Heuristics for an oil delivery vehicle routing problem

Eric Prescott-Gagnon*, Guy Desaulniers*, Louis-Martin Rousseau†

*École Polytechnique de Montréal and GERAD

C.P. 6079, Succursale Centre-Ville, Montréal, Québec H3C 3AT, Canada

eric.prescott-gagnon@gerad.ca, guy.desaulniers@gerad.ca

†École Polytechnique de Montréal and CIRRELT

C.P. 6079, Succursale Centre-Ville, Montréal, Québec H3C 3AT, Canada

louis-martin.rousseau@cirrelt.ca

Abstract

Companies distributing heating oil typically solve vehicle routing problems on a daily basis. Their problems may involve various features such as a heterogeneous vehicle fleet, multiple depots, intra-route replenishments, time windows, driver shifts and optional customers. In this paper, we consider such a rich vehicle routing problem that arises in practice and develop three metaheuristics to address it, namely, a tabu search (TS) algorithm, a large neighborhood search (LNS) heuristic based on this TS heuristic and another LNS heuristic based on a column generation (CG) heuristic. Computational results obtained on instances derived from a real-world dataset indicate that the LNS methods outperform the TS heuristic. Furthermore, the LNS method based on CG tends to produce better quality results than the TS-based LNS heuristic, especially when sufficient computational time is available.

Keywords: Oil delivery, rich vehicle routing, large neighborhood search, tabu search, column generation.

1 Introduction

This paper addresses a real-life application arising in the heating oil (or propane) distribution industry. In this application, an oil distributor supplies a set of customers with a single oil product that is stored at each customer in an oil tank of a known capacity. For most of these customers (called the VMI customers, for vendor-managed inventory), the oil inventory is managed by the distributor that must ensure, as much as possible, no oil shortages. The other customers are called *spot* customers because they can request a delivery at any

time. When such a request is accepted by the distributor, it must be fulfilled within a predetermined time limit (for instance, within 24 or 48 hours). Without spot customers, the problem corresponds to an inventory routing problem (IRP) in which the customers to service every day and the oil quantity they require must be determined along with the vehicle delivery routes. In practice, the size of such an IRP makes it intractable. Indeed, a real-life instance (for a region) can involve between 3 to 20 delivery vehicles and between 4,000 and 30,000 customers, each vehicle visiting between 30 and 50 customers per day, and each customer being serviced approximately at every 40 days during winter. Furthermore, the future customer demands are difficult to forecast because they highly depend on the forthcoming temperatures. Consequently, the industry considers this problem as a dynamic problem (every day, spot customers are revealed and VMI customer demands are updated) that must be tackled as a sequence of one-day problems.

Nowadays, the distributors rely on sophisticated forecasting systems (based on the historical consumption of each customer and past daily temperatures) to provide, at a given day of the year, a good estimate of the oil remaining in the tank of every of its VMI customers. Based on these estimates, a distributor can determine which of its VMI customers must receive a delivery on the next day, called the *day of operation*. Typically, a customer qualifies for a delivery if the estimate of oil remaining in its tank is about to fall below a safety stock level (say below 20% of the volume of its tank on the next day). The VMI and spot customers that must receive a delivery on the next day are called the *mandatory* customers. Those that will become mandatory in the following few days are called the *optional* customers. If these customers are located nearby some of the mandatory customers, it might be profitable to visit them as well even if their estimated oil inventory is not considered low enough for triggering a delivery request. To evaluate the profitability of visiting an optional customer, one must thus approximate the future refill cost (detour) that would be incurred if it was visited on a subsequent day.

Note that servicing an optional customer before the day it will become mandatory increases the frequency of its visits and, thus, its average servicing cost in the long term. To take this into account, one can consider an objective function that minimizes a weighted sum of the traveling distance and the expected inventory levels at the end of the day of operation. This can be useful when the optional customers can be serviced way ahead of time. In the application of interest, the optional customers are, typically, those that will become mandatory one or two days after the day of operation. In this case, service frequencies cannot increase by more than 1 or 2% and, thus, considering end inventory levels does not seem essential.

The one-day problem considered in this paper can be briefly stated as follows. Given a set of mandatory customers, a set of optional customers with their estimated future detours, find vehicle routes that visit all mandatory customers and possibly optional customers such that the total traveled distance minus the sum of the saved detours of the visited optional customers is minimized. This vehicle routing problem (VRP) includes several additional features such as customer time windows, multiple vehicle depots, a heterogeneous fleet of vehicles, driver shifts, and intra-route vehicle replenishments that will be detailed later. We call this rich VRP the *oil delivery vehicle routing problem* (ODVRP).

The classical VRP and several of its variants have been widely studied (Cordeau *et al.*, 2007, Golden *et al.*, 2008). As surveyed in Dror (2005), several papers have addressed oil and propane distribution. The closest works to ours have been realized by Dror *et al.* (1985) and Dror and Ball (1987) who modeled a propane delivery problem as an IRP defined over a short horizon (such that no customers are serviced twice in the horizon) and involving mandatory and optional customers. In this problem, the mandatory customers must be serviced once before a deadline in the horizon. The objective function includes traveling costs and future costs that favor servicing the mandatory customers as late as possible to avoid too frequent deliveries and the optional customers within the horizon to avoid a costly detour in a near future. These authors develop a two-stage solution method. In the first stage, the customers are assigned to delivery days (optional customers might remain unassigned) using linear programming and a rounding procedure. In the second stage, a VRP is solved for each day using a construction heuristic. This approach has not been designed to handle all the complexifying features (spot customers, time windows, driver shifts, intra-route replenishments, etc.) that need to be considered in the ODVRP.

In the following two decades, the research has focused on IRP with stochastic demands given that the future demands highly depend on the forthcoming temperatures. To deal with these problems, several authors (Kleywegt *et al.*, 2002, and Adelman, 2004, among others) developed Markov decision process models and corresponding approximate solution methods. Such solution approaches yield much higher computational times than the deterministic ones, which makes them difficult to use in practice for large instances involving complex features. The interested reader can consult the survey papers of Moin and Salhi (2007) and Bertazzi *et al.* (2008) for additional references on the IRP.

When considering a sequence of ODVRP to solve over a short horizon as a single problem, one can observe similarities between this problem and the multi-period VRP that consists of determining vehicle routes for each period such that each customer is serviced once within a predefined subset of the periods. Bostel *et al.* (2008) proposed a metaheuristic and

a column generation method for this problem that does not consider optional customers, spot customers, and intermediate vehicle replenishments. Intermediate replenishments were taken into account in different VRP variants by Angelelli and Speranza (2002), Tarantilis *et al.* (2008) and Crevier *et al.* (2007) using different methodologies.

The goal of this paper is to address a problem that matches with the current industry practice (that is, it corresponds to one problem of a sequence of one-day problems) and to propose solution methods that can solve real-life instances in relatively fast computational times. Our main contribution is thus to develop different heuristics for solving the ODVRP and compare their performances. We introduce three metaheuristics: a tabu search (TS) heuristic and two large neighborhood search (LNS) methods that rely, respectively, on the TS heuristic and on a CG heuristic for exploring the neighborhoods. The reasons for favoring these methods are as follows. TS is well known for producing in fast computational times good quality solutions to various complex VRP. In the last decade, LNS algorithms have also proven to be successful for several types of VRP. In particular, LNS combined with CG was used by Prescott-Gagnon *et al.* (2009, 2010) for the VRP with time windows (VRPTW) and the VRPTW with driver rules.

The paper is organized as follows. The ODVRP is formally stated in Section 2. How to compute the estimated detours of the optional customers is discussed in Section 3. The three proposed metaheuristics are described in Sections 4 to 6. Computational results comparing these three methods on instances derived from a real-life dataset are reported in Section 7. Finally, conclusions are drawn in Section 8.

