeed a major problem in a real application of this kind, especially to reach efficient solutions during the real-time execution stage. To the best of our knowledge, no previous work has been reported on the problem we just defined. Therefore, we will first focus on literature dealing with variants of the field service routing problem and then on literature dealing with the vehicle routing problem with time windows (VRPTW) with common characteristics to the MDVRPTWSSTT with priority.

Since a few decades, the problem of field service routing has arised in the literature with the development of the service industry. In 2002, Grötschel et al. [5] address a realtime variant of the field service routing problem and propose a column generation method with a dynamic pricing strategy. More recently, Kovacs et al. [6] propose an adaptive large neighborhood search for solving the field service routing problem with and without team building. Borenstein et al. [7] and Delage et al. [8] consider the field service routing problem with stochastic service times. Borenstein et al. [7] proceed in many steps: they first partition customers into different clusters, then assign technicians to them. To get a soft clustering, they make the tasks located at the boundary between areas belong to all adjacent areas. After this soft clustering, they define some rules to allocate tasks to technicians. Delage et al. [8] proposes a two-step method: he first establishes workload planning and then he deals with stochastic service times thanks to a dynamic programming approach. Last, Cortes et al. [9] and Souyris et al. [10] address the field service routing problem with stochastic service times and a priority within customers (corresponding to a target response time). Tricoire et al. [1,2] and Bostel et al. [4] have been interested in the deterministic multiple depot, multiperiod field service routing with priority within customers (MDVRPTW with priority), distinguishing mandatory and optional customers. For this problem, they propose a column generation-based method [11] as well as a memetic algorithm [12]. In 2006, Dugardin et al. [13] deals with the single-period field service routing problem with priority within customers, but he considers stochastic travel times. However, he does not take into account the stochasticity when building the route plan. Given a route plan, he defines simple rules to react to the different events which may occur. Even if the VRPTW has been addressed in many papers, few are those taking into account stochastic travel and/or service times. Ando et al. [14], Russell et al. [15], Jie [16], and Tas et al. [17] formulate the VRPTW with stochastic travel times as an integer program and seek to minimize the weighted sum of total traveled distance and penalties related to the violation of time windows. Moreover, Ando et al. [14] and Russell et al. [15] minimize the number of vehicles used. To solve this problem, Russell et al. [15] and Tas et al. [17] propose a tabu search based algorithm whereas Jie [16] present an evolutionary algorithm.

Some authors have been interested in the VRPTW with stochastic service times. In 2007, Flatberg et al. [18] propose a scenario-based approach with local search. In 2011, Lei et al. [19] propose a two-stage dynamic programming model with recourse, where the recourse consists in going back to the depot as soon as the end of the depot time window is reached. To solve the VRPTW with stochastic travel and service times, Wang et al. [20] present assignment models whereas Li et al. [21] propose a tabu search algorithm. In 2007, Zeimpekis et al. [22] add to this problem priorities within customers (depending on profit, on time window and on travel cost to access a customer) but limit the problem to a single vehicle. For this variant, they propose a variant of the *S*-algorithm [23].

In this paper, we address a different problem as we consider a variant of the VRPTW with multiple depots, priority within customers and stochastic travel and service times. All of these characteristics have not been considered all together previously in the literature. We assume that all customers are known a priori, as well as the minimal, modal and maximal values for speed and service times. Regarding the number of mandatory customers, we suppose that it is sufficient, in order for our method to be consistent. This assumption is not restrictive since, if we do not have enough mandatory customers, we can decide to make some optional customers become mandatory. Making these assumptions, we propose a two-stage method: a planning stage followed by an execution stage. As Delage et al. [8], we decompose the planning stage into two phases: (i) we build routes considering only mandatory customers (we call the set of these routes "skeleton"); (ii) we insert optional customers in this skeleton. At the beginning of the execution stage, we have a planned route for each vehicle. In this stage, we use dynamic programming to deal with the stochasticity on travel and service times. The key idea in this two-stage method is to use the optional customers as buffer to absorb variations on travel and service times. The remainder of the paper is organized as follows. We present the planning stage in Section 2, the execution stage in Section 3, computational results of the two-stage method in Section 4 and we conclude in Section 5.

2. Planning stage

In the planning stage, we aim at building optimal routes containing both mandatory and optional customers. Assuming that lower and upper bounds on travel and service times are known (as mentioned before), we proceed in two phases: first, we build a skeleton of routes serving mandatory customers and then we insert optional customers in this skeleton. In phase I (skeleton design), we formulate the problem as a mixed integer program and we solve it exactly using a commercial solver. In the second phase (insertion of optional customers), we formulate the problem as an integer program and we proceed in two steps: we first solve this model with pessimistic estimates using a branch and cut algorithm or a Lagrangian decomposition method and we then repair and improve the solution with a heuristic method.

2.1. Phase I: skeleton design

In this step, we consider only mandatory customers, which are a priori known and have an associated time window. In order to design the skeleton of routes including mandatory customers only, we just have to solve a m-TSPTW on mandatory customers. As we do not allow any delay for serving mandatory customers, we consider that travel and service times are maximal. Let K be the set of vehicles and M the set of mandatory customers. We note o^k and d^k the origin and destination depots for vehicle k and $[e_i, l_i]$ the time window for customer i (service should begin after e_i and before l_i). Let $\underline{\sigma}_i$ and $\overline{\sigma}_i$ be respectively the minimal and maximal service time for customer i, $\underline{\tau}_{ij}$ and $\overline{\tau}_{ij}$ be respectively the minimal and maximal travel time between i and j. Let T be a large constant. We define the following variables. A binary variable x_i^k indicates if mandatory customer i is served by vehicle k. A binary variable y_{ij}^k indicates if customer i is served just before j by vehicle k, and last t_i corresponds to the time at which service starts at customer i. Then the skeleton design is modelled as follows:

Model 1 (M1):

$$\min \sum_{k \in K} \sum_{i \in M \cup \{o^k\}} \sum_{j \in M \cup \{d^k\}} \overline{\tau}_{ij} y_{ij}^k$$

subject to:

$$\sum_{k \in K} x_i^k = 1 \qquad \forall i \in M \tag{1}$$

$$\sum_{j \in M \cup \{d^k\}} y_{ij}^k = x_i^k \qquad \forall i \in M, k \in K \tag{2}$$

$$\sum_{i \in M \cup \{o^k\}} y_{ij}^k = x_j^k \qquad \forall j \in M, k \in K \tag{3}$$

$$\sum_{i \in M \cup \{d^k\}} y_{o^k i}^k = 1 \qquad \forall k \in K \tag{4}$$

$$e_i \leqslant t_i \leqslant l_i \qquad \forall i \in M \cup \{o^k; d^k\} \tag{5}$$

$$t_j \geqslant t_i + \overline{\sigma}_i + \overline{\tau}_{ij} + \sum_{k \in K} T(y_{ij}^k - 1) \qquad \forall i \in M \cup \{o^k\}, j \in M \cup \{d^k\} \tag{6}$$

$$y_{ij}^k \in \{0;1\} \qquad \forall k \in K, i \in M \cup \{o^k\}, j \in M \cup \{d^k\}$$
$$x_i^k \in \{0;1\} \qquad \forall i \in M, k \in K$$
$$t_i \geqslant 0 \qquad \forall i \in M \cup \{o^k; d^k\}$$

Constraints (1) state that every mandatory customer must be served exactly once. Constraints (2) and (3) are in-degree and out-degree constraints. Constraints (4) state that each vehicle should leave its origin, even if going directly to destination. Constraints (5) ensure that the respect of time windows (service for customer i must begin within the time window $[e_i, l_i]$). Finally, constraints (6) are precedence constraints, which also ensure the elimination of subtour. Since the number of mandatory customers is low in the instances considered, this model is solved exactly using a commercial solver.

2.2. Phase II: inserting optional customers

Once the skeleton of routes serving mandatory customers is built, we have for each vehicle a sorted list of mandatory customers to be visited. In order to improve the quality of service, we update the earliest and latest times for beginning service, e_i and l_i, associated with mandatory customer i. They correspond to the beginning service time respectively in the best and in the worst case. Therefore we state $l_i = t_i$ and we calculate e_i with the minimal travel and service times (while ensuring that e_i respects the previous time window) according to the routes of the skeleton.

In this phase, we introduce the concept of *segment*, which is a route portion between two successive mandatory customers (the origin depot and destination depot of each vehicle are considered as mandatory customers). Segment p has three main characteristics: an origin o^p, a destination d^p and a length Δ^p. As we wish to use optional customers as buffer to absorb variations of travel and service times, we define this length as the largest possible one: $\Delta^p = l_{d^p} - e_{o^p} - \underline{\sigma}_{o^p}$ where l_d^p is the latest time to begin service at customer d^p, $\underline{\sigma}_{o^p}$ the minimal service time for the origin customer o^p and e_o^p the earliest time to begin service at customer o^p. Note that, for a given vehicle, segments are sorted the following way (see Fig. 1 below): segment $(p+1)$ has for origin d^p (the destination customer of segment p).

