Spot Market versus Full Charter Fleet

EURO Journal on Decision Processes 10 (2022) 100022

Contents lists available at ScienceDirect

EURO Journal on Decision Processes

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

Spot market versus full charter fleet: Decision support for full truck load tenders

Veaceslav Ghilas[a], Ivo Hedtke[a,*], Joachim Weise[a], Tom Van Woensel[b]

[a] Global Data Strategy & Analytics, Schenker AG, Kruppstr. 4, Essen 45128, Germany
[b] School of Industrial Engineering, Eindhoven University of Technology, the Netherlands

ARTICLE INFO

2020 MSC:
90-05
90B90
90B06
90B35

Keywords:
Full truck load
Fleet sizing
Dynamic routing
Tender support
Pickup and delivery problem with time windows

ABSTRACT

Given the dynamics of today's markets, decision support based on advanced analytics is required to help market players keep their top position. This paper presents an approach to help business decision-makers gain market share by providing competitive tender offers for Full Truck Load (FTL) services. In particular, we compare operating a fleet of full charter trucks (FCT), using Spot-Market (SM) capacity and a mixture of both options against each other. A Pickup and Delivery Problem is modeled, and solved using an Adaptive Large Neighborhood Search heuristic. Computational results indicate strong service benefits combining FCT and SM usage. Numerical experiments are presented in detail to support the findings. Additionally, a real-life case study originating from DB Schenker is presented.

1. Introduction

Full Truck Load (FTL) transportation gained limited attention from the scientific community, compared to Less-than-Truck Load (LTL) or express deliveries Wieberneit (2008). However, there is a multitude of aspects related to FTL transportation, which have a significant financial impact and are worth researching, *e.g.*, fleet sizing, dynamic routing, or FTL solution design.

Freight forwarders regularly compete at tenders organized by large shippers who need tailored full-load logistics solutions. To win such tenders, and thus gain more market share, the forwarders need to provide cost-efficient full-load solutions, which are relatively complex to analyze. The core trade-off when designing such solutions is the following: on the one hand, the solution needs to be economical and avoid overexposure to prohibitively high fixed costs; on the other hand, the shipper needs to be provided with sufficient capacity at nearly any moment, irrespective of volume development.

Consequently, a freight forwarder has two extreme options to organize capacity. One would be to cover all shipping demand exclusively through flexible capacity sourced from SM. Usually, SM can provide service at short notice. This flexibility, however, comes at a relatively high market price and can not be taken for granted in special market situations, *e.g.*, during peak seasons or around public holidays.

Alternatively, the forwarder could invest into a fleet of FCT dimensioned according to the (assumed) peak demand. The main idea is the following: rather than providing a customer an offer based on SM prices per trip or trade lane, the forwarder would instead calculate the costs of chartering and managing a fleet of trucks on a long-term basis, *e.g.*, several months or years. When sizing this fleet properly and dispatching it efficiently, the offer involving FCT may eventually be more competitive than an SM solution.

Finally, an effective mix of both solutions mentioned above may, at least in theory, provide an even more cost-efficient solution than both individual options introduced above.

As stated earlier, winning tenders and increasing market share is the primary goal when designing such logistics solutions. Yet, the design process itself includes costs and profits as secondary aspects. An internal analysis of a (cost-)effective mix of SM- and FCT-options provides internal transparency about expected costs which enables the submission of a competitive tender bid. The profit aspect is in the hands of the bid writer after studying the costs of the designed solution.

The problem has similarities with a multitude of problems extensively studied by researchers over the years. Specifically, it boils down to the Pickup and Delivery Problem with Time Windows (PDPTW) (Sol and Savelsbergh, 1995), in particular to the FTL variant of the PDPTW.

* Corresponding author.
E-mail address: ivo.hedtke@dbschenker.com (I. Hedtke).

https://doi.org/10.1016/j.ejdp.2022.100022
Received 8 November 2021; Received in revised form 15 June 2022; Accepted 30 June 2022

A recent study on FTL-PDPTW can be found in Soares et al. (2019). The authors propose a meta-heuristic to tackle the problem with synchronization constraints. The approach is validated in a case study on the biomass logistics industry. In a related study, Xue et al. (2021) develop a column generation heuristic algorithm to solve large problem instances for FTL routing problem with multiple shifts. The described approach outperforms the previous attempts in terms of computational time and solution quality. Due to the lengthy planning horizons considered, driving regulation constraints become crucial in achieving realistic solutions. Goel (2009, 2010, 2018) study the driving regulations and modeling approaches in the vehicle routing context. Goel and Vidal (2013) describe a heuristic algorithm designed to solve the vehicle scheduling problem and analyze different driving regulations around the world. In a related study, Goel and Irnich (2016) propose an exact approach to tackle the problem to optimality. In addition, considering the assumption that the trucks do not have pre-defined start and end depots leads us to a particular case of the well-known Open Vehicle Routing Problem (OVRP) (Repoussis et al., 2007), where vehicles do not have to return to a depot. A recent study on OVRP can be found in Lahyani et al. (2019). The authors propose a hybrid adaptive large neighborhood search approach to solve large-scale problems for OVRP and manage to find new best-known solutions for specific instances. Furthermore, various routing problems with practical side constraints, such as FTL property and driving regulations, have been studied by scientists in the last decades. The interested reader is referred to Annouch et al. (2016); Braekers et al. (2016); Vidal et al. (2013) for recent surveys on different types of routing problems.

The orienteering problem Archetti et al. (2009) is another related well-known problem. In this context, each resource (*i.e.*, truck) aims at finding a minimum cost (or the most profitable) path, starting at a pickup location and ending at a delivery location. Expensive shipments, in terms of FCT cost, would then be serviced by SM. In this sense, the aim is to find the right trade-off between FCT and SM costs. Note that both the vehicle routing problem and the orienteering problem are similar types of problems. The distinction is that the orienteering problem maximized profits, rather than minimizes costs. The orienteering problem also does not force that every location is visited. Orienteering problems can thus be seen as a routing problem with profits (Van Steenwegen and Gunawan, 2019).

The contributions of the paper at hand are as follows:

1. We analyse the needed decision support for FTL transportation and the related tendering process.
2. We model the described problem as a Pickup and Delivery Problem and propose a tailored adaptive large neighborhood search (ALNS) to solve the underlying routing problem.
3. Numerical results show that the mix of SM and FCT may lead to cheaper FTL solutions compared to traditional pure SM strategy, having the transformed VRP Gehring & Homberger instances as benchmarks.
4. We also present a practical case application to the tender support for FTL business based on a real-life setting originating from DB Schenker. DB Schenker is a freight forwarder that supports industry and trade in the global exchange of goods by land transport, worldwide air and ocean freight, contract logistics and supply chain management.

This paper is organized as follows. The tender decision process is briefly sketched in Section 2. In Section 3 we formally define the considered mathematical problem, we present the related literature, and discuss the transformation of the business scenario into the formal problem. In Section 4 we present a single-solution meta-heuristic tailored to solve the problem at hand. Finally, we discuss the experiments on transformed literature instances and real-world data sets involving two customers and the obtained solutions in Section 5.