2 Problem statement

Given a heterogeneous fleet of vehicles housed in a set of depots $\mathcal{K}$, a set of customers $\mathcal{N}$ divided into a subset of mandatory customers $\mathcal{N}_m$ and a subset of optional customers $\mathcal{N}_o$, the ODVRP consists of determining feasible routes to deliver one type of heating oil to a subset of the customers in $\mathcal{N}$ on a specific day of operation T. Every mandatory customer in $\mathcal{N}_m$ must be visited exactly once, whereas every optional customer in $\mathcal{N}_o$ must be visited at most once. When an optional customer $i \in \mathcal{N}_o$ is visited on day T, it incurs a bonus β_i corresponding to an approximation of the traveled distance saved for not visiting it in one of the subsequent days. The procedure that we use for computing these bonuses is exposed in the next section. The objective of the ODVRP is minimizing the total distance traveled by the vehicles (which is usually proportional to the total traveling cost) minus the bonuses of the visited optional customers. The following constraints and features must be taken into

account.

When planning the routes for day T, the quantity of oil q_i to be delivered to a customer $i \in \mathcal{N}$ is given by the difference between the capacity of the customer's tank and the estimated oil inventory in that tank, that is, the distributor uses an order-up-to level policy. As it is the case in general, we assume that q_i is relatively small compared to the capacity of any vehicle and, therefore, a single delivery can always fulfill a customer demand. Certain customers (mostly commercial customers) are subject to delivery time restrictions that are imposed, for instance, by the inaccessibility of their oil tank outside their business opening hours. Such a restriction for a customer i is modeled as a time window $[a_i, b_i]$ that indicates the admissible start of service times at this customer. When no time restrictions apply to a customer, its time window is set as the whole day T. Note that a driver can arrive before time a_i at customer i and wait until a_i before starting service. Every customer $i \in \mathcal{N}$ also has an associated service time s_i that can depend on the quantity delivered.

Let $\mathcal{E}$ be the set of drivers available on day T. Each driver $e \in \mathcal{E}$ is assigned to a depot $k_e \in \mathcal{K}$ (which can be his home in a rural area), a working shift defined by a time interval $[a_e, b_e]$ and a pre-assigned vehicle v_e of given capacity Q_e that must be picked up at the driver's depot and returned there. At most one route can be assigned to each driver $e \in \mathcal{E}$ and this route must start and end at depot k_e, must not exceed capacity Q_e (bearing the replenishment possibilities described below) and its time span must be included in shift $[a_e, b_e]$. Note that, because the vehicles are pre-assigned to the drivers, there is no need to consider vehicle availability constraints.

Each day and often in a middle of a shift, the vehicles need to be replenished. Replenishments must be performed at a replenishment station, which can correspond to a depot or to an arbitrary location. Denote by $\mathcal{F}$ the set of replenishment stations and by s_f^e the time required at station $f \in \mathcal{F}$ to replenish vehicle v_e of driver $e \in \mathcal{E}$. In general, when the vehicles start from a depot harboring a replenishment station, the vehicles are fully replenished before the beginning of the driver shifts. In other cases, they might be partially loaded or even empty. For simplicity reasons, we assume in this paper that they are fully loaded.

Let $\mathcal{L} = \mathcal{K} \cup \mathcal{F} \cup \mathcal{N}$ be the set of all locations (depots, replenishment stations, and customers). With every pair of locations $(i, j) \in \mathcal{L}^2$ that can be visited consecutively, a travel time t_{ij} and a travel distance d_{ij} are associated. The travel time t_{ij} is actually equal to the time required to go from location i to location j plus the service time, if any, at location i. The travel distance and the travel time are clearly related but the latter often depends on the types of road traveled (street, boulevard, highway, etc.) and the expected congestion, or simply on the expected average speed.

To summarize, a route for a driver e starts from depot k_e, visits a sequence of customers, possibly replenishing from time to time, before returning to the same depot. This route is feasible if it respects the driver's shift $[a_e, b_e]$ and the time window $[a_i, b_i]$ of each visited customer i and if the total amount of oil delivered between two consecutive replenishments (including those performed before the start and after the end of the route) does not exceed the vehicle capacity Q_e. The total distance traveled along this route is given by the sum of the travel distances d_{ij} between every pair (i, j) of consecutive locations visited along the route.

3 Optional customer bonuses

When the travel distances respect the triangle inequality, visiting an optional customer on the day of operation T can only increase the total traveled distance. However, it might be profitable to make a small detour to service an optional customer if it can be feasibly inserted into a route and if postponing its visit to a subsequent day yields a larger detour later on. The detour incurred by visiting a customer l between two other locations i and j is given by $d_{il} + d_{lj} - d_{ij}$ (i.e., the additional distance traveled to visit l between i and j). Consequently, when planning the routes of day T, one can take into account possible distance savings for the next few days by offsetting the detour incurred for servicing an optional customer l by its bonus β_l, which should estimate the detour incurred by a visit on a later day.

One intuitive way that can be used for computing the optional customer bonuses is to solve an ODVRP instance for each of the following few days, considering in each instance only the customers becoming mandatory on the corresponding day. The detours (bonuses) could then be computed directly from the solution obtained for each day. This approach has two main disadvantages. First, it is computationally expensive. Second, the computed detours do not take into account the fact that several optional customers can be serviced on day T. Indeed, if it turns out that the detour of a customer l is computed using (optional) customers i and j as adjacent customers, then this detour value is obsolete if customer i or j is serviced on day T.

A different approach was proposed by Dror and Ball (1987). Their goal was not to compute only a detour but rather to approximate the fraction of the total cost of a route that can be attributed to each customer served along this route. They consider that, for a given customer l, this fraction is the sum of two fractions, $\psi_1(l)$ and $\psi_2(l)$, that depend on the proximity of customer l to the others and on the distance between l and the (closest) depot, respectively. To compute $\psi_1(l)$, Dror and Ball define a neighborhood H_l around customer l,

compute the detour yielded by l in the shortest Hamiltonian tour visiting the customers in H_l, and divide this detour by the sum of the detours of all customers in this tour. Denoting by $\pi(i)$ and $\sigma(i)$ the predecessor and the successor of customer i in the tour, $\psi_1(l)$ is given by:

$$\psi_1(l) = \frac{d_{\pi(l),l} + d_{l,\sigma(l)} - d_{\pi(l),\sigma(l)}}{\sum_{i \in H_l} d_{\pi(i),i} + d_{i,\sigma(i)} - d_{\pi(i),\sigma(i)}}. \tag{1}$$

For $\psi_2(l)$, they compute the back-and-forth distance between customer l and the depot, and divide it by the sum of these back-and-forth distances for all customers in H_l:

$$\psi_2(l) = \frac{d_{ol} + d_{lo}}{\sum_{i \in H_l} d_{oi} + d_{io}}, \tag{2}$$

where o denotes the depot. The fraction of the cost attributed to customer l is equal to $\nu\psi_1(l) + (1-\nu)\psi_2(l)$, with $\nu \in [0,1]$ (Dror and Ball suggested $\nu = 0.5$), and the cost of servicing customer l (which corresponds to its bonus) is computed as

$$\beta_l = z(H_l)\left(\nu\psi_1(l) + (1-\nu)\psi_2(l)\right), \tag{3}$$

where $z(H_l)$ is the cost of the shortest Hamiltonian tour through the customers in H_l. Assuming relatively small neighborhoods, this approach is much faster than the previous one. Note that, in this case, the bonuses do not correspond to detours but rather to fractions of route costs. These bonuses are, thus, computed under the assumption that all customers will be served as optional, that is, the whole cost of a route will be saved.