With this new concept of segment, the problem of inserting optional customers in the skeleton consists in establishing routes associated with each segment of the skeleton while ensuring on each segment p that the length of the route does not exceed Δ^p. Two objective functions are considered in this insertion phase: the maximization of the profit associated with servicing optional customers and then, the minimization of the total traveled time. To deal with these objectives, we use the classical weighted sum technique to combine them into a single one, where α is the weight for the travel time ($\alpha \in [0; +\infty[$). Let O be the set of optional customers and P the set of segments over all the routes. Customer i has an associated profit p_i. We use reference values $\tilde{\tau}_{ij}$ and $\tilde{\sigma}_i$ respectively for travel and service times. We define the following variables. A binary variable x_i^p indicates if customer i is served on segment p, and a binary variable y_{ij}^p indicates if customer i is served just before j on segment p. With these notations, we formulate the insertion of optional customers in the skeleton of routes as follows:

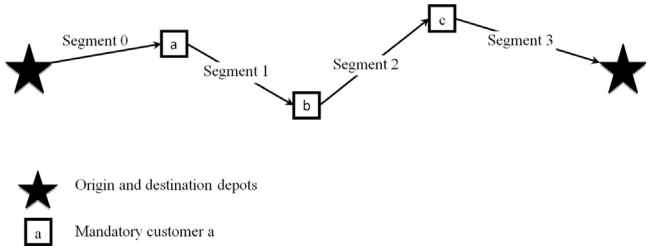

Fig. 1. Segment order for a given vehicle (route).

Model 2 (M2):

$$\max \sum_{p \in P} \sum_{i \in O} p_i x_i^p - \alpha \sum_{p \in P} \sum_{i \in O} \sum_{j \in O} \tilde{\tau}_{ij} y_{ij}^p$$

subject to:

$$\sum_{p \in P} x_i^p \leqslant 1 \qquad \forall i \in O \tag{7}$$

$$\sum_{j \in O \cup \{d^p\}} y_{ij}^p = x_i^p \qquad \forall i \in O, p \in P \tag{8}$$

$$\sum_{i \in O \cup \{o^p\}} y_{ij}^p = x_j^p \qquad \forall j \in O, p \in P \tag{9}$$

$$\sum_{i \in O \cup d^p} y_{o^p i}^p = 1 \qquad \forall p \in P \tag{10}$$

$$\sum_{i \in O \cup \{o^p\}} \sum_{j \in O \cup \{d^p\}} \tilde{\tau}_{ij} y_{ij}^p + \sum_{i \in O} \tilde{\sigma}_i x_i^p \leqslant \Delta^p \qquad \forall p \in P \tag{11}$$

$$\sum_{i \in S} \sum_{j \in S} y_{ij}^p \leqslant \sum_{i \in S \setminus \{l\}} x_i^p \qquad \forall S \subset O, |S| \geqslant 2, \forall l \in S, p \in P \tag{12}$$

$$y_{ij}^p \in \{0;1\} \qquad \forall i \in O \cup \{o^p\}, j \in O \cup \{d^p\}, p \in P$$
$$x_i^p \in \{0;1\} \qquad \forall i \in O, p \in P$$

Constraints (7) state that each optional customer is served at most once. Constraints (8) and (9) are in-degree and out-degree constraints. Constraints (10) ensure that each vehicle leaves the origin of the segment (eventually to go directly to the destination of the segment). Constraints (11) force the length of the route associated with a segment to be smaller than the length of this segment. Constraints (12) are subtour elimination constraints. They are necessary in this model because optional customers do not have any time window.

Regarding the value setting for travel and service times $\tilde{\tau}_{ij}$ and $\tilde{\sigma}_i$, many choices are possible. If we choose minimal values (optimistic estimates), we may obtain solutions with empty routes while some routes serve a large number of customers each. It is an admissible solution as long as we are in the optimistic case. However, during execution stage, many customers remain unserviced while some routes remain empty. In order to avoid this pitfall, we have to balance workload over vehicles during the insertion of optional customers. We thus perform the insertion of optional customers in two steps. First, we insert optional customers with *pessimistic estimations* of travel and

service times. Next, we insert optional customers with *optimistic estimations* of travel and service times, while keeping the assignment of customers to vehicles obtained in the first step. With these choices, we build route portions on segments with a duration limit corresponding to the length of the segments. But if we consider a route containing several segments, this route has a duration limit corresponding to the sum of the lengths of the associated segments, i.e. $l_{d^p} - e_{o^p} - \underline{\sigma}_{o^p}$ (optimistic estimate for the length of segment p). The obtained route can thus be much longer than the time horizon. Indeed, preliminary experiments lead us to observe that the feasibility probability for some routes can fall to 0.2. As a consequence, during the execution stage, we completely changed the planned routes. In order to avoid this, we choose to ensure a minimal probability for the routes to be feasible. The insertion of optional customers now consists in inserting first optional customers with *pessimistic estimations* of travel and service times. Then, we repair and improve the solution obtained, while ensuring that each route has a sufficient probability to be feasible (greater than a given threshold).

We develop a basic branch and cut algorithm for solving ($M2$). We obtain a first subproblem by relaxing the subtour elimination constraints (12) as well as the integrality constraints on the variables. Then, constraints (12) are added dynamically to improve the upper bound. When no cuts can be identified, branching is performed in priority on the x_i variables. The fractional variable x_i^p closest to 0.5 is selected for branching. When all x_i^p variables are integer, y_{ij}^p variables are considered using the same selection rule. As inserting optional customers with pessimistic estimations on small instances with this branch and cut algorithm can be time-consuming, we propose some speedup techniques to improve computation times. In order to solve larger instances, we also propose a Lagrangian relaxation to decompose the problem into one subproblem per segment. As the routes obtained by both methods (branch and cut and Lagrangian relaxation) may have a low probability to be feasible, we propose to repair and improve these routes afterwards in order to get routes with a sufficient feasibility probability. In what follows, we will first detail the different speedup techniques we propose for the branch and cut algorithm in Section 2.2.1, present the Lagrangian relaxation method in Section 2.2.2 and then describe the repair and improve algorithms in Section 2.2.3.

2.2.1. Speedup techniques for the branch and cut algorithm

As we mentioned before, solving model ($M2$) with a branch and cut algorithm can be very time-consuming. In order to improve computation times, we propose in this section different speedup techniques: preprocessing, an insertion heuristic to build an initial integer solution, reachability cuts and subset elimination inequalities.

Preprocessing: A commonly used speedup technique is to make some preprocessing to fix some variables before beginning the branch and cut method. This preprocessing is based on the idea that, when inserting optional customers in the skeleton, some customers cannot possibly be served on some segments. In order to avoid this, when comparing maximal travel and service times to the length of segments, we obtain conditions which allow us to set the values of some variables or to strengthen the formulation by adding some valid inequalities:

Proposition 1. *The following clauses are valid*:

$$\text{If} \quad \overline{\tau}_{o^p j} + \overline{\sigma}_j + \overline{\tau}_{jd^p} \geqslant \Delta^p, \quad \text{then } x_j^p = 0 \tag{13}$$

$$\text{If} \quad \overline{\tau}_{o^p i} + \overline{\sigma}_i + \overline{\tau}_{ij} + \overline{\sigma}_j + \overline{\tau}_{jd^p} \geqslant \Delta^p, \quad \text{then } y_{ij}^p = 0 \tag{14}$$

$$\text{If} \quad \left.\begin{array}{l} \overline{\tau}_{o^p i} + \overline{\sigma}_i + \overline{\tau}_{ij} + \overline{\sigma}_j + \overline{\tau}_{jd^p} \geqslant \Delta^p \\ \overline{\tau}_{o^p j} + \overline{\sigma}_j + \overline{\tau}_{ij} + \overline{\sigma}_i + \overline{\tau}_{id^p} \geqslant \Delta^p \end{array}\right\} \quad \text{then } x_i^p + x_j^p \leq 1 \tag{15}$$

Clause (13) states that customer i cannot be visited on segment p if the total time needed to serve only i on segment p (travel time from o^p to i, service time at i and travel time from i to d^p) exceeds the length Δ^p of segment p. Likewise, clause (14) indicates that customer j cannot be served after customer i if the total time needed to serve only customer i followed by customer j on segment p (travel time from o^p to i, service time at i, travel time from i to j, service time at j and travel time from j to d^p) exceeds the length Δ^p of this segment. Finally, clause (15) is a valid inequality based on clause (14). This clause expresses that if both the total time needed to visit only i and j in this order and the one needed to visit only j and i in this order on segment p exceed the length Δ^p of segment p, then i and j cannot be served together on segment p. Indeed, using clause (14) on the conditions of clause (15), we obtain $y_{ij}^p = 0$ and $y_{ji}^p = 0$ (i.e. j cannot be served after i and i cannot be served after j on segment p). Thus, we conclude that i and j cannot be served together on segment p.