2. Tender decision process

For tender preparation, freight forwarders generally are given a historical data-set of shipments, which is assumed to be representative. Shipment data includes origin, destination locations, along with timing/lead-time and other requirements, if applicable.

For operations planning, it is of great importance when the shipment information becomes available. In real-life, the shipments are generally not known well in advance (*e.g.*, one month in advance), and if so, the information often changes.

Once the shipment data is fixed and known, the dispatchers investigate the available options: possibly multiple SM offers, and, if available, FCT. If no FCT fleet is available, the dispatchers generally choose the cheapest SM offer. However, if the FCT option exists, the dispatchers would choose it only if the total cost, including the incurred empty travel, would not exceed the cheapest offer from SM (considering actual locations of the FCT vehicles).

The dispatching approach described above may have economic drawbacks, thus leading to high operating costs. In particular, the dispatchers might choose an insignificantly cheaper SM offer compared to FCT. However, at the end of the *e.g.*, month, the already available FCT vehicles may incur additional costs if they were underutilized due to such sub-optimal decisions by dispatchers. Hence, from an operational perspective, a dynamic decision support tool is needed to assure economic viability, however, this is not the scope of this paper.

As aforementioned, SM is dynamic and to reduce the risk, freight forwarders consider certain safety buffer in the assumed SM rates for the tender offers. For this purpose, predictive analytic methods are generally used. For the scope of this paper, we assume the SM rates to include the safety buffer given by the business experts.

3. Problem description and mathematical model

We consider a set R of transportation requests between a finite set of locations L. Each request $r \in R$ has an origin $r^o \in L$, a destination $r^d \in L$, as well as a time window r^o_{tw} at the origin and a set of time windows r^d_{tws} at the destination. A time window is defined by two absolute points in time. Every request must be picked up at its origin within its origin time window, transported directly to its destination and be delivered within one of its destination time windows. The travel-time and travel-distance between two locations ℓ, ℓ' are given by $t_{\ell,\ell'}$ and $d_{\ell,\ell'}$.

Each transportation request r can be outsourced at a cost of s_r. The remaining requests must be served by a set of vehicles at a cost of κ per unit of driven distance.

The task at hand is to find a cost-optimal assignment of all requests to the options of outsourcing it or serving it by a vehicle. Note that the set of vehicles is determined as part of the task.

Serving the non-outsourced requests by vehicles is subject to the following constraints:

- Each vehicle can serve at most one request at any given point in time.
- Each vehicle starts at the origin of its first request, ends at the destination of its last request, and has to drive a distance of at least μ.[1]
- Each (un-)loading operation requires a time of σ. The complete (un-)loading operation has to be executed within one of the respective time windows. Hence, if a vehicle arrives outside the time windows, it has to wait for the next one.

[1] Note that this induces a minimum cost of $\mu\kappa$ per vehicle. Thus, an explicit fixed cost per vehicle is not considered.

Table 1
Parameters and variables in the model.

Variable	Meaning
$\boldsymbol{x}_a$	vehicle is traveling along arc a (=1) or not (=0)
$\boldsymbol{y}_{n\to m}$	distance from origin to m when a vehicle travels along $n \to m$
$\boldsymbol{l}_n = (l_{n,1}, l_{n,2})$	state of the driver at node n (arrival time, nonstop driving time)
Parameter	Meaning
d_a	distance of arc a
κ	cost of driving per unit distance
μ	minimum driving distance of a vehicle
M	sufficiently large constant
s_r	cost of out-sourcing request r to the spot-market

- Each vehicle has to fulfill the following driving time regulations: A shift-break is a break of duration of at least τ_b and a Sunday-break is a break of duration of at least $\tau_s \geq time$. After a duration of τ_n cumulative driving without shift-breaks, a shift-break is required. (Un-)Loading does not count as break time, whereas waiting time does. Every Sunday, a Sunday-break is required.

A mathematical model combining classical arc-based vehicle routing, the minimum driving distance constraint from (Kara, 2011, Section III), and the labeling technique from Goel and Gruhn (2006) for the driving time regulations is constructed in three phases: First, we define a graph as basis for an arc-based vehicle routing model. Then, a model respecting everything except for the timing constraints is built. Finally, the timing constraints (time windows, simplified driving regulation) are added.

Variables are denoted by bold lower-case letters. Instance independent parameters are denoted by greek letters. We write $\mathbb{B} := \{0,1\}$ and $\mathbb{N} := \{0,1,2,\dots\}$. By M we denote a sufficiently large constant. An overview of the variables and parameters can be found in Table 1.

Let $G := (N, A)$ denote the directed graph constructed as follows: The set N of nodes consists of

$$N := \{(r^o, r) : r \in R\} \cup \left\{\left(r^d, r\right) : r \in R\right\} \cup \left\{n_0, n_\infty\right\}$$

all origin- and destination locations together with two artificial nodes n_0 and n_∞ denoting the start and end of all tours. We write $N_R := N \setminus \{n_0, n_\infty\}$. For a request r we define r^s as the start of the time window r^o_{tw} and r^e as its end. The set A of arcs represents the vehicles realizing the non-outsourced request, traveling empty between locations or waiting at a location. It consists of the arcs serving requests $r \in R$, the tour start, the tour end, and waiting/empty travel:

$$\begin{aligned} A := &\{(r^o, r) \to (r^d, r) : r \in R\} \\ &\cup \{n_0 \to (r^o, r) : r \in R\} \\ &\cup \{(r^d, r) \to n_\infty : r \in R\} \\ &\cup \left\{(r_1^d, r_1) \to (r_2^o, r_2) : r_1 \neq r_2 \in R, r_1^s + t_{r_1^o, r_1^d} + t_{r_1^d, r_2^o} + 3\sigma \leq r_2^e\right\}. \end{aligned}$$

For an arc $a = n \to m$ we define $\text{target}(a) := m$ and $\text{source}(a) := n$ as the target and source of a. We extend the distances $d_{\ell,k}$ and travel times $t_{\ell,k}$ between locations to d_a and t_a on arcs $a \in A$ by

$$d_a := \begin{cases} 0 & \text{source}(a) = n_0, \\ 0 & \text{target}(a) = n_\infty, \\ d_{\ell,k} & a = (\ell, r) \to (k, r') \end{cases} \quad \text{and} \quad t_a := \begin{cases} 0 & \text{source}(a) = n_0, \\ 0 & \text{target}(a) = n_\infty, \\ t_{\ell,k} & a = (\ell, r) \to (k, r'). \end{cases}$$