Given that it is not possible to compute precisely the detour incurred by a customer l if it is not visited on day T, we develop a simple procedure for computing the optional customer bonuses. We propose to use a weighted average of the detours for all pairs of customers i and j that can yield the real detour. Weights are used for two reasons. First, it is difficult to determine which pairs should be considered and weights allow to consider them all, each with a relative importance. Second, the detour for a customer l yielded by customers i and j that are remotely located from l can be very small compared to the one produced by customers in the vicinity of customer l as illustrated in Figure 1. In this figure, customers i_1 and j_1 are closer to customer l than customers i_2 and j_2 are, but they yield a larger detour because i_2, j_2 and l are almost colinear. As a customer is, more often than not, serviced between customers that are in its vicinity, attributing small (resp. large) weights to detours generated by remotely (resp. closely) located customers provides a better estimate of the expected detour.

The proposed procedure is as follows. Let l be a customer in $\mathcal{N}_o$ and let $\mathcal{P}^l$ be the list of all pairs of customers i and j that can yield the real detour for l, that is, all pairs (i,j) such

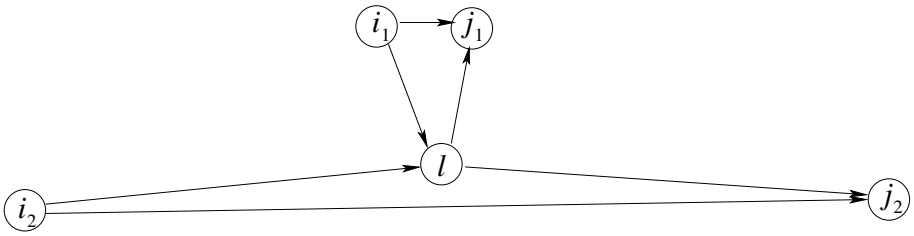

Figure 1: Two different detours for customer l

that $i, j \in \mathcal{N}_o$, $i \neq j$, $i \neq l$ and $j \neq l$. First, for each pair $(i, j) \in \mathcal{P}^l$, compute the sum of the distances $D^l_{ij} = d_{il} + d_{lj}$ between l and the two customers i and j. Second, sort the pairs $(i, j) \in \mathcal{P}^l$ in increasing order of D^l_{ij}. Third, compute the bonus β_l as follows:

$$\beta_l = w \sum_{p=1}^{|\mathcal{P}^l|} (1-w)^{p-1} (d_{i_p l} + d_{l j_p} - d_{i_p j_p}), \tag{4}$$

where $w \in (0, 1)$ is a parameter whose value provides the weight of the first detour, index p gives the order of the pair of customers in the ordered list $\mathcal{P}^l$, and (i_p, j_p) denotes the pair of customers of rank p. With this formula and $w = 10\%$, the weights of the detours yielded by the first four pairs are 10.00%, 9.00%, 8.10%, 7.29%. Note that the right-hand side of (4) is not (exactly) a weighted average of the detours because the series is not infinite. However, in practice, $|\mathcal{P}^l|$ is large enough to yield a sum of the weights $(w(1-w)^{p-1})$ very close to one. Note also that we have chosen to not consider the depots and the replenishment stations in this procedure because, in practice, they are not often visited in a route (approximately once every 15 customers).

In our computational experiments, we used formula (4) for computing the bonuses. In Section 7.2, we also report, for comparison purposes, computational results obtained when the bonuses are computed with formula (3).

4 Tabu search heuristic

To solve the ODVRP, we first introduce a TS algorithm. TS (see Glover and Laguna, 1997) is a metaheuristic that has been successful at solving a wide variety of combinatorial optimization problems, including many VRP variants (Cordeau *et al.*, 2001). It is an iterative method that starts from an initial solution and applies local modifications (moves) to improve it. Possible moves can be defined by a set of operators and are generally quite simple. A neighbor is a solution that differs from a current solution by only one move, and at each iteration, the move creating the best neighbor is chosen even if the objective value deteriorates. To avoid

cycling, a memory of past moves, often called the tabu list, is kept in order to forbid recent moves to be reversed for a number of iterations. This allows the search to escape from local minima.

Algorithm 1 presents the pseudo-code of a generic tabu procedure when there are multiple move types. It requires an initial solution Sol_0 and a parameter indicating the total number of TS iterations $TotTsIter$ to perform. The process starts by initializing the tabu list and an iteration counter i. Every possible non tabu move of every move type is evaluated on the current solution Sol_i. This evaluation computes the cost difference between the cost of the solution that would result from this move and the cost of Sol_i. This cost difference is stored in Move.Cost. The move yielding the best cost is then applied to create a new solution Sol_{i+1} and added afterwards to the tabu list. Certain moves in the tabu list may also be removed from this list to limit the number of tabu moves. The process repeats with the new solution until reaching $TotTsIter$ iterations.

The length of the tabu list is defined by a parameter L_{tabu} but has a random aspect. When a move is applied to a given element of the problem, this element becomes fixed for a number of iterations selected in $L_{tabu} \pm 5\%$. For instance, if a move is to insert an unserved customer, no other moves will be applied to this customer for the selected number of iterations. An aspiration criterion is however applied (but not described in Algorithm 1): when a move involving a tabu element generates a new overall best solution, it is accepted regardless of the tabu consideration.

The tabu search method that we propose for the ODVRP uses multiple move types to take into account the different aspects of the problem. They are:

- Remove a (mandatory or optional) customer from a route;
- Insert an unserved customer into a route;
- Exchange a customer from one route to another;
- Insert a visit to a replenishment station;
- Remove a visit to a replenishment station;
- Change the location of a replenishment;
- Exchange routes between two driver shifts.

The search is allowed to visit solutions that are infeasible because not all mandatory customers are visited, some time windows or driver shifts are violated, or vehicle capacity is ex-

Algorithm 1 Generic tabu search algorithm

Require: Initial solution Sol_0
Require: Total number of iterations $TotTsIter$
  BestSol $\leftarrow \text{Sol}_0$
  Create an empty tabu list
  $i \leftarrow 0$
  repeat
    BestCost $\leftarrow \infty$
    for all MoveType $\in$ MOVETYPES **do**
      for all Move $\in$ MoveType.MOVES **do**
        if Move.Cost $<$ BestCost and Move is not tabu **then**
          BestMove $\leftarrow$ Move
          BestMoveType $\leftarrow$ MoveType
          BestCost $\leftarrow$ Move.Cost
    $\text{Sol}_{i+1} \leftarrow$ BestMoveType.ApplyMove(BestMove, Sol_i)
    Sol_{i+1}.Cost = Sol_i.Cost + BestCost
    Update tabu list (add BestMove and remove, if any, past moves becoming non tabu)
    if Sol_{i+1}.Cost $\leq$ BestSol.Cost *and* Sol_{i+1} is feasible **then**
      BestSol $\leftarrow \text{Sol}_{i+1}$
    $i \leftarrow i + 1$
  until $i = TotTsIter$
  Return BestSol

ceeded. For a given solution x, its cost is given by $z(x) = d(x) - \beta(x) + \zeta q(x) + \alpha t(x) + \gamma m(x)$, where $d(x)$ is the total distance traveled, $\beta(x)$ is the sum of the bonuses of the visited optional customers, $q(x)$ and $t(x)$ are the total capacity and time window violations, and $m(x)$ is the total number of unserved mandatory customers. The computation of $q(x)$ and $t(x)$ is performed as suggested by Nagata *et al.* (2010). $q(x)$ is equal to the sum of the exceeding demands between two replenishments, while $t(x)$ is equal to the sum of the violations of the time window at each visited customer assuming (to avoid a cascading effect) that the time is reset to the time window lower bound when there is a violation (see Nagata *et al.*, 2010, for more details). The values of the parameters α, ζ and γ are adjusted dynamically throughout the solution process in order to obtain a *strategic oscillation* between feasible and infeasible solutions (see Glover and Laguna, 1997). They are all equal at the beginning of the algorithm and may change at every I^{adj} iterations. Each parameter value is adjusted separately considering the history of the solutions visited in the last I^{hst} iterations. The value of α is multiplied by 2 if the number of solutions in the history that are time-window infeasible exceeds ρ^{max} and divided by 2 if it is less than ρ^{min}, where ρ^{max} and ρ^{min} are predetermined parameters. The same adjustment procedure applies for parameters ζ and γ. Keeping a relatively small number of infeasible solutions with respect to each criterion ensures that the overall number of infeasible solutions is reasonable. To achieve this in our

computational experiments, we used the following parameter values: $I^{adj} = 20$, $I^{hst} = 100$, $\rho^{max} = 25$, and $\rho^{min} = 15$.