Insertion Heuristic: To build an initial solution for the branch and cut, we propose a heuristic based on the insertion of customers. This heuristic can be described as follows. We compute the insertion costs of each customer on each segment (if a customer i cannot be inserted on a segment p, the insertion cost c_i^p associated will be $+\infty$). Then we solve an assignment problem (assigning customers to segments). Given the assignment solution, we have for each segment a route serving potentially one optional customer. Thus, insertion costs are possibly modified for the unassigned customers. This process is then reiterated until one of the following stopping conditions is satisfied:

(i) All optional customers have been inserted ;
(ii) No optional customer was inserted in the last iteration.

Reachability Cuts: In order to speedup the branch and cut algorithm, we propose to strengthen subtour elimination constraints (12) in model ($M2$). They can be reformulated as follows. Let $\delta^-(S) = \{(i,j) | i \notin S, j \in S\}$ for a set of customers S, we have

$$\sum_{(k,l) \in \delta^-(S)} y_{kl}^p \geqslant x_i^p \; \forall S \subset O, |S| \geq 2, \quad \forall i \in S, p \in P \tag{16}$$

Constraints (16) are known to be equivalent to constraints (12). Lysgaard [24] propose to reinforce such constraints by defining the so-called reachability cuts. Following the lines of Lysgaard [24], we define the reachability set A_i^{p-} as being the minimal set of arcs enabling access to customer i from the origin of segment p. For instance, if (o^p, a, b, i) is a feasible path (with respect to the maximum duration constraint) from o^p to i, then $\{(o^p, a); (a, b); (b, i)\} \subset A_i^{p-}$ as long as the time matrix satisfies the triangle inequality. Then, constraints (16) can be strengthened by considering the following reachability cuts:

$$\sum_{(k,l) \in \delta^-(S) \cap A_i^{p-}} y_{kl}^p \geqslant x_i^p \; \forall S \subset O, |S| \geq 2, \quad \forall i \in S, p \in P \tag{17}$$

In other words, let p be a segment, i a customer and S a set of customers (with $o^p \notin S$ and $i \in S$), if customer i is served on segment $p (x_i^p = 1)$, then there exists a path from the origin of the segment o^p to i, and every arc on this path belongs to A_i^{p-}. Especially, as $i \in S$ and $o^p \notin S$, there exists at least one arc in A_i^{p-} entering S. These constraints are clearly stronger than the classical subtour elimination constraints (SEC) since $(\delta^-(S) \cap A_i^{p-}) \subseteq \delta^-(S)$. In constraints (17), we can see that the number of terms in the lefthand side depends on the size of A_i^{p-}. Therefore, when the accessibility set A_i^{p-} is large, the constraint $R_i^{p-}(S)$ is not so strong compared with the associated SEC constraint. For a given customer, the size of the accessibility set increases with the length of the corresponding segment. For this reason, in a first approach, we choose to generate these new constraints only on segments whose length does not exceed a threshold $L_{\max}$ and to

generate the classical subtour elimination constraints on other segments. To separate the reachability cuts, we determine a priori the set of arcs A_i^{p-} for each $i \in O$. In the preprocessing phase, we create a list C^p of customers who can be served on each segment p (with respect to the length of the segments). For each customer i and segment p, we go through all arcs (j,k) such that $j \in C^p$ and $k \in C^p$. If the path (o^p, j, k, i) is feasible on segment p, we add the corresponding arcs to the set A_i^{p-} if they do not already belong to this set. Then, to identify violated reachability cuts, we consider all possible couples (customer i, segment p) and for each of them, we solve a maximum flow problem on the support graph $G = (\{j | j \in C^p\} \cup \{o^p\}, A_i^{p-})$.

Subset elimination inequalities: Another commonly used speedup technique consists in using a very well-known family of valid inequalities known as "subset elimination inequalities". These inequalities are derived from constraints (10).

Proposition 2. *Given a set S of customers and a segment p, let $L^p(S)$ be the length of the shortest path from node o^p to node d^p serving all customers contained in S. If $L^p(S) > \Delta^p$, then the following subset elimination inequality*:

$$x^p(S) \leqslant |S| - 1 \tag{18}$$

is valid.

Note that for $|S| = 2$, we have $x_i^p + x_j^p \leqslant 1$. These inequalities have already been generated when preprocessing. In our branch and cut algorithm, such inequalities are generated for $3 \leqslant |S| \leqslant S_{\max}$. Constraints (18) are separated heuristically. Given a solution $(\tilde{x}, \tilde{y})$ and a segment p, we first sort customers i verifying $x_i^p > 0$ in decreasing order of $\tilde{x}_i^p$. Given this sorted list, we identify k-tuples $\{i_1, i_2, \ldots, i_k\}$ verifying $x_{i_1}^p + x_{i_2}^p + \cdots + x_{i_k}^p > k-1$. For each of these k-tuples, we check first if $(o^p, i_1, i_2, \ldots, i_k, d^p)$ is admissible. In such case, there is no violated subset elimination inequality associated with this k-tuple and this segment. Otherwise, we identify the shortest path from o^p to d^p serving all customers of the k-tuple. If the shortest path is not admissible, we just identified a violated subset elimination inequality of size k with $S = \{i_1, i_2, \ldots, i_k\}$.

2.2.2. Lagrangian relaxation

We can observe that the problem of inserting optional customers in the skeleton of routes can be decomposed into subproblems (one subproblem per segment) if we disregard binding constraints (7). A solution approach thus consists in relaxing these constraints and in giving them a Lagrangian multiplier in the objective function, obtaining the Lagrangian relaxation $R(u)$ for a given value of $u = (u_1, u_2, \ldots, u_{|O|}) \geqslant 0$:

$$\max \sum_{p \in P} \sum_{i \in O} p_i x_i^p - \alpha \sum_{p \in P} \sum_{i \in O} \sum_{j \in O} \tilde{\tau}_{ij} y_{ij}^p + \sum_{i \in O} u_i \left(1 - \sum_{p \in P} x_i^p\right)$$

subject to constraints (8)–(12). In order to be able to break this problem down into subproblems, we reformulate the objective function as

$$\max \sum_{p \in P} \sum_{i \in O} (p_i - u_i) x_i^p - \alpha \sum_{p \in P} \sum_{i \in O} \sum_{j \in O} \tilde{\tau}_{ij} y_{ij}^p + \sum_{i \in O} u_i$$

As the last term $\sum_{i \in O} u_i$ is a constant for a given vector u, we can decompose $R(u)$ into subproblems. Considering segment p, subproblem $R^p(u)$ can be formulated as follows:

$$\max \sum_{i \in O} (p_i - u_i) x_i^p - \alpha \sum_{i \in O} \sum_{j \in O} \tilde{\tau}_{ij} y_{ij}^p$$

subject to

$$\sum_{j \in O \cup \{d^p\}} y_{ij}^p = x_i^p \qquad \forall i \in O \tag{19}$$

$$\sum_{i \in O \cup \{o^p\}} y_{ij}^p = x_j^p \qquad \forall j \in O \tag{20}$$

$$\sum_{i \in O \cup d^p} y_{o^p i}^p = 1 \tag{21}$$

$$\sum_{i \in O \cup \{o^p\}} \sum_{j \in O \cup \{d^p\}} \tilde{\tau}_{ij} y_{ij}^p + \sum_{i \in O} \tilde{\sigma}_i x_i^p \leqslant \Delta^p \tag{22}$$

$$\sum_{i \in S} \sum_{j \in S} y_{ij}^p \leqslant \sum_{i \in S \setminus \{l\}} x_i^p \qquad \forall S \subset O, |S| \geqslant 2, \forall l \in S \tag{23}$$

$$y_{ij}^p \in \{0;1\} \qquad \forall i \in O \cup \{o^p\}, j \in O \cup \{d^p\}$$

$$x_i^p \in \{0;1\} \qquad \forall i \in O$$

$R^p(u)$ is solved using the branch and cut method with speedup techniques mentioned above. In order to proceed to the insertion of optional customers in the skeleton, we have to solve the Lagrangian dual, which can be formulated as follows:

$$\min_{u \geqslant 0} \left(\max_{x} \sum_{p \in P} \sum_{i \in O} p_i x_i^p - \alpha \sum_{p \in P} \sum_{i \in O} \sum_{j \in O} \tilde{\tau}_{ij} y_{ij}^p + \sum_{i \in O} u_i \left(1 - \sum_{p \in P} x_i^p\right) \right)$$

subject to (19)–(23) To solve the Lagrangian dual, we apply a subgradient algorithm. For this algorithm, we need a good lower bound for the optimal value in order to ensure the convergence of the method. For this purpose, we build a feasible solution, using the insertion heuristic presented in Section (2.2.1) and we improve the obtained solution using string exchange, arc exchange and relocalisation (without moving mandatory customers). Let $\underline{\omega}$ be the value of this solution (lower bound value) and $u^0 = (0, 0, \ldots, 0)$, we proceed as follows at each iteration k of the subgradient algorithm:

Iteration k:

- $u = u^k$
- For each segment $p \in P$, solve $R^p(u^k)$ with pessimistic estimates for travel and service times, using the branch and cut method with speedup techniques.
- With the values $(x_i^p)_{i \in O, p \in P}$ of the obtained solution, calculate the value $z(u^k)$ of the solution of $R(u^k)$.
- Calculate the step length μ_k and the step directions D_i^k for each $i \in O$ with formulas:

$$\mu_k = \frac{\epsilon(z(u^k) - \underline{\omega})}{\sum_{i \in O} \left(1 - \sum_{p \in P} x_i^p\right)^2} \quad \text{and} \quad D_i^k = 1 - \sum_{p \in P} x_i^p$$

- Calculate the new vector u^{k+1} with $u_i^{k+1} = \max(u_i^k + \mu_k D_i^k, 0)$.
- $k \leftarrow k+1$

The subgradient algorithm terminates prematurely either when the gap between the best upper bound $\min_k z(u^k)$ and the best lower bound $\max_k \underline{\omega}^k$ falls under a given threshold or when the number of iterations reaches a given maximal value. This turns the subgradient algorithm into a Lagrangian heuristic. In order to accelerate convergence, we calculate a new lower bound at each iteration. After obtaining the solution at iteration k, we repair this solution (by removing repeated customers) and improve the solution (proceeding to successive string exchanges, arc exchanges and relocalisations without moving mandatory customers). We thus obtain a feasible solution, i.e., a new lower bound $\underline{\omega}^k$. The solution returned by the subgradient algorithm is the best feasible solution, i.e. the best lower bound. This Lagrangian heuristic provides us a good lower bound for the problem of inserting optional customers in the skeleton. As

mentioned previously, after this Lagrangian relaxation, we can obtain a solution with a very low probability to be feasible. We thus apply the repair and improve algorithms to the obtained solution (see Section 2.2.3).

2.2.3. Solution repairing and improving algorithm

When proceeding to the insertion of optional customers in the skeleton, we may obtain routes with a low probability to be feasible (see Section 2.2). Indeed, in phase II, we insert optional customers on segments, while ensuring on each segment p that the length of the route does not exceed Δ^p. As mentioned above, the length of a segment is defined with the formula $\Delta^p = l_{d^p} - e_{o^p} - \underline{\sigma}_{o^p}$. In the worst case, the length of the segment is rather $l_{d^p} - \underline{l}_{o^p} - \overline{\sigma}_{o^p}$. For this reason, the probability for a route to be feasible can be very low. To avoid this pitfall, we introduce a new parameter F as being the feasibility threshold (a route must have a probability to be feasible greater or equal to F). We propose a procedure that consists in repairing first the solution so that each route has a feasibility probability greater or equal to F. Then, we improve the solution (trying to insert unserved customers and to relocate customers while ensuring the probability for each route to be feasible to be greater or equal to F). A pseudo-code description of the repair procedure is given in Algorithm (1). Let P_k be the set of segments associated with vehicle k. For a vehicle k, given the planned route (and associated segments P_k), using the probability distributions for travel and service times, we can calculate at each mandatory customer the possible arrival times and associated probabilities. *CalculateProbaFeasible*(o^p, d^p) returns the probability for the route to be feasible at d^p (probability to arrive before l^p_d), knowing the possible arrival times and associated probabilities at o^p. We note *CustomerLargestDetour*(a, b) the function returning the customer located between a and b (a and b excluded) generating the largest detour regarding travel times. The repair algorithm consists in removing the customer generating the largest detour until the feasibility probability is sufficient.

Algorithm 1. Repair solution.

```
for each k ∈ K do
  for each p ∈ P_k do
    proba ← CalculateProbaFeasible(o^p, d^p)
    while proba < F do
      c ← CustomerLargestDetour(o^p, d^p)
      remove customer c
      proba ← CalculateProbaFeasible(o^p, d^p)
    end while
  end for
end for
```

The improvement algorithm pseudo-code description is given in algorithm (2). In this algorithm, let L be an empty list, U the list of unserved customers and N the total number of customers. *CustomerLargestDetour*(L) returns the customer generating the largest detour within the solution and not belonging to L. *BestInsertion*(c) returns the best possible insertion of customer c in the solution with respect to travel times. An insertion is feasible if it generates a route with a feasibility probability larger than F and if time windows are respected. *BestInsertion*(U), with U a list of customers, is a method consisting in getting the best possible insertion over all the routes and all the customers in U and proceeding to this insertion if this one is valuable until no more customers can be inserted. The improvement algorithm consists in proceeding to the best possible insertion, to improve the quality of the solution, until no more customer can be inserted. In this algorithm, an insertion will not be considered if the feasibility probability threshold is violated.

Algorithm 2. Improve solution.

```
U ← GetUnservedCustomers()
BestInsertion(U)
L ← []
while size(L) < N − size(U) do
  c ← CustomerLargestDetour(L);
  (bestRoute, bestPosition, bestProfit) ← BestInsertion(c);
  if bestProfit > 0 then
    remove customer c
    insert customer c on the route bestRoute in the position
   bestPosition
  end if
  add c to the list L
end while
```

3. Execution stage

At the beginning of the execution stage, we have a planned route for each vehicle obtained at the end of the planning stage. The goal of the execution stage is to adjust planned routes to reality. In this stage, we consider stochastic travel and service times and we react in real-time to face stochasticity. We thus use dynamic programming tools to determine the optimal policy. We first describe the probability distributions used in this stage and then we detail the dynamic programming algorithm.

3.1. Probability distributions

In this stage, as we just mentioned, we consider stochastic travel and service times. To model bounded travel and service times, we need truncated probability distributions. We also assume that time units are discrete (i.e. minutes). Hence we choose to model stochastic travel and service times with discrete triangular distributions. For service time at customer i, we use a symmetric discrete triangular distribution law between $\underline{\sigma}_i - 1$ and $\overline{\sigma}_i + 1$ where $\underline{\sigma}_i$ and $\overline{\sigma}_i$ correspond respectively to the minimal and maximal service time at customer i. For distance units we use the arbitrary units (a.u.) proposed by Tricoire [1]. Regarding travel time (min) per distance unit (a.u.) δ, we use a discrete triangular distribution between $\underline{\delta}$ and $\overline{\delta}$. As travel speed varies between 20 km/h and 50 km/h and as 41 km = 1000 a.u. (see Tricoire [1]), travel speed v (in a.u./min) lies in the interval $[8.13, 20.33]$ and travel time per distance unit (in min/a.u.) lies in the interval $[0.05; 0.12]$. In order to get discrete values for travel time per distance unit, we use a scale factor and we set $\underline{\delta} = \lceil 100/v_{\max} \rceil - 1$ and $\overline{\delta} = \lceil 100/v_{\min} \rceil + 1$ with mode $\lceil 100/v_{mode} \rceil$ where $v_{\min}$, $v_{\max}$ and v_{mode} correspond respectively to the minimal, maximal and modal travel speed (in u.a./min). Then, the probability for the travel time between customers i and j is given by the formula: $P(\tau_{ij} = m) = P(\lceil D_{ij}\delta/100 \rceil = m)$.

3.2. Dynamic programming algorithm

For each vehicle, we have a route including both mandatory and optional customers. So far we assumed that travel and service times were either minimal or maximal. In this last stage, we take into account the stochasticity on travel and service times and we modify the planning in real time to face stochasticity. We assume that segments are sorted the following way: segment $p+1$ has for origin mandatory customer $o^{p+1} = d^p$ (mandatory customer corresponding to the destination of segment p). Each dynamic programming step corresponds to the end of service at a customer's location. At each

step, we have a list of unserviced optional customers who could possibly be served before the next mandatory customer. In this context, two choices are to be taken into account: either the vehicle goes directly to the next mandatory customer or it visits the optional customer in this list that maximizes the profit. Let Γ_j be the lateness penalty and l_j the end of the time window at mandatory customer j. In this phase, we consider real travel and service times, noted respectively τ_{ij} and σ_i (with $\underline{\tau}_{ij} \leqslant \tau_{ij} \leqslant \overline{\tau}_{ij}$ and $\underline{\sigma}_i \leqslant \sigma_i \leqslant \overline{\sigma}_i$). At step k, we finish servicing customer v_k at time t_k. Let $\overline{V}^p$ be the sorted list of optional customers associated with segment p and $\overline{V}_k^p$ the list of optional customers of segment p located after v_k (see Fig. 2).