Let $\boldsymbol{x}_a \in \mathbb{B}$ denote whether a vehicle is traveling along arc $a \in A$ ($\boldsymbol{x}_a = 1$) or not ($\boldsymbol{x}_a = 0$). Note that for an arc $a = (r^o, r) \to (r^d, r)$ this denotes whether the request r is served by vehicles ($\boldsymbol{x}_a = 1$) or out-sourced ($\boldsymbol{x}_a = 0$). As introduced in (Kara, 2011, Section III) we define $\boldsymbol{y}_{n\to m} \in \mathbb{N}$ as the total distance from the origin n_0 to node $m \in N$ traveled by a vehicle when it goes along $n \to m \in A$. We focus purely on the routing and the minimum distance per vehicle constraint:

$$\min \sum_{a\in A} \kappa d_a \boldsymbol{x}_a + \sum_{r\in R} s_r (1 - \boldsymbol{x}_{(r^o,r)\to(r^d,r)}) \tag{1a}$$

$$\text{s.t.} \sum_{\substack{a\in A \\ \text{target}(a)=(r^o,r)}} \boldsymbol{x}_a = \boldsymbol{x}_{(r^o,r)\to(r^d,r)} \qquad \forall r \in R \tag{1b}$$

$$\sum_{\substack{a\in A \\ \text{source}(a)=(r^d,r)}} \boldsymbol{x}_a = \boldsymbol{x}_{(r^o,r)\to(r^d,r)} \qquad \forall r \in R \tag{1c}$$

$$\sum_{\substack{a\in A \\ \text{source}(a)=n}} \boldsymbol{y}_a = \sum_{\substack{a\in A \\ \text{source}(a)=n}} d_a \boldsymbol{x}_a + \sum_{\substack{a\in A \\ \text{target}(a)=n}} \boldsymbol{y}_a \qquad \forall n \in N_R \tag{1d}$$

$$\boldsymbol{y}_{n_0\to(r^o,r)} = 0 \qquad \forall r \in R \tag{1e}$$

$$\boldsymbol{y}_a \leq M \boldsymbol{x}_a \qquad \forall a \in A \tag{1f}$$

$$\boldsymbol{y}_a \geq \mu \boldsymbol{x}_a \qquad \forall a \in A : \text{target}(a) = n_\infty \tag{1g}$$

$$\boldsymbol{x}_a \in \mathbb{B}, \boldsymbol{y}_a \in \mathbb{R}^+ \qquad \forall a \in A \tag{1h}$$

Description: The model minimizes the sum (1a) of the driving cost of non-outsourced requests and the outsourcing cost. If a request r is realized by own vehicles, there needs to be a vehicle entering the node (r^o, r), see (1b), and a vehicle leaving the node (r^d, r), see (1c). Driven distances $\boldsymbol{y}$ are initially set by (1e) at the artificial origin node, updated via (1d), and restricted to used arcs by (1f). As shown in Kara (2011), further subtour-elimination constraints are not required. (1g) ensures the minimum travel distance for each route.

Finally, we consider the simplified driving time regulations. Inspired by Goel and Gruhn (2006), we define for all nodes $n \in N_R$ a label:

$$\boldsymbol{l}_n = \begin{pmatrix} l_{n,1} \\ l_{n,2} \end{pmatrix} = \begin{pmatrix} \text{arrival time} \\ \text{nonstop driving time} \end{pmatrix}$$

to represent the state of the driver at the node. The vehicle can start the service at node $n \in N_R$ at time $l_{n,1}$ and can depart from n at time $l_{n,1} + \sigma$. It may drive $\tau_n - l_{n,2}$ before the next break.

As shown in (Goel and Gruhn, 2006, Section V), it is possible to precompute the set $\mathcal{L}_m(\boldsymbol{l}_n)$ of potential labels for a vehicle that is supposed to travel from node $n \in N_R$ with label $\boldsymbol{l}_n$ to a node $m \in N_R$. This takes

- the simplified driving regulation,
- the time window(s) of the requests $r \in R$ at origin and destination, and
- the Sunday-break τ_s

into account. Let $l^r := (r^s, 0)^\top$ denote the label of a vehicle when the request r is the first request within the tour.

The extended model taking time windows and driving time regulation into account is then

$$\min (1a)$$

$$\text{s.t.} (1b), (1c), (1d), (1e), (1f), (1g), (1h)$$

$$\boldsymbol{x}_{n_0\to(r^o,r)} = 1 \Longrightarrow \boldsymbol{l}_{(r^o,r)} = l^r \qquad \forall r \in R \tag{2a}$$

$$\boldsymbol{x}_{n\to m} = 1 \Longrightarrow \boldsymbol{l}_m \in \mathcal{L}_m(\boldsymbol{l}_n) \qquad \forall n \to m \in A : n, m \in N_R \tag{2b}$$

4. Solution approach

The algorithm presented in this section is a single-solution metaheuristic based on Large Neighborhood Search (LNS). The general idea originates in Shaw (1998), and has been gaining significant popularity in the recent past. In particular, it has proven to be highly effective in tackling vehicle routing problems and provides a good trade-off between solution quality and computational time. Note that in the literature the term *shipment* is used instead of *request*.

The main workflow of the algorithm is as follows: given an initial solution, iteratively modify it until a stopping criterion is reached. Two basic operators in the LNS framework are *removal* and *insertion* procedures. In the current context, *i.e.*, the pickup and delivery problem, the removal method un-plans a percentage ξ of already planned requests (pickup and delivery vertices associated with a request) from the solution, and the insertion method would re-plan these requests, by using, *e.g.*, an insertion heuristic. Note that for large instances, removing, *e.g.*, 20 % of the requests might still lead to a large optimization sub-problem, hence we limit it by an absolute upper bound ψ on the number of requests to remove. At every iteration, the modified solution is compared to the best solution found, and promoted to the new best solution if proved to be better. It is worth noting that a solution might contain unplanned requests, which are assumed to be serviced by SM, and thus incur additional costs. The original framework is extended to consider an adaptive mechanism (Pisinger and Ropke, 2007), *i.e.*, Adaptive LNS (ALNS), where user-defined removal and insertion methods are selected based on their performance during the search.

The high-level pseudo-code of ALNS is presented in Algorithm 1.

Algorithm 1 ALNS

```
s ← generateInitialSolution()
s_best ← s
while !stoppingCriterion do
    r* ← chooseRemovalOperator()
    i* ← chooseInsertionOperator()
    s' ← partiallyDestroySolution(r*, s)
    s_new ← repairSolution(i*, s')
    if accept(s_new) then
        s ← s_new
    if s_new < s_best then
        s_best ← s_new
    updateOperatorProbabilities()
return s_best
```

The interested reader is referred to Ghilas et al. (2016); Grimault et al. (2017); Pisinger and Ropke (2007); Ropke and Pisinger (2006) for more details about applying ALNS to a broad range of VRPs.

The starting solution s is constructed using a greedy algorithm. Its basic idea is to insert an unplanned request in the feasible position which increases the objective function value the least. Note that the initial solution out-sources all requests, nothing is planned on vehicles. A request is given to a vehicle only if the insertion cost is smaller than the corresponding out-sourcing cost. While the stopping criterion is not reached, *e.g.*, the maximum number of iterations m, the algorithm randomly selects one removal and one insertion operator, and applies them to the current solution s, thus generating a new solution s_{new}. The method `accept` verifies whether the newly-built solution is accepted. Simulated annealing acceptance criteria are used in the current implementation, similarly to Pisinger and Ropke (2007); Ropke and Pisinger (2006), so that worse solutions may also get accepted. If the newly-built solution is accepted, the current solution s is overwritten. Finally, the best solution found is returned.