The initial solution Sol_0 is found using a greedy heuristic. This heuristic computes sequentially a route for each driver in an arbitrary order. Such a route is build greedily by selecting the next customer to service as the one that can be serviced the earliest among the unserved mandatory customers as long as it is feasible with respect to the driver's shift (a feasible return to the depot must be possible), the customer's time window, and the vehicle capacity. When no customers can be serviced due to vehicle capacity, then the route is extended towards the closest replenishment station before adding further customers (if time permits). This heuristic does not guarantee that all mandatory customers can be serviced. However, to favor a wide coverage of these customers, no optional customers are included in the initial routes.

5 Large neighborhood search heuristic based on tabu search

Starting from an initial solution, an LNS method (Shaw, 1998) is an iterative method that alternately remove elements from the current solution (destruction step) and reinsert them (reconstruction step) in order to find, hopefully, a better solution. A neighborhood at a given iteration is the set of all solutions that contain the undestroyed parts of the current solution. Because the size of the neighborhood increases exponentially with the number of elements removed, it has the potential to change a large portion of the solution. The algorithm stops when a certain number of iterations has been performed.

In Algorithm 2, we present the pseudo-code of a generic LNS algorithm that relies on different operators in the destruction and the construction phases. It requires an initial solution Sol_0 and a parameter indicating the total number of LNS iterations $TotLnsIter$ to perform. It starts by initializing the iteration counter i. Then, it repeats the same steps: choose a destruction operator, destroy a part of the current solution Sol_i to yield a partial solution $\overline{\text{Sol}}_{i+1}$, choose a reconstruction operator, and reconstruct a complete solution Sol_{i+1}. If this solution is better than the best solution found so far, then it becomes the best solution found. The process repeats with solution Sol_{i+1} (even if it was not better than the previous solution Sol_i) until reaching $TotLnsIter$ iterations.

In this section, we propose a first LNS heuristic for the ODVRP that uses TS in the construction phase. This heuristic, denoted LNS-TS, starts from the initial solution computed by the

Algorithm 2 Generic large neighborhood search algorithm

Require: Initial solution Sol_0
Require: Total number of iterations $TotLnsIter$
  $i \leftarrow 0$
  BestSol $\leftarrow \text{Sol}_0$
  repeat
    OpDestroy $\leftarrow$ ChooseDestructionOperator()
    $\overline{\text{Sol}}_{i+1} \leftarrow$ Destroy(Sol_i,OpDestroy)
    OpReconstruct $\leftarrow$ ChooseReconstructionOperator()
    $\text{Sol}_{i+1} \leftarrow$ Reconstruct($\overline{\text{Sol}}_{i+1}$,OpReconstruct)
    if Sol_{i+1}.cost $<$ BestSol.cost **then**
      BestSol $\leftarrow \text{Sol}_{i+1}$
    $i \leftarrow i + 1$
  until $i = TotLnsIter$
  Return BestSol

greedy algorithm described in Section 4. The destruction procedure selects a fixed number of customers to remove from their current route using a destruction operator. The other customers are said to be fixed for this LNS iteration, that is, they must remain in their route except if such a customer is serviced between two selected (thus freed) customers in the current solution. In this case, this customer is also freed. The sequence of visits to fixed customers must remain the same. Furthermore, no customers can be inserted between two fixed customers that are adjacent in a current route. Note that visits to replenishment stations or depots are never fixed. Also, the destruction operator may select customers that were not served in the current solution. Unserved customers that are not selected are fixed in their unserved state and, therefore, cannot be inserted in any route in this iteration.

At each LNS iteration, one of four destruction operators is chosen by a roulette-wheel procedure (Pisinger and Ropke, 2007) which favors the selection of the most efficient operators according to the improvements they yielded in the past iterations (for complete details, see Prescott-Gagnon *et al.*, 2009). The four operators are inspired from those of Prescott-Gagnon *et al.* (2009) and adapted to the ODVRP. They are as follows.

Time operator: For each customer, the time interval in which the customer can be visited without yielding an infeasible route is first computed. For an unserved customer, this interval is simply its time window or the whole planning horizon (day T) if there is no time window. Then, a time τ is selected randomly within the planning horizon. Finally, customers are selected randomly favoring those with time intervals containing τ or close to τ.

Proximity operator (Shaw, 1998): A seed customer is first selected randomly. The other customers are selected randomly, favoring those located near the location of the seed cus-

tomer.

Longest detour operator: Customers are selected randomly, favoring those who generate a longer detour in the current solution. The detour associated with an unserved customer i corresponds to its bonus β_i.

Smart operator (Rousseau *et al.*, 2002): A first seed customer is selected. If it is serviced in the current solution, some of its adjacent customers in its route are also selected. A next seed customer is selected randomly, favoring those located near the location of the first seed customer. Once again, if this customer belongs to a route, some customers adjacent to it are also selected. The process is repeated until selecting the right number of customers. Note that once a customer and its adjacent customers are selected, no more customers from the same route can be selected.

The TS method of Section 4 is used in the reconstruction phase. At a given LNS iteration, the TS method starts from the incumbent solution and applies the moves presented in Section 4 but only to the parts of the solution that were freed in the destruction phase. The TS method is thus limited to the neighborhood defined for this LNS iteration.

After a fixed number of TS iterations, the TS heuristic returns the best feasible solution found to the LNS algorithm which then becomes the new incumbent solution. It may happen that the solution returned is the initial solution. This does not imply that the LNS algorithm will stall forever as the destruction phase will destroy different parts of the solution in the next iterations. Nevertheless, stalling was observed in preliminary tests. This led us to introduce a diversification strategy that is invoked when the TS method is unable to improve the current solution after a fixed number n_{same} of LNS iterations: instead of returning the best feasible solution found, the TS heuristic returns the last feasible solution visited. This strategy was devised in the hope of moving the search to a different feasible region of the search space instead of going back to the same region over again. Through the tuning of parameter n_{same}, we can achieve a suitable trade-off between intensification and diversification. For our computational experiments, n_{same} was set to 5.

Note that the LNS framework can be viewed as a guiding procedure for the TS method, restricting the number of TS moves at each iteration and leading it to promising areas of the search space.

6 Large neighborhood search heuristic based on column generation

The second LNS heuristic that we propose is denoted LNS-CG and differs from the LNS-TS heuristic only by the reconstruction phase, where reconstruction is performed by a CG heuristic instead of a TS heuristic. CG (see Desrosiers and Lübbecke, 2005) is one of the leading methodology for solving exactly various constrained VRP and crew scheduling problems. As for all exact methods, it can easily be transformed into a heuristic. To tackle the ODVRP restricted to a neighborhood, we propose to adapt the column generation heuristic of Prescott-Gagnon *et al.* (2009) that was developed for the VRPTW and embedded into an LNS framework to produce very competitive results on well-known VRPTW benchmark instances. For reasons of clarity, we first describe a column generation heuristic for the whole ODVRP. Afterwards, we discuss how this heuristic can take into account the neighborhoods imposed by the LNS framework.

The ODVRP can be modeled as a set partitioning type problem where all the variables are associated with feasible routes. Let $\mathcal{R}^e$ be the set of all feasible routes for driver $e \in \mathcal{E}$. For each route $r \in \mathcal{R}^e$, denote by d_r the total distance traveled along r, β_r the total of the bonuses collected along r, and n_{ri} the number of times (0 or 1) that customer $i \in \mathcal{N}$ is serviced along r. Furthermore, define a binary variable θ_r^e that is equal to 1 if route r is assigned to driver e and 0 otherwise.