Using these notations, we propose two different dynamic programming methods. In the first algorithm, we consider only one segment p whereas we consider the remaining route in the second one.

- *Method considering one segment* (*OS*): In this method, when a vehicle finishes servicing customer v_k at time t_k with $\overline{V}_k^p$ the remaining customers to be visited before the next mandatory customer, there are two possibilities. Either it goes directly to the next mandatory customer d^p, earning thus a profit corresponding to the expected profit at d^p. Or it visits the optional customer $\overline{v}$ from $\overline{V}_k^p$ maximizing the expected profit (profit for visiting customer $\overline{v}$ + expected profit at $\overline{v}$). Moreover, in this method, the revenue associated with mandatory customer d^p is the opposite of the penalty cost for arriving late at this customer. The revenue function for the first method can thus be stated as follows:

$$f(v_k, t_k, \overline{V}_k^p) = \max\Big(E[f(d^p, t_k + \tau_{v_k d^p}, \varnothing)], \max_{\overline{v} \in \overline{V}_k^p}(p_{\overline{v}} + E[f(\overline{v}, t_k + \tau_{v_k \overline{v}} + \sigma_{\overline{v}}, \overline{V}_k^p \setminus \{\overline{v}\})])\Big)$$

with $f(d^p, t, \varnothing) = -\Gamma_{d^p} \max(t - l_{d^p}, 0)$

- *Method considering the whole route* (*WR*): In this method, the only change concerns the revenue associated with mandatory customer d^p at time t: in this revenue, we still have the opposite of the penalty cost for arriving late at this customer but we also have the expected profit associated with finishing the service at customer d^p at time $t + \sigma_{d^p}$. The revenue function for the second algorithm can thus be stated as follows:

$$f(v_k, t_k, \overline{V}_k^p) = \max\Big(E[\hat{f}(d^p, t_k + \tau_{v_k d^p})], \max_{\overline{v} \in \overline{V}_k^p}(p_{\overline{v}} + E[f(\overline{v}, t_k + \tau_{v_k \overline{v}} + \sigma_{\overline{v}}, \overline{V}_k^p \setminus \{\overline{v}\})])\Big)$$

with $\hat{f}(d^p, t) = -\Gamma_{d^p} \max(t - l_{d^p}, 0) + f(d^p, t + \sigma_{d^p}, V^{p+1})$

Solving the Bellman's equations, it can be shown that the optimal policies for both methods are threshold policies. A detailed proof is provided in [3]. For each customer, we obtain different time thresholds defining time intervals. With each interval is associated a decision (for example, if $t_1 \leqslant t \leqslant t_2$, go to customer c).

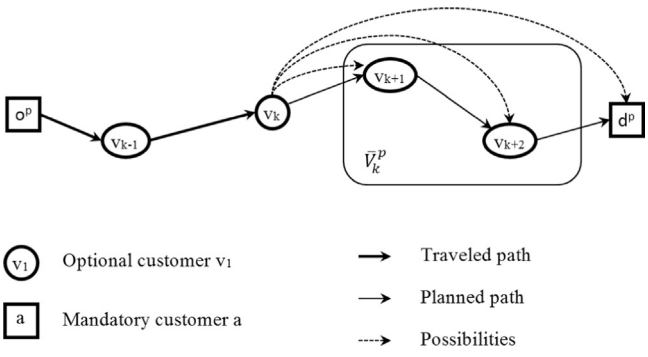

Fig. 2. Step k of the dynamic programming algorithm.

4. Computational results

4.1. Instances

To test the 2-stage method we just described, we chose to use instances proposed by Tricoire [1] since they correspond to the multi-period, multi-depot vehicle routing problem with time windows and priority within customers (i.e. the multi-depot and deterministic variant of our problem). In his instances, Tricoire [1] considers 3 vehicles and a planning horizon of 5 days. He proposes to take into account two types of customers: appointments with an associated time window of 2 or 4 h and a one-day validity period (mandatory customers), and postponable customers without time windows and with a validity period of some days (optional customers). From these instances, we extracted three sets of instances with 30, 40 and 50 customers. Moreover, as mentioned before, our method assumes having a sufficient number of mandatory customers in our instances (if this is not the case, it can be decided to make some optional customers become mandatory). Thus we chose to keep a number of mandatory customers between 5 and 9. Indeed, as mandatory customers correspond to repair operations, which may require long service times, it seems reasonable to consider serving no more than 3 mandatory customers per vehicle while ensuring a good service quality. We thus choose to keep between 2 and 3 mandatory customers per vehicle. Therefore, from each original instance, we extracted 5 instances in each set, each instance considering 3 technicians but having a different number of mandatory customers. From the instances called I (with $I \in \{C1\_1; C1\_2; C1\_3; C1\_4; C1\_5\}$) from Tricoire [1], we build instances called *I_N* with *N* being the number of mandatory customers ($N \in \{5; 6; 7; 8; 9\}$). For example, instance $C1\_1\_5$ is extracted from instance $C1\_1$ and contains 5 mandatory customers. We also modified the time windows of Tricoire [1] associated with mandatory customers in order to have only half-day (4 h) time windows. Thus we attributed to the first half of mandatory customers the morning and to the rest the afternoon.

4.2. Experimental context

The planning stage is solved exactly with Cplex 12.4 for instances with up to 40 customers. As mentioned above, for the skeleton design, we use the integer programming algorithm proposed in Cplex whereas, for the insertion of optional customers, we use a branch and cut algorithm by adding dynamically the subtour elimination cuts (12). For larger instances containing 50 customers, we use a branch and bound algorithm for the skeleton design, but we use the Lagrangian heuristic to insert optional customers in the skeleton. In both cases, at the end of the planning stage, we use repair and improvement algorithms to get routes with a sufficient feasibility probability. As for the execution stage, we use dynamic programming tools to get the optimal policy and its thresholds. We thus obtain, for each customer, a list of time thresholds and associated decisions corresponding to the optimal policy. We then proceed to 100 simulations per instance, each simulation consisting in: first, generating stochastic travel and service times and then, each time the service at a customer's location is completed, in comparing the current time to the thresholds and in meeting the optimal decision determined by the dynamic programming algorithm. Our experiments were conducted on a machine with 4 CPU, 2.8 Ghz, 30 Go of RAM. For setting our parameters, we based ourselves on the industrial context from Tricoire's [1] work. For instance, as he proposed to consider an average speed of 35 km/h in a suburban area, we chose a minimal speed of 20 km/h and a maximal speed of 50 km/h (to keep an average of

35 km/h). We also set the modal speed, speed that has the biggest probability to occur, to 40 km/h. Regarding service time, we considered that optional customers service times varied between 15 and 30 min whereas mandatory customers service times were between 30 and 60 min. For the insertion of optional customers, we assumed that the service of any optional customer yields a revenue $p_i = 100$. To set the parameter α for weighting travel time in the second step objective function, we made some experimentation. By choosing $\alpha = 1$, we ensure a hierarchical objective function, thus maximizing the profit for visiting optional customers, and, for the same profit, minimizing the total traveled distance. Finally, for the execution phase, we choose $\Gamma_{d^p} = 5000$ for any p.

4.3. Planning stage results

Before presenting the planning stage results, we set values for two parameters of the planning stage: L_{max} (maximal size of segments on which we generate reachability cuts instead of classical subtour elimination constraints) and S_{max}(the maximal size of subset elimination inequalities to be generated).

To adjust the L_{max} value, we made some experimentation on small instances containing 30 customers and 3 vehicles. Results were obtained by varying L_{max} from 0 to the time horizon ($L_{max} = 480$) and are summarized in Table 1. Unlike what we could expect, we observe in this table that the best computation times are obtained for $L_{max} = 480$ (time horizon), i.e. when we generate reachability cuts on all segments. We thus set $L_{max} = 480$ in further experiments.

For choosing S_{max}, we also made some experimentation summarized in Table 2. We can observe in this table that the best computation times are obtained for $S_{max} = 6$, i.e., when we generate subset elimination inequalities for subsets containing at most 6 customers. We thus choose $S_{max} = 6$.

In Table 3 we report the average computation times for our 2-stage method in which the insertion of optional customers consists in a single step (the one with pessimistic estimation). We limited the computation times to 2 h (i.e., 7200). If the problem could not be solved within this time, we replace the computation time by a "–". Many variants of the branch and cut algorithm are considered. The column headings are the following ones: # solved: number of instances solved optimally within 2 h; B&C: Branch and cut; P: Preprocessing on variables; H: Heuristic to generate an initial solution; RC: Reachability cuts; SEI: Subset elimination inequalities. In these tables, we can observe a significant decrease of the computation times thanks to the different speedup techniques. We are able to solve instances

Table 2
S_{max} adjustment (maximal size of subset elimination inequalities).