Note that the probability of an operator being chosen is dynamically updated every nth iteration. The better the performance of the operator, the higher its chance of being chosen. Eventually, the algorithm will converge to using only good-performing operators, see Pisinger and Ropke (2007); Ropke and Pisinger (2006).

It is important to note that the number of vehicles is unlimited, however at the end of each ALNS iteration, certain number of requests may remain unplanned. This is mainly due to the fact that out-sourcing may be cheaper if a significant empty travel would be induced by servicing a specific request using a vehicle, or if simply the SM rate is cheaper than the FCT rate.

4.1. Removal operators

Several removal operators are used in the current implementation, as described below. Note the that acronyms are taken from the literature and use the term shipment instead of request.

RRR: Random Route Removal randomly selects a route and removes it from the solution. All routes have the same probability of being selected;

TRR: Time-based Route Removal is similar to RRR, however, the probability of a route being chosen depends on the total travel time of the corresponding route in the current solution. In other words, longer total travel time leads to a higher probability of being selected;

SRR: Stop-based Route Removal is similar to TRR, however, smaller number of requests planned within a route leads to a higher probability of being selected. The intuition behind this is that the fewer requests are planned in a route, the easier it is to re-plan them into other routes, thus avoiding this route in the solution.

RSR: Random Shipment Removal randomly selects a set of requests to be removed from the current solution. This operator helps in terms of search diversification.

TSR: Time-based Shipment Removal is similar to RSR, however, the probabilities of being selected depend on the incurred driving times. In particular, the probability of selecting a request is higher if, in a given solution, total driving time to its pickup location, and from its delivery location, is longer.

SR: Shaw Removal removes a set of requests similar to each other (Shaw, 1998). The similarity is defined by the distance between pickup locations and delivery locations of each pair of requests. In addition, the time windows of the requests are part of the similarity function. Two variants of SR are used: (*i*) with distance and time windows as similarity criteria, and (*ii*) only time windows as similarity criterion.

4.2. Insertion operators

We provide more details about the insertion operators used as follows.

Classical greedy: identifies in each iteration the request (of the set of unplanned requests and the set of (partial) routes) which incurs the lowest insertion cost and inserts it in its best feasible position. It repeats this operation until no unplanned request exists or no insertion cost (*i.e.*, FCT cost) is lower then the corresponding SM cost.

Regret insertion: finds the request which incurs the maximum *regret* if not inserted in its cheapest feasible position at every iteration. Let c_1 represent the cost of inserting the requests in the route with the cheapest feasible position within the FCT routes, c_2 – in another route with the second cheapest position, c_3 – third, etc. Then, the *regret function* can be defined as $c_2 - c_1$, known as 2-regret. For more details on regret insertion, please refer to Potvin and Rousseau (1993). In order to take into account more information when deciding which request to insert next, the regret function is generalized, known as k-regret, considering multiple routes. For example, it is possible to look at the three cheapest insertion positions. *i.e.*, $\sum_{i=2}^{k}(c_i - c_1)$, where $k = 3$ is the number of look-ahead insertion positions to take into account (*i.e.*, 3-regret). Note that several regret operators (with different k values) can be used within the ALNS.

The general framework of an insertion operator is shown in Algorithm 2: Given a set S_{in} of unplanned requests and a set P of partial routes, iteratively find the cost of inserting a request into a route

Algorithm 2 Insertion procedure

```
Require: S_in = unplannedRequests, P = partialRoutes, s_r (out-sourcing costs)
  S_out ← ∅
  while S_in ≠ ∅ do
    for r ∈ S_in, p ∈ P do
      c_rp ← computeCostOfInsertingRequestInRoute(r, p)
    (r*, p*) ← requestToInsertNext(c)
    if c_{r*p*} < s_{r*} then
      insertRequestInRoute(r*, p*)
    else
      p_new ← ∅
      insertRequestInRoute(r*, p_new)
      if allRoutesAreWellUtilized(P) and cost(p_new) ≤ s_{r*} then
        P ← P ∪ {p_new}
      else
        S_out ← S_out ∪ {r*}
    S_in ← S_in \ r*
  return S_out
```

that incurs the most beneficial change of the given *cost function*. If the insertion cost is cheaper than the corresponding out-sourcing cost, the insertion is performed.

Otherwise, a new trip is created only if no feasible insertion is found in the existing vehicle trips, and the corresponding out-sourcing cost is higher or equal than using a new vehicle trip. In addition, a new trip is created only if all vehicles used are utilized w.r.t. distance traveled at least μ per planning horizon.

Otherwise, the request is placed into the out-sourcing bank S_{out}, *i.e.*, the set of the requests which are out-sourced.

The algorithm is assumed to start with one vehicle available. As soon as the vehicles used are well-utilized as aforementioned, the algorithm makes an additional vehicle available.

We compute an insertion cost matrix c_{rp} for each unplanned request and each existing route. The method `requestToInsertNext` returns the request with the least incurred insertion cost, along with the route and the corresponding position within the route. Here, either classical greedy or a k-regret insertion is used. The selected request is then evaluated, *i.e.*, if the incurred insertion cost is cheaper compared to out-sourcing, it is inserted in the chosen route (in its cheapest position). The request is then removed from S_{in}. Note that in the event that no feasible insertion is found, the request r with the least cost per distance unit is selected (*i.e.*, $s_r/d_{s_o,s_d}$). Finally, the insertion procedure returns the out-sourcing bank S_{out}.

4.3. Constraints

The feasibility of (intermediate) solutions is enforced at any time during the run of the algorithm. To recap, the following constraints need to be satisfied during cost matrix computation: (*i*) every request is out-sourced or must be served by a vehicle, (*ii*) time window(s) at origin and destination of the requests must be respected, and (*iii*) breaks between working shifts as well as (*iv*) Sunday-breaks must be taken into account. Constraints (*ii*), (*iii*), and (*iv*) are enforced only if the requests are served by a vehicle. As aforementioned, out-sourced requests are assumed to satisfy all the considered constraints.

Constraints (*i*) and (*ii*) are straightforward to implement. Many researchers have investigated ways to consider these constraints in an efficient manner, *i.e.*, by using auxiliary data structures, *e.g.*, Campbell and Savelsbergh (2004); Savelsbergh (1990). However, when combined with constraints (*iii*) and (*iv*), it is not trivial to efficiently implement them within ALNS.

Table 2
Business-specific parameters.

Parameter	Description	Value	Unit
τ_n	duration of cumulative driving without shift-breaks	450	min
$time$	minimum duration of a break	990	min
τ_s	minimum duration of a Sunday-break	1,320	min
σ	duration of an (un-)loading operation	120	min
v	average speed of vehicles	70	km/h

Table 3
Technical parameters.