Using this notation, the ODVRP can be formulated as follows.

$$\text{Minimize} \qquad \sum_{e\in\mathcal{E}} \sum_{r\in\mathcal{R}^e} (d_r - \beta_r)\theta_r^e \tag{5}$$

$$\text{subject to:} \qquad \sum_{e\in\mathcal{E}} \sum_{r\in\mathcal{R}^e} n_{ri}\theta_r^e = 1, \qquad \forall\, i \in \mathcal{N}_m \tag{6}$$

$$\sum_{e\in\mathcal{E}} \sum_{r\in\mathcal{R}^e} n_{ri}\theta_r^e \leq 1, \qquad \forall\, i \in \mathcal{N}_o \tag{7}$$

$$\sum_{r\in\mathcal{R}^e} \theta_r^e \leq 1, \qquad \forall\, e \in \mathcal{E} \tag{8}$$

$$\theta_r^e \text{ binary}, \qquad \forall\, e \in \mathcal{E},\ r \in \mathcal{R}^e. \tag{9}$$

The objective function (5) aims at minimizing the total traveled distance minus the bonuses of the optional customers serviced. Set partitioning constraints (6) ensure that each mandatory customer is visited exactly once by a driver, whereas set packing constraints (7) impose a maximum of one visit to each optional customer. Constraints (8) specify that at most

one route can be assigned to each driver. Finally, (9) enforce binary requirements on the variables.

In practice, model (5)–(9) contains a huge number of variables. To overcome this difficulty, column generation is used to solve its linear relaxation, called the master problem in this context. This iterative method solves at each iteration a restricted master problem (RMP) and subproblems. The RMP is simply the master problem restricted to a subset of its variables. Solving it provides a primal and a dual solution. This dual solution is transferred to the subproblems that aim at finding negative reduced cost variables (columns). When such columns are found, they are added to the RMP before starting a new iteration. This iterative process is repeated until no negative reduced cost variables can be generated. In an exact algorithm, one must prove that there are no more negative reduced cost variables before stopping with an optimal master problem solution (the current RMP optimal solution). In our context, the subproblems are solved heuristically which means that the last RMP solution might not be optimal for the master problem.

The subproblems correspond to elementary shortest path problems with resource constraints (ESPPRC), where resources (see Irnich and Desaulniers, 2005) are used to impose the capacity and time window constraints, and elementarity is not required for replenishment stations that can be visited multiple times along the same route. There is one subproblem per driver $e \in \mathcal{E}$. Its underlying network $\mathcal{G}^e = (\mathcal{V}^e, \mathcal{A}^e)$ can be defined as follows. The node set contains $|\mathcal{N}|+|\mathcal{F}|+2$ nodes: one node for each customer $i \in \mathcal{N}$; one node for each station $f \in \mathcal{F}$; and a pair of source node o_1^e and sink node o_2^e representing the driver's depot at the beginning and the end of the driver's shift, respectively. The arc set $\mathcal{A}^e$ contains: start arcs (o_1^e, j), $\forall\, j \in \mathcal{N}$; end arcs (i, o_2^e), $\forall\, i \in \mathcal{N}$; travel arcs (i, j), $\forall\, i, j \in \mathcal{N}$ such that customer j can be visited immediately after customer i in at least one feasible route (that is, if $a_i + t_{ij} \leq b_j$ and $q_i + q_j \leq Q_e$); to replenishment arcs (i, f), $\forall\, i \in \mathcal{N}$ and $f \in \mathcal{F}$; and from replenishment arcs (f, i), $\forall\, i \in \mathcal{N}$ and $f \in \mathcal{F}$. With each arc $(i, j) \in \mathcal{A}^e$ is associated its corresponding travel distance d_{ij}.

At a given CG iteration, the subproblem for driver e, $e \in \mathcal{E}$, searches for the least reduced cost variable θ_r^e, $r \in \mathcal{R}^e$. The reduced cost $\bar{c}_r^e$ of such a variable is given by

$$\bar{c}_r^e = d_r - \beta_r - \lambda_e - \sum_{i \in \mathcal{N}} n_{ri}\phi_i = \sum_{(i,j) \in r} d_{ij} - \sum_{(i,j) \in r \mid i \in \mathcal{N}_o} \beta_i - \lambda_e - \sum_{(i,j) \in r \mid i \in \mathcal{N}} \phi_i,$$

where ϕ_i, $i \in \mathcal{N}$, are the dual variables of the constraints (6) and (7), λ_e is the dual variable of the constraint (8) for driver e, and $(i, j) \in r$ means an arc $(i, j) \in \mathcal{A}^e$ belonging to the path in $\mathcal{G}^e$ representing route r. Consequently, in the ESPPRC subproblem for driver e, the

(reduced) cost $\bar{c}^e_{ij}$ of the arc $(i,j) \in \mathcal{A}^e$ is equal to

$$\bar{c}^e_{ij} = \begin{cases} d_{ij} - \lambda_e & \text{if } i = o^e_1 \\ d_{ij} - \phi_i & \text{if } i \in \mathcal{N}_m \\ d_{ij} - \beta_i - \phi_i & \text{if } i \in \mathcal{N}_o \\ d_{ij} & \text{otherwise} \end{cases}$$

to ensure that the sum of the reduced costs of the arcs of a path is equal to the reduce cost of the corresponding route.

The ESPPRC is usually solved by dynamic programming (see Irnich and Desaulniers, 2005) which can be highly time-consuming when the time windows are wide as it is the case in our experiments. Consequently, we propose to solve the ESPPRC subproblems using a tabu search algorithm similar to the one introduced by Desaulniers *et al.* (2008) for generating rapidly columns in an exact CG method for the VRPTW. At every CG iteration, all columns in the current optimal basis for the RMP (they all have a zero reduced cost) are considered as initial solutions for the tabu search algorithm which performs a fixed number of tabu search iterations for each initial solution. The tabu moves considered here are a subset of the ones defined for the tabu search method described in the previous section because individual routes are sought when solving a subproblem, not a whole set of routes as for the ODVRP. They are: insert or remove a customer from a route, insert or remove a replenishment station from a route, and change the driver assigned to a route. Note that this last operator allows to switch from one subproblem to another for the same initial solution. In this tabu search algorithm, a move can be accepted only if it yields a feasible solution (route).

To derive an integer solution, the CG algorithm is embedded into a rounding procedure. After solving a linear relaxation whose computed solution is fractional, the variable θ^e_r with the largest fractional value is fixed at 1, defining a new linear relaxation that is also solved by CG.

In the LNS-CG algorithm, this CG heuristic is limited to the neighborhood defined for the current LNS iteration. Sequences of fixed customers are treated as aggregated customers in the subproblems, yielding routes that always respect the fixed parts of the current solution. Note that artificial variables are added to the covering constraints (6) of the mandatory customers in the RMP. This allows to not cover all these customers bearing a high penalty. This is essential when starting the solution process with an initial solution Sol_0 that does not visit all mandatory customers.

As in Prescott-Gagnon *et al.* (2009), we keep in memory a pool of columns that contains all the routes generated throughout the whole LNS solution process. At the start of a new LNS iteration, all columns in this pool that respect the fixed parts of the current solution

are directly added to the RMP. This pool of columns thus acts as a long term memory mechanism.

7 Computational experiments

To test the proposed heuristics, ODVRP instances were derived from a database of an oil distribution company containing more than 4,000 customers. For each customer, the database provides its location, its tank size, its safety stock level, and a history of its last deliveries (delivered quantities and dates for up to 10 deliveries). To service these customers, the company relies on four drivers that are preassigned to vehicles of different capacities. These drivers operate out of two depots equipped with replenishment facilities. There are also three additional replenishment stations scattered through the serviced region, which comprises a rural part. The drivers have all the same shift. Distances and travel times between every pair of locations were computed using a geographical information system (GIS).