Number of mandatory customers	Instances	$S_{max} = 3$	$S_{max} = 4$	$S_{max} = 5$	$S_{max} = 6$	$S_{max} = 7$
	C1_1_5	4	4	4	4	4
	C1_2_5	8	8	8	12	17
5	C1_3_5	27	28	27	26	28
	C1_4_5	163	131	177	139	153
	C1_5_5	32	34	34	34	46
	C1_1_6	61	40	44	40	44
	C1_2_6	65	58	59	55	60
6	C1_3_6	124	76	68	80	75
	C1_4_6	69	46	45	53	59
	C1_5_6	211	105	102	102	102
	C1_1_7	368	145	144	144	145
	C1_2_7	16	16	16	15	15
7	C1_3_7	28	21	21	21	21
	C1_4_7	65	64	51	53	49
	C1_5_7	52	33	33	33	33
	C1_1_8	66	56	56	56	56
	C1_2_8	21	28	21	23	22
8	C1_3_8	17	13	12	13	12
	C1_4_8	63	46	58	60	70
	C1_5_8	22	23	25	17	16
	C1_1_9	31	23	23	23	23
	C1_2_9	21	26	25	26	27
9	C1_3_9	28	29	33	26	27
	C1_4_9	69	46	42	40	40
	C1_5_9	5	7	7	7	6

Table 1
L_{max} adjustment (maximal size of segments for reachability cuts).

Number of mandatory customers	Instances	$L_{max} = 0$	$L_{max} = 80$	$L_{max} = 160$	$L_{max} = 240$	$L_{max} = 320$	$L_{max} = 400$	$L_{max} = 480$
	C1_1_5	4	5	4	27	14	4	4
	C1_2_5	10	11	11	23	23	24	10
5	C1_3_5	39	40	40	46	46	46	29
	C1_4_5	1159	1208	1190	609	607	610	263
	C1_5_5	55	55	35	57	57	57	41
	C1_1_6	69	69	90	97	97	97	68
	C1_2_6	139	136	100	64	64	64	82
6	C1_3_6	267	259	263	206	165	166	148
	C1_4_6	120	120	120	82	83	83	76
	C1_5_6	1024	1033	1041	247	319	322	320
	C1_1_7	1219	1230	1427	565	559	561	559
	C1_2_7	26	27	23	16	19	19	18
7	C1_3_7	75	75	75	63	47	47	47
	C1_4_7	96	97	77	92	92	92	47
	C1_5_7	54	54	54	36	37	37	36
	C1_1_8	315	317	316	115	117	117	117
	C1_2_8	25	25	29	25	62	18	18
8	C1_3_8	13	13	14	24	12	12	12
	C1_4_8	76	78	75	80	80	80	76
	C1_5_8	23	24	23	23	31	31	31
	C1_1_9	101	98	98	35	35	35	35
	C1_2_9	28	27	27	25	41	41	41
9	C1_3_9	28	28	28	41	41	24	24
	C1_4_9	134	132	132	110	110	80	80
	C1_5_9	7	8	8	11	12	6	6

Table 3
Planning stage: average computation times (in seconds).

# of customers	# of mandatory	# of instances	B&C		B&C, P		B&C, P, H		B&C, P, H, RC		B&C, P, H, RC, SEI	
			# solved	Avg. time	# solved	Avg. time	# solved	Avg. time	# solved	Avg. time	# solved	Avg. time
30	5	5	5	1813	5	237	5	253	5	68	5	43
	6	5	4	2797	5	288	5	324	5	137	5	66
	7	5	2	1118	5	408	5	294	5	138	5	53
	8	5	4	1689	5	109	5	90	5	50	5	34
	9	5	5	1613	5	69	5	60	5	37	5	24
40	5	5	0	–	0	–	0	–	1	3869	3	3277
	6	5	0	–	3	1952	3	1720	4	1601	4	962
	7	5	0	–	1	519	1	493	3	3736	4	1451
	8	5	0	–	1	556	2	3428	2	1196	5	1880
	9	5	0	–	3	2250	3	1988	3	1201	5	1053

Table 4
Planning stage: branch and cut versus Lagrangian relaxation.

# of customers	# of mandatory	# of instances	Branch and cut			Lagrangian relaxation			
			# solved	Avg. gap to LB (%)	Avg. time	Avg. gap to opt. (%)	Avg. gap to LB (%)	Avg. gap LB to opt. (%)	Avg. time
30	5	5	5	0.0	43	1.75	2.03	0.27	192
	6	5	5	0.0	66	3.14	3.32	0.18	262
	7	5	5	0.0	53	2.13	2.83	0.66	102
	8	5	5	0.0	34	3.13	3.32	0.19	119
	9	5	5	0.0	24	2.22	2.95	0.70	99
40	5	5	3	1.8	3277	1.98	2.64	0.46	1211
	6	5	4	2.0	962	1.99	3.09	0.52	1106
	7	5	4	0.8	1451	1.60	2.81	0.27	572
	8	5	5	0.0	1880	2.09	2.42	0.32	532
	9	5	5	0.0	1053	1.35	2.43	1.04	507
50	5	5	0	4.04	7200	–	1.44	–	1227
	6	5	1	4.99	5960	0.33	1.63	0.00	3064
	7	5	1	4.03	6820	0.88	2.57	0.07	2844
	8	5	0	7.50	7200	–	2.40	–	2323
	9	5	2	2.78	5365	0.54	2.13	0.46	1408

with 30 customers in less than 3 min and most of the instances with 40 customers in less than 2 h.

The results obtained comparing Lagrangian heuristic and branch and cut methods are summarized in Table 4. We choose $\beta = 0.9$, $\epsilon = 1$ and the subgradient method ends either when the gap falls under 2% or after 50 iterations. The column headings are the following ones: # solved: number of instances solved to optimality; Avg. gap to opt: average gap between the best upper bound $\overline{z}$ and the optimal value $z^*(\text{gap} = \overline{z} - z^*/z^*)$; Avg. gap to LB: average gap between $\overline{z}$ and the best lower bound $\underline{z}(\text{gap} = \overline{z} - \underline{z}/\underline{z})$; Avg. gap LB to opt: average gap between $\underline{z}$ and $z^*(\text{gap} = z^* - \underline{z}/z^*)$. In Table 4, we observe that the branch and cut algorithm computes the optimal solution on almost all instances containing up to 40 customers. On larger instances (with 50 customers), most of the instances cannot be solved to optimality within 2 h with the branch and cut algorithm. The gap between solutions obtained with this algorithm on instances with 50 customers and the lower bound is located between 3% and 7.5% whereas the Lagrangian heuristic provides better solutions (with a gap to lower bound smaller than 3%) within less than 1 h. For the remaining part of experiments, we thus choose the branch and cut method for instances with up to 40 customers and the Lagrangian heuristic for instances containing 50 customers.

In order to validate our method for inserting optional customers, we performed some computational experiments on the Team Orienteering Problem (TOP) instances proposed by Chao et al. [25], and we compared our results with those obtained with a branch and price algorithm by Boussier et al. [26]. Results for both methods are shown in Table 5. We can observe that the branch and price algorithm is more efficient that our algorithm on most instances. In the TOP instances, a single depot is considered. It generates symmetry we did not have to deal with in our multi-depot variant. In our branch and cut algorithm, in order to strengthen the model with respect to this symmetry, we add the following constraints: $\sum_{i \in O} x_i^p \leqslant \sum_{i \in O} x_i^{p+1}$, $\forall p \in P$. Our branch and cut algorithm solves 15 unsolved instances with the branch and price algorithm proposed by Boussier et al. [26]. Results obtained on these open instances are shown in Table 6. We can observe that, on small-size instances, we are able to solve all of them, even if the branch and price algorithm may remain faster.

4.4. Execution stage results

For the execution phase results, we experiment the two strategies mentioned before: considering one segment (OS) and considering the whole route (WR). During simulations, in order to get unpredictable service and travel times, we increase the maximal service time $\overline{\sigma}_i$ for mandatory customers (we will note this value $s_{\max}$) and we decrease the minimal speed $v_{\min}$. For instances with 30 and 40 customers, we used results obtained with branch and cut method for the insertion of optional customers (even if the result is not the optimal one), whereas we used results obtained with Lagrangian heuristic for instances with 50 customers. The results obtained after simulation (with variant $V1$, $F=0.8$ for the repair and improve algorithms) are summarized in Tables 7–9 giving the mean values, before and after simulations, for the number of unserved customers, the total traveled distance and the total lateness (Table 10). In Tables 7–9, we observe that, for a given strategy (OS or WR), when we decrease the minimal speed and

Table 5
Branch and cut compared to branch and price on team orienteering problem instances.