Parameter	Description	Value
m	maximum number of ALNS iterations	25,000
ψ	maximum absolute number of requests to remove	100
ξ	maximum relative number of requests to remove	35 %
n	every n iterations update operators weights	200

5. Computational experiments

First, we describe the assumptions made during the transformation of Gehring & Homberger VRP instances, along with their corresponding results. The second part presents a case study at DB Schenker and describes the instance characteristics of the input data sets and costs. In both sections, the costs for out-sourcing no request and out-sourcing all requests are computed for comparison. Finally, the solutions that assign all requests to outsourcing it or serving it by a vehicle are discussed.

The ALNS is implemented in C++11 and all experiments are run on an Intel Core i7-8750H machine @2.2GHz, with 16 GB DDR4-RAM @2.4GHz.

Algorithm parameters The algorithm contains various parameters that need to be set. Some of them are business-related, others are heuristics technical in nature. The business-related parameters, along with corresponding explanation and values, are shown in Table 2.

After performing extensive computational experiments, we found that the best performance can be achieved by applying the removal operators RRR, SRR, SR, TSR, and RSR and the insertion operators greedy, 4-regret, 5-regret, and 6-regret, as well as the technical parameter settings of Table 3.

Furthermore, we apply simulated annealing (SA) acceptance criteria in our ALNS framework, as inspired by Ropke and Pisinger (2006). We considered a similar parameter setup as in Ghilas et al. (2016), as it proved to be beneficial for solution quality: In particular, generating a new best solution is rewarded with 33, generating an improved current solution, but not better the best-known solution is rewarded with 9, and finally, generating a solution, not better than the current one, but accepted by the SA mechanism is rewarded with 13. In such setup diversification is rewarded.

Note that this presented parameter setup is used throughout this section.

5.1. Transformed Gehring & Homberger instances

In this section, we present the computational results obtained by solving the transformed well-known Gehring & Homberger VRP instances (Gehring and Homberger, 1999). To convert the literature instances, we made the following assumptions:

- Node 0 is ignored, as it corresponds to the depot;
- Demand, capacity and service time data is ignored;
- For an instance with N nodes, node n and $N/2 + n$ correspond to pickup and delivery nodes of a request;
- To assure that the planning horizon consists of multiple days, we multiplied the Euclidean distances and pickup start time windows

Table 4
Assumptions on spot-market rates.

distance δ	δ < 150 km	150 km ≤ δ <350 km	350 km ≤ δ
cost in EUR/km	1.75	1.40	1.15

by a factor $f = 6$. The resulting distances are then rounded to the closest integer;

- For simplicity, all locations (*i.e.*, pickup and delivery locations) are assumed to be open between 06:00 and 18:00;
- The transformed pickup time is used to determine the ready day within the planning horizon. Then, the corresponding time window from 06:00 to 18:00 of that day is used as pickup time window. Both time windows of the delivery location are computed given the pickup time window;
- SM rates per km are assumed as shown in Table 4;
- FCT rate is assumed 1.06 EUR/km;
- μ is computed for each instance separately: 250 km per day with pickups in the planning horizon.

Table 5 presents the results obtained after solving the transformed Gehring & Homberger instances. In particular, the numbers show the averages over all corresponding instances for each instance class (*e.g.*, C100). Three scenarios are computed: all requests served by an own fleet, all requests out-sourced, and a mix between those as mentioned above.

Not surprisingly, the results indicate that considering the mix of FCT and SM leads to best results w.r.t. operating costs. In particular, on average 0.5 % savings can be achieved for the clustered (C) instances by serving approx. 5 % of the requests using own vehicles compared to outsourcing everything. On the other hand, an average of 0.2 % cost savings can be achieved in the mixed scenario for random (R) and randomly-clustered (RC) instances by serving approx. 2 % of the requests using own vehicles.

For the C instances, more savings can be achieved compared to R and RC instances. The main reason is that requests that are compatible from a temporal point of view are more compatible from a geographical point of view. In other words, the chances that a pickup location of a request is close to a delivery location of another request are higher in C instances. Hence less empty driving can be achieved.

Note that serving all requests by own vehicles is the most costly out of all scenarios considered. A minimum-driving-distance constraint is enforced, and the requests are not perfectly compatible in terms of timing and spatial aspects.

5.2. DB Schenker case study

DB Schenker is one of the key players in the global logistics sector. Founded in 1872 by Gottfried Schenker in Vienna, Austria, Schenker & Co. began its business by consolidating rail consignments from Paris, France, to Vienna, Austria. DB Schenker is a freight forwarder that supports industry and trade in the global exchange of goods by land transport, worldwide air and ocean freight, contract logistics, and supply chain management. With more than 76,000 employees working in approx. 2,000 locations around the world, DB Schenker is a leader in its industry.

DB Schenker's land transport in Europe covers 36 countries and offers a variety of products and services. One of them is *direct* product for large loads, *e.g.*, FTLs that are transported directly from consignor to consignee.

Large shippers frequently conduct tenders that require logistics providers to tailor dedicated full-load solutions. DB Schenker participates in such tenders via sales department representatives, who need to prepare the business offers. To come up with competitive solutions, different scenarios need to be analyzed, such as: operating a fleet of FCT, using SM capacity, as well as a mix of both options mentioned above.

The approach presented in this paper aims at helping sales departments get more insights about various scenarios, given *e.g.*, historical/forecasted FTL transportation demands. As a result, sales teams can become more competitive during tenders.

Two of DB Schenker's customers conducted a tender, and the sales department was given the following problem (per customer):

A set of plants is operating on a weekly schedule with working hours per day. For an entire month, all full-load shipments that have to be transported between the plants are given. Every shipment has a pickup date and must be transported directly (no consolidation, no pickup on a later date, etc.) to its destination.

For small instances, an offer can be created by assuming that everything is given to the spot market or that the volume can be included in DB Schenker's internal transportation network. However, for large instances as given by the two customers (with up to 12,427 full-load shipments per month), none of the approaches has enough capacity, and none of them would enable the sales department to present a competitive offer. An option with enough capacity that can be operated at a competitive cost is the combination of its own dedicated fleet of trucks and outsourcing to the spot market.

To determine the size of such a fleet and the cost of operating it, the above model can be applied: The full-load shipments correspond to requests R that have to be transported between the locations L given by the plants. The origin time window r^o_{tw} of a request r is given by the operating hours of the pickup plant. The destination time windows r^d_{tws} is given by the operating hours of the destination plant. Note that up to two such time windows are required, as a truck can arrive late and wait for the next day. The two options of using the spot market and operating an own fleet correspond to outsourcing requests and serving them by a set of vehicles, respectively.

As the result should be a business offer within the tender process, no party requires a detailed schedule for the trucks of the own fleet. A simplified driving regulations model based on shifts and including the Sunday-driving ban is sufficient.

5.2.1. Data description

In this section, we describe the request- and cost-related data we used in the analysis.