Using the previous data, two types of instances were devised. The first type (Section 7.1) considers a single day of operation, while the second (Section 7.2) considers a whole week of planning for which a sequence of ODVRP will be solved as in a rolling horizon procedure. Except when stated otherwise, the bonuses were computed using formula (4).

7.1 Single day

To create the daily test instances, we used the following methodology. First, three dates covered by the database and relatively far apart were selected. For each of these dates, an estimation of the volume of oil q_i consumed by each customer i since its last refueling was computed using linear interpolation through the history of deliveries. Customers were then sorted in decreasing order of the percentage of oil consumed with respect to their effective tank size (denoted U_i for customer i), which is equal to the tank size minus the safety stock level.

For each date, six instances involving the first 250 customers as the set of customers $\mathcal{N}$ were created. They differ by their subset of mandatory customers or their service and replenishment times. Three subsets of mandatory customers $\mathcal{N}_m$ are considered, containing the first 70, 85 and 100 customers. In each case, the remaining customers are optional customers. Two realistic settings are proposed for the delivery and replenishment times. In one setting, all service times s_i are fixed at 7 minutes and all replenishment times s_f^e at 10 minutes. In the other, they are all equal to 8 and 15 minutes, respectively. Table 1 summarizes the

	250 customers		500 customers		750 customers		Service	Replen.
ID	Mand.	Opt.	Mand.	Opt.	Mand.	Opt.	time (min)	time (min)
1	70	180	140	360	210	540	7	10
2	85	165	170	330	255	495	7	10
3	100	150	200	300	300	450	7	10
4	70	180	140	360	210	540	8	15
5	85	165	170	330	255	495	8	15
6	100	150	200	300	300	450	8	15

Table 1: Characteristics of the instances created for each date

characteristics of the six instances for each date. Time windows (as encountered in practice) were also imposed on 10% of the customers. Finally, following preliminary tests, the value of the parameter w used to compute the optional customer bonuses in formula (4) was set to 10%.

Larger instances involving the first 500 and the first 750 customers were also created for each date. Again, for each date and each instance size, six instances were devised. Compared to the 250-customer instances, the numbers of mandatory and optional customers are multiplied by 2 and 3 for the 500- and 750-customer instances, respectively. The two settings described above for the service and replenishment times are also considered (see Table 1). In order to simulate a company with a larger fleet, the available drivers and vehicles were duplicated once and twice, yielding 8 and 12 drivers, respectively. Because a larger company, typically, services more customers every day that have a demand q_i very close to their effective tank size U_i, the demands derived from the database were also adjusted as follows: for the 500-customer (resp. 750-customer) instances, every computed demand q_i was increased by $\frac{U_i - q_i}{3}$ (resp. $\frac{2(U_i - q_i)}{3}$).

Overall, there are 18 instances (3 dates, 3 subsets of mandatory customers, 2 settings for service and replenishment times) for each instance size (250, 500, and 750 customers). They were all solved using each of the three proposed heuristics, namely, TS, LNS-TS, and LNS-CG. For the TS heuristic, the total number of iterations ($TotTsIter$) was set to 800,000 and the tabu list length L_{tabu} was fixed at 60% of the number of customers in the instance. For the LNS-TS heuristic, a total of $TotLnsIter = 1600$ LNS iterations were performed in which 1000 TS iterations were executed, yielding a total of 1,600,000 TS iterations. The neighborhoods were defined by removing 110, 180 and 240 customers for the 250-, 500- and 750-customer instances, respectively. Parameter L_{tabu} was fixed at 60% of the number of customers removed. Finally, for the LNS-CG heuristic, $TotLnsIter$ was also set to 1600 but the number of customers removed in the destruction phase was fixed at 70 for all instance

sizes. Based on preliminary test results, we chose these parameter values so that each heuristic performed at its best given a targeted time limit (one hour for the 750-customer instances) that seems reasonable in practice. Because there is randomness in each proposed method, ten runs were executed for each instance with each method. All our experiments were conducted on an AMD Opteron processor clocked at 2.3GHz. The column generation heuristic in the LNS-CG heuristic was implemented using the Gencol library, version 4.5, and Cplex, verison 12.1, for solving the RMP.

The results of these experiments are reported in Table 2, which is divided horizontally into three parts, one for each instance size. Each part contains three rows, one for each method. The results in a row correspond to averages over the 18 instances solved ten times. In order, the columns indicate: the heuristic used, the best solution cost (i.e., total distance minus awarded bonuses) obtained over the ten runs, the average solution cost over these runs, the total computational time, the number of optional customers visited, and the average number of replenishments per route. From these results, we make the following observations. The TS heuristic is clearly outperformed by the other two heuristics: it generates worst-quality solutions in higher computational times. Embedding it into an LNS framework provides much better results. In particular, it allows intensifying the search in medium-size neighborhoods and executing twice the number of TS iterations in less computational times. On the other hand, the LNS-CG heuristic yields the best solutions over all instances, and, in less computational times for the 500- and 750-customer instances. Consequently, with its global view of a neighborhood, the CG heuristic seems to be a better tool for reconstruction. As particularly obvious for the largest instances, better costs (obtained from the LNS-CG solutions) can be achieved when visiting less optional customers. This suggests that the TS and LNS-TS algorithms insert too many optional customers into the routes which do not give them much leeway afterwards to change the current solution and retrieve feasibility easily. As expected, the average number of replenishments per route is proportional to the total number of (mandatory and optional) customers visited. This total number indicates that a driver visits on average over 30 customers per day, which corresponds to practice (especially when the serviced region includes a rural part as is the case here).

Figure 2 allows to compare the quality of the best solutions produced by the LNS-TS and the LNS-CG heuristics over the 1600 iterations for the 750-customer instance with service and replenishment times of 7 and 10 minutes, respectively. The small graph is an enlargement of the large graph that focuses on the first 100 iterations. The curves correspond to averages over 10 runs. As one can observe, both heuristics yield best solutions of similar quality in the first 100 iterations. After that, the LNS-CG heuristic clearly outperforms the LNS-TS heuristic

Heuristic	Best cost	Avg cost	Time (sec)	Nb opt. cust. visited	Avg no. replen. per route
250 customers					
TS	264.3	279.1	1070	57.5	1.73
LNS-TS	251.2	265.4	717	56.3	1.67
LNS-CG	243.8	256.7	792	53.5	1.62
500 customers					
TS	391.0	414.8	3739	136.5	1.76
LNS-TS	377.9	403.4	2043	131.4	1.62
LNS-CG	374.3	400.5	1532	92.3	1.38
750 customers					
TS	544.3	586.1	8296	215.2	1.73
LNS-TS	534.8	581.9	3341	200.1	1.56
LNS-CG	522.8	549.3	2888	115.8	1.28

Table 2: Results for the single-day instances

that struggles much more rapidly at finding new best solutions. Similar results are obtained for the other instances.

For the 250-customer instances, Table 3 presents a breakdown per instance ID of the results obtained by the LNS-CG heuristic. For each instance ID (see Table 1), the same statistics as above are given. They correspond to averages over three instances (one for each date considered) solved ten times. From these results, one can observe, on the one hand, that longer service and replenishment times (in instance IDs 4 to 6) yield, as expected, costlier solutions because the problems are more constrained and less optional customers can be serviced by the available drivers (reducing the average number of replenishments per route). On the other hand, augmenting the number of mandatory customers (that is, from instance 1 to instance 3, and from instance 4 to instance 6) also increases the solution cost for the same reasons and the average computational time because set partitioning constraints (6) for the mandatory customers are harder to satisfy than the set packing constraints (7) for the optional customers in a CG heuristic. Similar remarks can be made for the 500- and 750-customer instances.