Instance type	Instance	# of instances	Branch and cut			Branch and price		
			# solved	Avg value	Avg CPU	# solved	Avg value	Avg CPU
p1	p1_2	18	18	140.8	13.3	15	116.0	127.5
	p1_3	18	18	111.1	68.6	18	111.1	2.1
	p1_4	18	18	84.2	105.4	18	84.2	0.1
p2	p2_2	11	11	190.5	0.8	11	190.5	0.1
	p2_3	11	11	136.4	0.5	11	136.4	0.1
	p2_4	11	11	94.5	0.4	11	94.5	0.0
p3	p3_2	20	20	496.0	15.4	12	357.5	283.4
	p3_3	20	20	411.5	288.5	18	375.0	98.5
	p3_4	20	19	324.7	601.3	20	336.5	0.9
p4	p4_2	20	4	382.5	2159.5	5	429.6	996.6
	p4_3	20	4	141.5	337.5	9	422.7	643.6
	p4_4	20	6	90.8	224.2	11	344.3	65.8
p5	p5_2	26	9	331.1	1064.1	11	275.5	470.2
	p5_3	26	9	120.0	285.7	16	369.1	161.7
	p5_4	26	10	100.0	82.0	23	476.2	103.7
p6	p6_2	14	6	190.0	115.8	9	385.3	157.1
	p6_3	14	7	40.3	12.9	13	404.8	683.0
	p6_4	14	10	36.6	76.6	14	255.0	0.6
p7	p7_2	20	7	217.3	1143.0	6	177.0	9.3
	p7_3	20	8	179.1	1012.4	9	213.3	379.6
	p7_4	20	8	118.0	569.4	12	240.2	71.0

Table 6
Open team orienteering problem instances solved to optimality.

Instance type	# of customers	Instance	Value	CPU (s)
p1	32	p1_2_p	250	13
		p1_2_q	265	16
		p1_2_r	280	49
p3	33	p3_2_l	590	14
		p3_2_m	620	20
		p3_2_n	660	23
		p3_2_o	690	22
		p3_2_p	720	17
		p3_2_q	760	28
		p3_2_r	790	46
		p3_2_s	800	21
		p3_3_s	720	778
		p3_3_t	760	2394
p5	66	p5_2_z	1680	2943
p7	102	p7_2_g	459	5870

increase the maximal service time at mandatory customers, the quality of the obtained solution decreases. Indeed, we get more unserved customers, more lateness and, as we serve less customers, the traveled distance decreases. When comparing results obtained before and after simulations, we observe that, with $v_{min} = 15$ km/h, we modify the planned routes less than with $v_{min} = 10$ km/h (with 3–5 customers becoming unserved instead of 5–8). Moreover, in Tables 7–9, we observe that the strategy considering only one segment (OS) is better than the strategy considering the whole route (WR) regarding the average number of unserved customers. On the other hand, we should prefer the whole route strategy for the traveled distance and the lateness. For the remaining part of our experiments, we use $v_{min} = 15$ km/h and $s_{max} = 90$ min.

During the insertion of optional customers (in the planning stage), we proceed to the improvement of the solution (see Section 2.2.3). Two strategies are considered: in variant V1, we proceed to a single

Table 7
Average number of unserved customers on instances with 30 customers.

Number of mandatory customers	A priori	$v_{min}=15$ km/h				$v_{min}=10$ km/h			
		$s_{max}=75$ min		$s_{max}=90$ min		$s_{max}=75$ min		$s_{max}=90$ min	
		WR	OS	WR	OS	WR	OS	WR	OS
5	1.6	5.48	3.35	5.82	3.72	7.37	5.50	7.70	5.88
6	3.0	8.04	7.64	8.48	8.09	9.70	9.30	10.11	9.72
7	2.8	5.86	5.21	6.47	5.84	7.46	6.90	8.10	7.57
8	2.8	5.38	5.46	6.24	6.32	7.43	7.53	8.19	8.29
9	2.8	5.72	5.73	6.65	6.65	7.73	7.72	8.64	8.64

Table 8
Average traveled distance on instances with 30 customers.

Number of mandatory customers	A priori	$v_{min}=15$ km/h				$v_{min}=10$ km/h			
		$s_{max}=75$ min		$s_{max}=90$ min		$s_{max}=75$ min		$s_{max}=90$ min	
		WR	OS	WR	OS	WR	OS	WR	OS
5	219	200	211	198	209	193	203	191	201
6	226	214	216	211	214	205	208	203	205
7	221	204	207	200	203	194	196	189	192
8	232	217	218	212	213	206	207	202	202
9	224	207	207	202	202	197	198	193	193

Table 9
Average lateness on instances with 30 customers.

Number of mandatory customers	$v_{min}=15$ km/h				$v_{min}=10$ km/h			
	$s_{max}=75$ min		$s_{max}=90$ min		$s_{max}=75$ min		$s_{max}=90$ min	
	WR	OS	WR	OS	WR	OS	WR	OS
5	1.45	1.50	2.03	2.06	14.81	16.82	16.05	18.12
6	40.09	40.10	43.07	43.07	87.20	87.23	95.07	95.17
7	1.82	1.84	2.68	2.76	20.97	21.57	24.20	24.88
8	2.91	2.05	4.76	3.48	28.55	28.21	34.50	32.97
9	2.38	2.43	4.82	4.79	25.80	27.06	34.17	35.10

Table 10
Impact of the improvement variant on the average number of unserved customers for instances with 30 customers.

Number of mandatory customers	Variant V1				Variant V2			
	$F=0.8$		$F=0.9$		$F=0.8$		$F=0.9$	
	WR	OS	WR	OS	WR	OS	WR	OS
5	5.82	3.72	5.98	3.93	4.03	3.53	5.33	3.36
6	8.48	8.09	5.53	5.54	8.44	8.00	5.59	5.55
7	6.47	5.84	9.24	8.61	6.41	5.65	5.52	5.51
8	6.24	6.32	7.57	6.17	6.86	6.23	7.55	6.13
9	6.65	6.65	7.08	6.42	6.59	6.59	7.03	6.36

improvement, whereas in variant V2, we improve the solution as long as we can. We made some experiments to compare both strategies and also to choose the value of the feasibility threshold F (Table 11).

We observe in Tables 8–10 that the variant V2, consisting in improving the solution as long as we can, produces better solutions than the variant V1. In fact, the number of unserved customers is lower for variant V2 (thus increasing sometimes the traveled distance)

Table 11
Impact of the improvement variant on the average traveled distance for instances with 30 customers.

Number of mandatory customers	Variant V1				Variant V2			
	$F=0.8$		$F=0.9$		$F=0.8$		$F=0.9$	
	WR	OS	WR	OS	WR	OS	WR	OS
5	198	209	199	210	206	210	199	211
6	211	214	212	212	212	214	212	212
7	200	203	201	204	204	207	209	209
8	212	213	195	204	209	211	195	204
9	202	202	197	202	202	202	197	201

Table 12
Impact of the improvement variant on the average lateness for instances with 30 customers.

Number of mandatory customers	Variant V1				Variant V2			
	$F=0.8$		$F=0.9$		$F=0.8$		$F=0.9$	
	WR	OS	WR	OS	WR	OS	WR	OS
5	2.03	2.06	2.01	2.05	2.03	2.06	1.99	2.03
6	43.07	43.07	3.09	3.10	43.07	43.12	3.09	3.15
7	2.68	2.76	2.68	2.75	2.71	2.79	2.70	2.77
8	4.76	3.48	3.14	3.24	3.51	3.58	3.18	3.24
9	4.82	4.79	4.65	4.65	4.92	4.89	4.62	4.63

whereas the lateness remains almost the same. Regarding the feasibility threshold, we observe that increasing F induces a decrease of the number of unserved customers and of the lateness (for variant V2) while the traveled distance does not increase too much (Table 12).

The results obtained on instances with 40 customers are summarized in Tables 13–15. In these tables, we observe the same behaviour between variant V1 and V2 (smaller number of unserved customers and increased traveled distances for V2, and lateness remaining almost the same). On the other hand, when F increases in variant V2, the number of unserved customer increases with lateness whereas traveled distance decreases. However, variations remain small in comparison to those on 30 customers. We thus choose variant V2 with $F=0.9$.

The results obtained for the variant *V2*, with $F=0.9$, on instances with 50 customers are summarized in Table 16. Like previously, we can observe in this table that the one segment strategy turns out to be more effective regarding the number of unserved customers whereas the average lateness and traveled distance remain almost the same, regardless of the strategy used.