Request data Two data sets obtained from DB Schenker *customers* are used. We consider two instances for Customer 1 and three instances for Customer 2, each consisting of monthly request data, representing low-, average- and high-demand months. In contrast to Customer 2, the demand for Customer 1 is relatively stable over time. Hence the difference between high- and low-volume months is rather insignificant. Table 6 provides the total number of requests, along with the corresponding sum of the direct origin-destination distances over all requests for all instances. Fig. 1 visualizes the number of in- and out-going requests per location. Note that most locations have a significant imbalance between in- and out-going requests. Additionally, the maps in Figs. 2 and 3 visualize the number of requests as well as the in- and out-degree.

Fig. 3 indicates that for Customer 1, quite some locations are imbalanced, *i.e.*, has either (almost) only incoming, or (almost) only outgoing requests. In contrast, the demand from Customer 2 looks more balanced, except for the western location.

Distances between physical locations were computed using the Open Source Routing Machine (Luxen and Vetter, 2011).

Overall, it can be observed that the problem instances are large, with up to 12,427 requests.

As aligned with the FTL operations team, the minimum amount to travel by an own vehicle was set to μ = 8,000 km and μ = 5,000 km, for Customer 1 and 2, respectively.

Note that the ultimate goal is to generate insights regarding the FCT fleet size for each customer, as operational plans are out of scope in this paper. Hence, solving multiple scenarios depending on different monthly volumes helps the sales department develop cost-efficient and reliable business offers.

Table 5
Costs (average: grouped by class and number of nodes) for the mixed scenarios. Distances are given in 1,000 km. Minimum distance μ per own vehicle is enforced in all scenarios (*i.e.*, assumed fixed cost $\mu\kappa$). %req. denotes the percentage of requests. Costs are given in 1,000 EUR. The highlighted cost is better.

Class	out-sourcing		out-sourcing		mixed scenario						
	nothing		everything		own vehicles				out-sourced		Σ
& Nodes	km	cost	km	cost	% req.	km loaded	km empty	cost	km	cost	cost
C 100	58.0	64.5	45.0	54.6	2.2	1.14	0.04	1.3	44.0	53.2	**54.5**
C 200	145.0	154.5	115.0	136.4	5.2	6.00	0.18	6.5	109.0	129.0	**135.6**
C 300	324.0	343.3	267.0	310.1	5.2	17.29	0.41	18.8	250.0	290.0	**308.8**
C 400	527.0	558.3	449.0	518.6	5.6	29.70	0.74	32.3	419.0	484.2	**516.5**
C 500	869.0	921.2	764.0	881.0	7.2	67.31	1.15	72.6	697.0	803.4	**876.0**
R 100	58.0	63.3	44.0	53.0	2.0	0.90	0.02	1.0	43.0	51.9	**52.9**
R 200	154.0	163.6	120.0	142.2	2.0	2.35	0.08	2.6	118.0	139.4	**141.9**
R 300	342.0	362.7	272.0	314.9	1.7	6.06	0.11	6.5	266.0	308.0	**314.5**
R 400	607.0	643.0	497.0	573.8	2.3	15.26	0.32	16.5	482.0	556.2	**572.7**
R 500	960.0	1017.9	791.0	910.9	3.2	34.09	0.49	36.7	757.0	871.7	**908.4**
RC 100	59.0	63.1	44.0	53.4	2.1	1.20	0.03	1.3	43.0	52.0	**53.3**
RC 200	155.0	164.5	126.0	147.7	2.4	3.55	0.08	3.8	122.0	143.5	**147.4**
RC 300	338.0	357.8	262.0	305.2	2.0	6.35	0.19	6.9	256.0	297.8	**304.7**
RC 400	622.0	659.2	500.0	577.8	2.5	17.82	0.43	19.3	482.0	557.3	**576.6**
RC 500	980.0	1039.2	806.0	929.5	2.8	30.08	0.69	32.6	776.0	894.8	**927.5**

Table 6
Instances

	High Volume Month		Average Volume Month		Low Volume Month	
Customer	#requests	kilometer	#requests	kilometer	#requests	kilometer
1	12,427	10,215,308	—	—	11,432	9,197,733
2	2,316	1,094,550	2,257	1,145,214	1,162	678,370

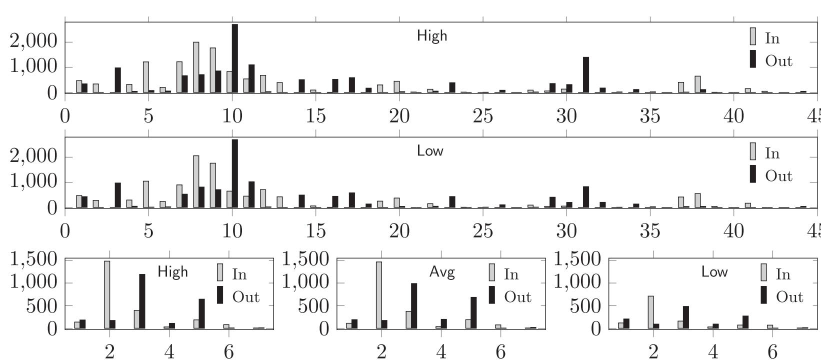

Fig. 1. In- and out-going requests per physical location. The plots show the number of in- (gray) and out-going (black) requests per location. The top and middle plot refer to Customer 1 and the remaining ones for Customer 2.

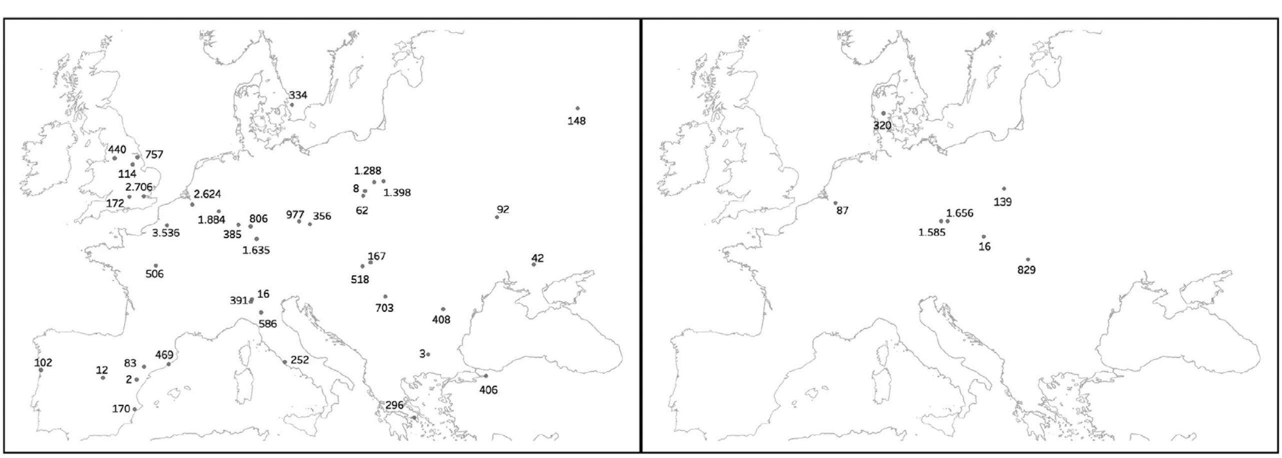

Fig. 2. Number of Requests per Location for Customer 1 (left) and 2 (right).