In another series of experiments with the LNS-CG heuristic, we performed a sensitivity analysis on the number of customers removed in the destruction phase of each LNS iteration. We conducted these experiments only on the 250-customer instances. Again, $TotLnsIter$ was set to 1600 and ten runs were executed for each instance. The results are reported in Table 4. They show that increasing the size of the neighborhoods helps computing better-quality so-

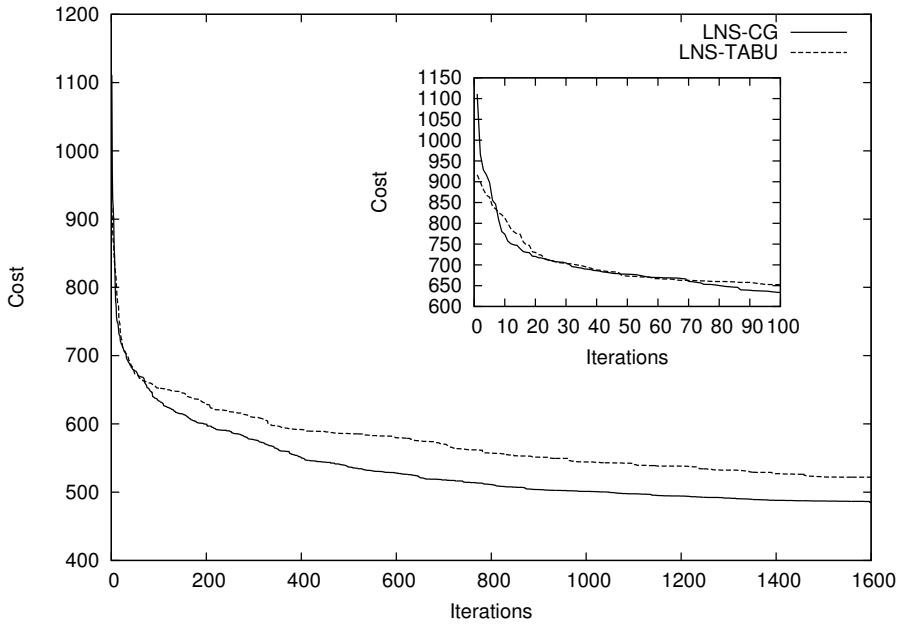

Figure 2: Average best solution cost over the iterations for a 750-customer instance

ID	Best cost	Avg cost	Time (sec.)	Nb opt. cust. visited	Avg no. replen. per route
1	204.2	209.9	703	73.1	1.66
2	241.2	255.2	798	60.6	1.68
3	269.6	287.1	896	46.7	1.73
4	209.2	219.3	689	61.1	1.54
5	252.2	265.7	778	47.3	1.57
6	286.1	303.0	885	32.0	1.56

Table 3: Breakdown of the LNS-CG results for the 250-customer instances

lutions and improving the robustness of the solution method. Indeed, the gap between the average cost and the best cost reduces as this size increases. On the other hand, the average computational time increases quite rapidly with the size of the neighborhoods.

7.2 Week planning

Presented in Section 3, the bonus β_i for an optional customer $i \in \mathcal{N}_o$ estimates the detour incurred by visiting this customer later than on the day of operation T. Consequently, if it is visited on day T, we consider that this detour is saved and can be subtracted from the objective function. These bonuses are thus incentives for servicing the optional customers

No. cust. removed	Best cost	Avg cost	Time (sec)	Nb opt. cust. visited	Avg no. replen. per route
50	248.2	262.9	681	52.7	1.65
70	243.8	256.7	792	53.5	1.62
90	242.0	252.5	1083	54.5	1.64
110	241.8	250.5	1409	54.0	1.64
130	240.1	246.9	1958	54.0	1.63

Table 4: LNS-CG results for different numbers of customers removed (250-customer instances)

on day T and minimizing the total distance traveled over a medium-term horizon if they estimate futur detours adequately. To assess their impact on the medium-term solution quality, we propose to consider a one-week horizon and solve a sequence of ODVRP, one for each day of the week, over a rolling horizon spanning the week.

For these experiments, we created six one-week instances involving only VMI customers. For each of the three dates considered in the previous experiments, we selected the first 750 customers after sorting them as above. On day 1, the customer demands q_i are those computed by linear interpolation through the history of the deliveries. For the subsequent days, a fixed consumption rate, in percentage of the customer's tank capacity, is used for all the customers to determine the demands of the customers that are not visited earlier. For each date, two instances were created: one with 7-minute service times and 10-minute replenishment times, another with 8-minute service times and 15-minute replenishment times.

Each instance was solved using the following rolling horizon procedure. First, an ODVRP is built for the first day. The mandatory customers are those with a demand q_i above 95% of their effective tank size U_i and the optional customers are those that will meet this condition in the next two days (assuming the fixed consumption rate stated above). This ODVRP is solved using the proposed LNS-CG heuristic. All customers serviced in the computed solution are then removed from the set of customers and a new ODVRP is defined for the next day. For this ODVRP, the sets of mandatory customers and optional customers are defined as for the first day (from the set of customers not serviced yet), using the adjusted demands of the customers. This ODVRP is solved and serviced customers are removed from the set of customers. This process repeats until reaching the last day of the week. Note that the optional customer bonuses must be recomputed each day. For these experiments, the LNS-CG heuristic is the only method used because it outperformed the other methods in Section 7.1. Its parameter setting is the same as the one applied for the single-day 250-customer instances.

Three series of experiments were conducted. In the first series, we evaluate the impact of the value of parameter w used in formula (4). Recall that this parameter controls the weight attributed to each pair of customers that can be adjacent to a customer i for which the bonus β_i is computed. A w value very close to 1 indicates that only the pairs that are the closest to customer i have a significant weight when computing this bonus. On the other hand, a value close to 0 means that a larger number of pairs has an impact in this computation. In the second series of tests, we assess the magnitude of the bonuses with respect to the traveled distances. If the bonuses are too high, then a large number of optional customers should be serviced even if this is not profitable overall. At the opposite, if they are too low, then larger detours will be incurred for visiting certain optional customers in the subsequent days. Finally, we solved the instances using bonuses computed as in Dror and Ball (1987), that is, with formula (3).

For the first series of tests, the six one-week instances were solved using different values of w ranging from 0.01 to 1.0. Again, ten runs were performed for each instance. The quality of a solution is evaluated by the total distance traveled over the week. To ensure a fair comparison between the different parameter values, we force the covering of the same customers, namely, those that become mandatory during the week. Out of the 750 customers considered, there are 567 such customers on average. The other customers are, however, used to compute the bonuses of the optional customers. The results obtained are reported in Table 5 which provides for each value of w the following averages (taken over the best solution founds): the total distance traveled during the week to service the 567 customers, the total number of customers serviced on the day they were mandatory, and the total number of customers serviced as optional customers. These results indicate that a w value between 0.05 and 0.1 yields the least total distance. With a value of $w = 0.1$ (resp. $w = 0.05$), the detours for the first 22 (resp. 32) pairs of adjacent customers have a weight greater than 0.01 in the weighted average for computing the bonuses. For higher w values, the bonuses tend to underestimate the detours that can be saved in the future, resulting in less optional customers serviced in advance. At the opposite, very low w values overestimate the optional customer bonuses, yielding more optional customers serviced in advance even if it does not worth it for many of them.