5. Conclusions

In this paper, we presented a 2-stage method for solving a field service routing problem with time windows, multiple depots, priority within customers and stochastic travel and service times. The method consists in a planning stage followed by an execution stage. In the planning stage, we first design a skeleton of mandatory customers, and then we insert optional customers in this skeleton. In the execution stage, we use a dynamic programming algorithm to face stochastic travel and service times. We also proposed some speedup techniques to improve computation times. We experimented our method for inserting optional customers on TOP instances and solved 15 open TOP instances from the literature. We *also* tested our entire method on instances based on realistic data. In the experimental results, we could see that these techniques proved very effective on

Table 13
Impact of the improvement variant on the average number of unserved customers for instances with 40 customers.

Number of mandatory customers	Variant V1				Variant V2			
	$F=0.8$		$F=0.9$		$F=0.8$		$F=0.9$	
	WR	OS	WR	OS	WR	OS	WR	OS
5	10.91	10.13	10.87	10.09	9.74	9.73	9.66	9.66
6	13.49	12.73	12.96	12.96	12.54	12.53	12.69	12.68
7	13.84	13.11	13.28	13.28	13.00	13.02	13.28	13.28
8	13.43	13.42	13.32	13.32	13.40	13.38	13.21	13.22
9	14.93	14.41	14.24	14.33	13.94	14.02	14.51	14.00

Table 14
Impact of the improvement variant on the average traveled distance for instances with 40 customers.

Number of mandatory customers	Variant V1				Variant V2			
	$F=0.8$		$F=0.9$		$F=0.8$		$F=0.9$	
	WR	OS	WR	OS	WR	OS	WR	OS
5	189	191	189	191	196	196	193	194
6	200	202	202	202	205	204	204	204
7	191	194	198	198	195	196	198	198
8	197	198	198	198	198	198	198	198
9	188	189	193	191	192	192	189	191

Table 15
Impact of the improvement variant on the average lateness for instances with 40 customers.

Number of mandatory customers	Variant V1				Variant V2			
	$F=0.8$		$F=0.9$		$F=0.8$		$F=0.9$	
	WR	OS	WR	OS	WR	OS	WR	OS
5	1.82	1.85	1.79	1.80	1.80	1.85	1.77	1.81
6	2.59	2.62	2.65	2.65	2.61	2.64	2.65	2.65
7	2.52	2.54	2.41	2.45	2.52	2.54	2.41	2.45
8	3.63	3.67	3.62	3.61	3.59	3.69	3.60	3.61
9	6.69	5.61	6.71	5.60	6.71	5.57	6.75	5.59

Table 16
Average values after simulations for instances with 50 customers.

Number of mandatory	Avg # unserved		Avg traveled distance		Avg lateness	
	WR	OS	WR	OS	WR	OS
5	17.84	17.87	184	185	1.96	1.97
6	21.00	20.36	187	187	3.62	3.54
7	21.84	21.20	180	180	3.56	3.51
8	22.14	22.16	188	188	4.27	4.39
9	22.73	22.75	179	179	5.81	6.00

instances containing up to 50 customers and 3 vehicles. In the computational results, we could observe that the best solutions were obtained with variant V2 (improve the solution as long as we can during the insertion of optional customers) and setting the feasibility threshold $F=0.9$. We also noticed that the dynamic programming strategy considering only one segment turns out to be more effective regarding the number of unserved customers whereas the other strategy (considering the whole route) prevents from lateness and distance overcost. However, even if our method gave good results, it has a drawback: we have to consider a sufficient number of mandatory customers (otherwise, the skeleton design becomes useless). If

this is not the case, it would be possible to increase the number of mandatory customers by making some optional customers become mandatory. Nevertheless, it would be interesting, in the future, to group the two phases of the planning stage of this method into a single one. It would also be interesting to extend the method in order to deal with the multi-period problem, to see the impact of the dynamic programming strategy (whole route or one segment) on a multi-period horizon.

Acknowledgements

This research has been undertaken within the framework of the "THT-Tournées à Hautes Technologies" (Hich Technology Routing) scientific cooperation project number 62.114 between France and Quebec. This work was partially supported by the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the French Ministry of Higher Education and Research and the National Center for Scientific Research. This support is gratefully acknowledged. The authors are also grateful to the Natural Sciences and Engineering Research Council of Canada for supporting this research through its Discovery Grants program. The authors would like to thank the reviewers for improving the paper quality and Fabien Tricoire for providing us the experimental data.

References

[1] Tricoire F. Optimisation des tournées de véhicules et de personnels de maintenance : application à la distribution et au traitement des eaux (Ph.D. thesis). Ecole des Mines de Nantes, 2006.
[2] Tricoire F, Bostel N, Dejax P, Guez P. Exact and hybrid methods for the multiperiod field service routing problem. Cent Eur J Oper Res 2011:1–19.
[3] Binart S. Optimisation de tournées de service en temps réel (Ph.D. thesis). Université de Lille 1 and Ecole Polytechnique de Montréal, 2014.
[4] Bostel N, Dejax P, Guez. P. Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In: Golden BL, Raghavan S, Wasil EA, editors. Routing problem: latest advances and new challenges. US: Springer; 2008. p. 503–25.
[5] Grötschel M, Krumke SO, Rambau J, Torres, LM. Online-dispatching of automobile service units. In: Operations research proceedings 2002, Springer; 2003. p. 168–73.
[6] Kovacs AA, Parragh SN, Doerner KF, Hartl RF. Adaptive large neighborhood search for service technician routing and scheduling problems. J Sched 2012;15(5):579–600.
[7] Borenstein Y, Shah N, Tsang E, Dorne R, Alsheddy A, Voudouris C. On the partitioning of dynamic workforce scheduling problems. J Sched 2009;13:411–25.
[8] Delage E, Bostel N, Dejax P, Gendreau M. Re-optimization of technician tours in dynamic environments with stochastic service time. Research report, Ecole des Mines de Nantes, 2010.
[9] Cortés CE, Gendreau M, Leng D, Weintraub A. A simulation-based approach for fleet design in a technician dispatch problem with stochastic demand. J Oper Res Soc 2010;62:1510–23.
[10] Souyris S, Cortés CE, Ordónez F, Weintraub A. A robust optimization approach to dispatching technicians under stochastic service times. Optim Lett 2012:1–20 Online First.
[11] Desrochers M, Desrosiers J, Solomon M. A new optimization algorithm for the vehicle routing problem with time windows. Oper Res 1992;40:342–54.
[12] Moscato P. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech concurrent computation program, C3P Report, 826, 1989.
[13] Dugardin F, Dejax P, Tricoire F. Optimisation réactive de tournées de service en environnement dynamique. Research report, Ecole des Mines de Nantes, 2006.
[14] Ando N, Taniguchi E. Travel time reliability in vehicle routing and scheduling with time windows. Netw Spat Econ 2006;6:293–311.
[15] Russell RA, Urban TL. Vehicle routing with soft time windows and Erlang travel times. J Oper Res Soc 2007;59:1220–8.
[16] Jie G. Model and algorithm of vehicle routing problem with time windows in stochastic traffic network. In: 2010 international conference on logistics systems and intelligent management, vol. 2, IEEE; 2010. p. 848–51.
[17] Tas D, Dellaert N, Woensel TV, Kok TD. Vehicle routing problem with stochastic travel times including soft time windows and service costs. Comput Oper Res 2013;40:214–24.
[18] Flatberg T, Hasle G, Kloster O, Nilssen EJ, Riise A. Dynamic and stochastic vehicle routing in practice. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I, editors. Dynamic fleet management. US: Springer; 2007. p. 41–63 [chapter 3].
[19] Lei H, Laporte G, Guo B. A generalized variable neighborhood search heuristic for the capacitated vehicle routing problem with stochastic service times. Top, Online Fir, 2011.
[20] Wang X, Regan AC. Assignment models for local truckload trucking problems with stochastic service times and time window constraints. Transp. Netw. Model. 2001;1771:61–8.
[21] Li X, Tian P, Leung SCH. Vehicle routing problems with time windows and stochastic travel and service times: models and algorithm. Int J Prod Econ 2010;125:137–45.
[22] Zeimpekis V, Minis I, Mamassis K, Giaglis GM. Dynamic management of a delayed delivery vehicle in a city logistics environment. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I, editors. Dynamic fleet management. US: Springer; 2007. p. 197–217.
[23] Tsiligirides T. Heuristic methods applied to orienteering. J Oper Res Soc 1984;35:797–809.
[24] Lysgaard J. Reachability cuts for the vehicle routing problem with time windows. Eur J Oper Res 2006;175:210–23 [chapter 9].
[25] Chao I, Golden BL, Wasil EA, et al. The team orienteering problem. Eur J Oper Res 1996;88(3):464–74.
[26] Boussier S, Feillet D, Gendreau M. An exact algorithm for team orienteering problems. 4or 2007;5(3):211–30.