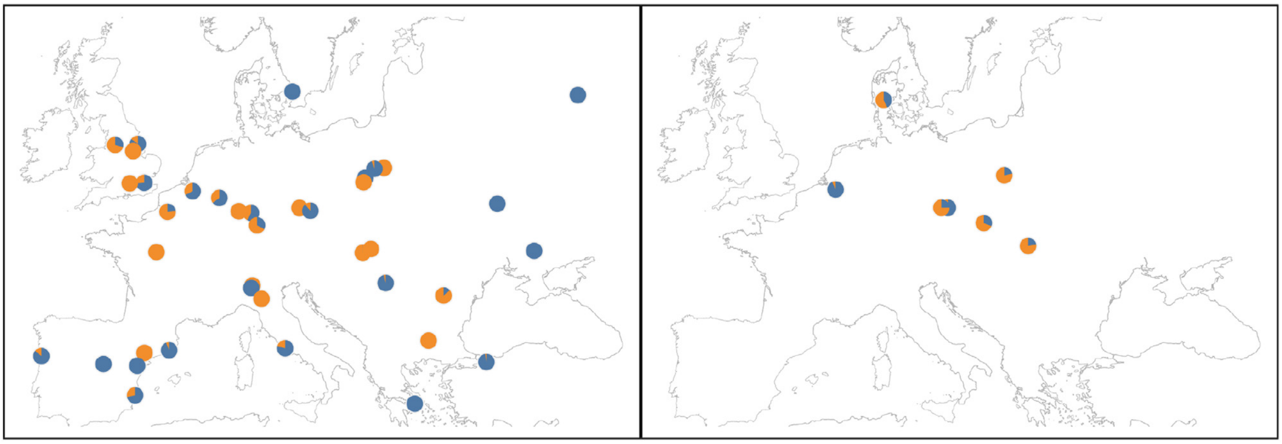

Fig. 3. In- and Out-Degree per Location for Customer 1 (left) and 2 (right). The pie-chart per location shows the relative in-degree in blue and the out-degree in orange.

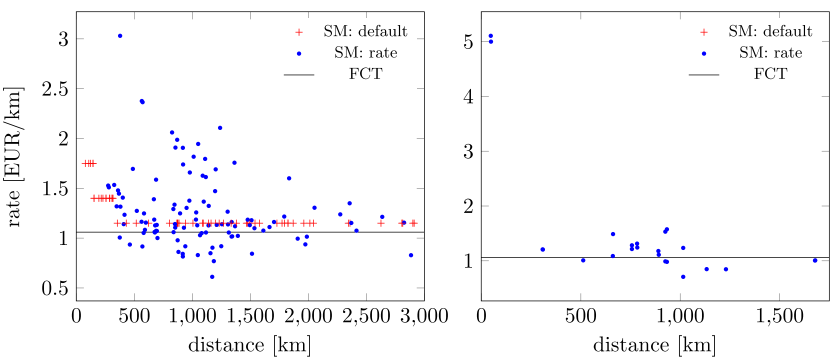

Fig. 4. Real SM rates for out-sourcing vs. approximated rates for out-sourcing vs. costs per km for vehicles for Customer 1 (left) and 2 (right). The individual points show the approximated *SM default* rates (red crosses) as well as the real *SM rates* for outsourcing (blue dots). The static cost of 1.06 EUR/km for vehicles is shown as a black reference line.

Cost data DB Schenker's FTL operations team provided us with SM rates to be used for outsourced requests corresponding to the given datasets. Since it was impossible to obtain all rates (for all points in time and all origin-destination pairs), we had to make assumptions regarding missing rates. In close collaboration with our business partners, we managed to develop realistic assumptions for static SM rates per kilometer.

In particular, we distinguish three different specific rate levels, depending on the distance between origin and destination, as shown in Table 4. Regarding the cost per driven kilometer of the FCT vehicles, our business partners provided a cost benchmark of 1.06 EUR/km (both empty or loaded). Fig. 4 displays the relation between specific rates in EUR/km and the distance from the origin to the destination. In particular, *SM: default* rates indicate the distance-specific approximation described above, whereas *SM: rate* indicates the real rates for out-sourcing to the SM obtained from the operations team.

For Customer 1, approximately 35 % of the SM rates could not be obtained by the operations team and were, thus, approximated using the approach described above. In contrast to that, all SM rates could be retrieved for Customer 2. No clear trend can be observed in Fig. 4. This is mainly because specific rate levels differ strongly between trade lanes, and that they are not symmetric per trade lane: a transport from Eastern Europe to Western Europe, with a distance of, *e.g.*, 750 km would cost less than the corresponding trip in the opposite direction. Also, rate levels are dynamic in time. Certain months of the year experience high demand for transportation services, hence the costs get higher. Similarly, for low-volume months the transportation costs are expected to be cheaper.

The enforced minimum total distance μ per vehicle, along with an appropriate cost κ per unit of driven distance, reflects the fixed cost of using a vehicle, *i.e.*, labor, truck leasing, etc. In other words, we assume that if a vehicle is used, it would cost at least $\mu\kappa$. In addition, the vehicles are assumed to be deployed only for the customer which *e.g.*, organizes the tender (set R).

In order to assure that the final business offer is economically viable, considering spot market volatility, s_r (for $r \in R$) contains a certain amount of buffer, as discussed with the operations team.

Note that these costs reflect the market at the time of writing this article, and an additional uncertainty buffer.

5.2.2. *Out-sourcing everything vs. nothing*

Here, we present the computational results of enforcing all requests to be outsourced and all requests to be served by vehicles. Computing the cost of outsourcing is easy. We have to sum up all outsourcing costs of all requests. For computing the cost of serving all requests by own vehicles, we consider the cost of unplanned requests within the ALNS as a huge number.

Tables 7 and 8 indicate the results considering complete sets of requests for each customer and month. In particular, Table 7 shows the number of vehicles needed to perform the service. Additional columns present the empty travel distance between loads relative to the loaded travel distance, *i.e.*, between the previous delivery and the next pickup, KPIs related to the distance traveled per vehicle, and, finally, the computational time needed.

It can easily be seen that significant additional empty travel would be required in a scenario where nothing is outsourced. One of the main reasons for this is, of course, the imbalanced demand structure that was displayed in Fig. 3. In particular, several locations in both data sets serve either as origins (sources) or destinations (sinks). This naturally leads to empty runs, *e.g.*, when a vehicle delivers at a pure sink location and then is forced to travel empty to its following source location.

Table 7
Computational results for a solution out-sourcing nothing. Distances per vehicle are given in 1,000 km. The additional empty distances are given relatively to the loaded distances in Table 6.