For the second series of tests, we first introduced a bonus multiplier μ that is applied to every bonus, that is, every bonus β_i, $i \in \mathcal{N}_o$, is replaced by $\mu\beta_i$. Then, we solved the six one-week instances (ten times each) with $w = 0.1$ and different μ values varying between 0 and 2. Again, only the customers becoming mandatory during the week must be serviced. Table 6 reports the results of these experiments. They show that formula (4) with $\mu = 1$ produces

w	Total dist.	No. cust. served as mandatory	optional
0.01	1982.5	196.0	371.0
0.025	1898.2	200.0	367.0
0.05	1847.5	227.5	339.5
0.10	1846.8	239.5	327.5
0.15	1856.3	255.0	312.0
0.25	1904.4	273.0	294.0
0.50	1943.1	288.1	278.9
1.00	2084.1	315.0	252.0

Table 5: LNS-CG results for different values of w (one-week instances)

μ	Total dist.	No. cust. served as mandatory	optional
0.00	2689.6	567.0	0.0
0.25	2176.4	429.7	137.3
0.50	1996.7	332.8	234.2
0.75	1900.5	282.5	284.5
1.00	1846.8	239.5	327.5
1.25	1907.0	233.3	333.7
1.50	2034.9	205.7	361.3
1.75	2068.6	171.5	395.5
2.00	2123.7	166.5	400.5

Table 6: LNS-CG results for different values of μ (one-week instances)

the best solutions, that is, those with the least total traveled distance. Hence, with $\mu = 1$, the bonuses seem to estimate with a sufficient accuracy the detours that will be incurred by visiting the optional customers later.

In the final series of tests, we applied the LNS-CG heuristic embedded into a rolling horizon procedure for solving the one-week instances using formula (3). To compute the fraction $\psi_1(l)$, we chose neighborhoods H_l of the same size for all customers $l \in N$. We performed tests with the sizes 3, 5, and 7. To compute the fraction $\psi_2(l)$, we used the depot or replenishment station that was closer to each customer $l \in N$. Different values of parameter ν were tested, namely, 0.2, 0.4, 0.6, and 0.8. The results of these experiments are reported in Table 7. As one can observe, the average total distance increases when the neighborhood size increases. This may be explained by smaller bonuses when more customers are considered in a neighborhood (the total tour cost is divided among the customers), yielding less customers served as optional. The results also indicate that smaller distances are obtained when $\nu = 0.4$,

$\|H_l\|$	ν	dist.	No. cust. served as mandatory	optional
3	0.2	2341.2	391.2	175.8
	0.4	2314.2	396.0	171.0
	0.6	2329.8	406.9	160.1
	0.8	2350.5	421.2	145.5
5	0.2	2384.5	432.0	135.0
	0.4	2356.2	436.7	130.3
	0.6	2383.9	437.0	130.0
	0.8	2395.6	455.5	111.5
7	0.2	2397.2	452.8	114.2
	0.4	2388.3	455.8	111.2
	0.6	2390.8	460.2	106.8
	0.8	2419.6	466.3	100.7

Table 7: LNS-CG results using formula (3) for computing the bonuses (one-week instances)

showing that it is important to consider both criteria (proximity between the customers and distance between the customers and the closest depot) used by Dror and Ball (1987).

Comparing these results with those obtained with the proposed formula (4), we clearly see that formula (4) yields much better quality solutions. Indeed, the best total distance it yielded was 1846.8, which is much smaller than 2314.2, the best total distance produced with formula (3). These results suggest that considering a set of possible detours to compute the bonuses is more accurate than focussing on a single one.

8 Conclusion

In this paper, we addressed the ODVRP that arises in the heating oil distribution industry. To solve this problem, we developed three metaheuristics: a TS algorithm and two LNS algorithms, one based on TS and the other based on CG. Computational results on instances derived from a real-life database showed that the LNS-CG heuristic outperforms the other two algorithms. On the other hand, the LNS-TS heuristic can be considered as a good alternative for a company that does not want to invest into a commercial linear programming solver that is required for the LNS-CG method. In another series of experiments that consisted of solving a sequence of ODVRP over a one-week rolling horizon, we showed that the bonuses used as incentives for covering the optional customers are adequate to minimize the total distance traveled over the week. Our computational results also indicate that the bonuses

should be computed using a set of possible detours and not only a single one. As a future research direction, one can consider extending the proposed heuristics to treat a multiple product version of the ODVRP where the vehicles have several compartments of fixed sizes.

Acknowledgements: The authors wish to thank the personnel of Info-Sys Solutions (in particular, Yannick Grovalet, Asghar Mahmood and Michel Lafontaine) for proposing this problem to us and providing the data used for our computational experiments.

9 References

Adelman D. (2004). A price-directive approach to stochastic/inventory routing. *Operations Research* 52, 499–514.

Angelelli E., Speranza M.G. (2002). The periodic vehicle routing problem with intermediate facilities. *European Journal of Operational Research* 137, 233–247.

Bertazzi L., Savelsbergh M. and Speranza M.G. (2008). Inventory routing. In: B. Golden, S. Raghavan and E. Wasil (eds.), *The ehicle Routing Problem: Latest Advances and New Challenges*, Springer, New York, NY, 49–72.

Bostel N., Dejax P., Guez P. and Tricoire F. (2008). Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In: B. Golden, S. Raghavan and E. Wasil (eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges*, Springer, New York, NY, 503–525.

Cordeau J.-F., Laporte G., and Mercier A. (2001). A unified tabu search heuristic for the vehicle routing problems with time windows. *Journal of the Operational Research Society* 52, 928–936.

Cordeau J.-F., Laporte G., Savelsbergh M.W.P., Vigo D. (2007). Vehicle routing. In: C. Barnhart, G. Laporte (eds.), *Transportation, Handbooks in Operations Research and Management Science* 14, North-Holland, Amsterdam, 410–417.

Crevier B., Cordeau J.-F. and Laporte G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research* 176(2), 756–773.

Desaulniers G., Lessard F. and Hadjar A. (2008). Tabu search, generalized k-path inequalities, and partial elementarity for the vehicle routing problem with time windows. *Transportation Science* 42(3), 387–404.

Desrosiers J. and Lübbecke M.E. (2005). A primer in column generation. In G. Desaulniers, J. Desrosiers, and M.M. Solomon (eds), *Column Generation*, Springer, New

York, NY, 1–32.

Dror M. (2005). Routing propane deliveries. In: Langevin, A. and Riopel, D. (eds.), *Logistics Systems: Design and Optimization*, Springer, New York, NY, 299–322.

Dror M., Ball M.O. (1987). Inventory/routing: Reduction from annual to a short period problem. *Naval Research Logistics* 34, 891–905.

Dror M., Ball M.O., Golden B. (1985/6). Computational comparison of algorithms for inventory routing. *Annals of Operations Research* 4, 3–23.

Golden B., Raghavan R., Wasil E., eds. (2008). *The vehicle routing problem: Latest advances and new challenges*, Springer, New York, NY.

Glover F. and Laguna M. (1997). Tabu search. Kluwer, Norwell, MA.

Irnich S. and Desaulniers G. (2005). Shortest path problems with resource constraints. In: G. Desaulniers, J. Desrosiers, and M.M. Solomon (eds.), *Column Generation*, Springer, New York, NY, 33–65.

Kleywegt A.J., Nori V.S. and Savelsbergh M.W.P (2002). The stochastic inventory routing problem with direct deliveries. *Transportation Science* 36(1), 94–118.

Moin N.H. and Salhi S. (2007). Inventory routing problem: A logisitical overview. *Journal of the Operational Research Society* 58, 1185–1194.

Nagata Y., Bräysy O. and Dullaert W. (2010). A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research* 37(4), 724–737.

Pisinger D. and Ropke S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research* 34, 2403–2435.

Prescott-Gagnon E., Desaulniers G. and Rousseau L.-M. (2009). A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks* 54(4), 190–204.

Prescott-Gagnon E., Desaulniers G., Drexl M. and Rousseau L.-M. (2010). European driver rules in vehicle routing with time windows. *Transportation Science* 44(4), 455–473.

Rousseau L.-M., Gendreau M., and Pesant G. (2002). Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics* 8, 43–58.

Shaw P. (1998). Using constraint programming and local search methods to solve vehi-

cle routing problems. In: Maher, M., and Puget, J.F. (eds.), *Proceedings of Constraints Programming '98*, Springer-Verlag, New York, NY, 417–431.

Tarantilis C.D., Zachariadis E.E. and Kiranoudis C.T. (2008). A hybrid guided local search for the vehicle routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing* 20(1), 154-168.