Cus-tomer	Month	#requests	#vehicles	Additional Empty Dist.	Dist. p. Vehicle Max	Avg	Min	CPU [sec]
1	High	12,427	1,172	+31.3 %	18.8	11.7	8.0	4,386
1	Low	11,432	1,095	+35.1 %	18.8	11.9	8.0	3,917
2	High	2,316	153	+35.6 %	13.2	9.7	5.4	525
2	Avg	2,257	153	+41.1 %	13.8	10.6	5.4	727
2	Low	1,162	95	+33.8 %	12.5	9.6	5.4	345

Table 8
Costs for solutions out-sourcing everything and out-sourcing nothing. Distances are given in 1,000 km. Costs are given in 1 million EUR. The highlighted cost is better. #req. denotes the number of requests.

out-sourcing		everything							nothing	
		requests grouped per orig.-dest. dist. δ								
Customer		δ<150		150≤δ<350		350≤δ				
	Month	#req.	Σkm	#req.	Σkm	#req.	Σkm	cost	km	cost
1	High	458	46.9	2,651	726.3	9,318	9,442.1	**12.43**	13,417	14.22
1	Low	435	44.3	2,650	747.4	8,347	8,406.0	**11.40**	12,431	13.18
2	High	1,099	52.7	16	4.9	1,201	1,036.9	**1.50**	1,485	1.57
2	Avg	958	45.9	24	7.4	1,275	1,091.9	**1.50**	1,616	1.71
2	Low	439	21.0	17	5.3	706	652.1	**0.85**	908	0.96

Table 9
Costs for the mixed scenarios. Distances are given in 1,000 km. %req. denotes the percentage of requests. #veh. denotes the number of vehicles. Costs are given in 1 million EUR. The highlighted cost is better.

	out-sourcing nothing		out-sourcing everything		mixed scenario own vehicles					out-sourced		Σ
Customer & Month	km	cost	km	cost	% req.	km loaded	km empty	# veh.	cost	km	cost	cost
1 High	13,417	14.22	10,215	12.43	6.9	294.4	39.3	36	0.35	9,921	11.83	**12.18**
1 Low	12,431	13.18	9,198	11.40	6.2	264.8	24.2	31	0.31	8,933	10.87	**11.18**
2 High	1,485	1.57	1,095	1.50	41.1	56.2	35.7	14	0.10	1,038	1.26	**1.36**
2 Avg	1,616	1.71	1,145	1.50	35.4	50.2	26.0	11	0.08	1,095	1.30	**1.38**
2 Low	908	0.96	678	0.85	34.5	26.3	15.0	7	0.04	652	0.74	**0.79**

To assess the merits of a scenario without outsourcing, we compare its overall cost to that of a scenario where everything is outsourced, as summarized in Table 8. In all five instances, outsourcing nothing is less competitive than outsourcing everything, with an overall cost disadvantage of up to 16 %. Again, this result does not come as a real surprise since an outsourcing approach is, at least in theory, fully flexible and of unlimited capacity and, thus, more efficient in serving the rather imbalanced and erratic demand patterns inherent to our customer instances.

5.2.3. Cost-optimal mix

After computing the costs for outsourcing no request and outsourcing all requests for comparison, we compute the cost-optimal assignment of requests to outsourcing options or an own fleet of vehicles. We call a scenario where both options are allowed a *mixed scenario*.

Table 9 indicates results for the most cost-efficient mixed scenarios found and compares them to outsourcing nothing and outsourcing everything. In all five instances, the mixed scenario is more competitive than any other two options, although the relative gain is more marked for Customer 2 than Customer 1.

The demand structure of Customer 1 (as discussed in Section 5.2.1) shows an extreme imbalance of in- and out-going requests for many locations. The geographic distribution of the locations induces many long-distance connections, which lead to a high amount of empty travel. To be financially beneficial, a spot-market rate has to be twice as high as the rate for operating an own vehicle. This is rarely the case, as shown before. As a result of this effect, we see that only a minimal amount (up to 7 %) of the requests is served by own vehicles, whereas the vast majority is outsourced. For a high-volume month, the mixed scenario is 2 % cheaper (12.18 million instead of 12.43 million EUR) than outsourcing everything to the SM. In a low-volume month, savings are even less.

As certain spot market rates for Customer 2 are quite expensive, compared to own vehicle rate κ, 41 % of the requests are served by own vehicles. The significant relative empty travel of the vehicles would still outweigh the high spot market costs. Consequently, for a high/average/low-volume month, the mixed scenario is 9 %/7 %/6 % cheaper than out-sourcing everything to the spot-market, respectively.

6. Conclusions

In this paper, we have presented a meta-heuristic to effectively support business development units at DB Schenker in designing competitive offers for complex full-load solutions. In particular, in order to calculate the total number of vehicles needed, the distance traveled, and the full-load requests that are served by an own fleet of vehicles instead of out sourcing them to SM. We modeled the problem as a variant of the PDPTW with driving regulations, and tackled it using a tailored ALNS.

We compared three scenarios, namely out-sourcing nothing, out-sourcing everything to the spot-market, and a mix of both, and quantified the corresponding costs. For evaluation purposes, we used transformed VRP instances widely used in the scientific literature (*i.e.*, Gehring & Homberger) and real-life instances from two potential customers of DB Schenker, containing monthly demand data with up to

12,427 requests. For these instances, out-sourcing nothing is outperformed by out-sourcing everything due to the underlying demand structure. However, a mixed setup may yield benefits of up to 9 % compared to out-sourcing everything.

To actually realize the potential benefits presented in this paper, the developed approach would need to be complemented by an operational decision support system, which can help control towers continuously plan, execute, and re-plan scenarios in case of demand fluctuations, driving time deviations or other unexpected events. The operational system needs access to real-time traffic data and own vehicles should be equipped with tracking devices that allow timely detection of potential deviations or disruptions.

Also from an algorithmic point of view, our approach could be extended in a number of ways. As aforementioned, when using the presented algorithm as operational decision support, additional aspects/parameters need to be added to allow for considering dynamic aspects of the problem as well as problem heterogeneity, *e.g.*, different cost assumptions per lane or geography when operating own vehicles. We have also indicated above that we took a rather pragmatic approach towards driving time constraints which currently doesn't reflect the full complexity of various national regulatory regimes.

In addition, when using the approach for tender calculations, demand uncertainty should be incorporated into the optimization procedure. This implies that routing solutions would need to be more conservative, hence more costly to some extent. However, incorporating the uncertainty explicitly into the model would increase the robustness of the solutions, and thus lead to less probability of unexpected costs due to demand fluctuations.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Annouch, A., Bouyahyaoui, K., Bellabdaoui, A., 2016. A literature review on the full truckload vehicle routing problems. In: Alaoui, A.E.H., Benadada, Y., Boukachour, J. (Eds.), 3rd International Conference on Logistics Operations Management, GOL 2016, Fez, Morocco, May 23-25, 2016. IEEE, pp. 1–6. doi:10.1109/GOL.2016.7731723.

Archetti, C., Feillet, D., Hertz, A., Speranza, M.G., 2009. The capacitated team orienteering and profitable tour problems. J. Oper. Res. Soc. 60 (6), 831–842. doi:10.1057/palgrave.jors.2602603.

Braekers, K., Ramaekers, K., Nieuwenhuyse, I.V., 2016. The vehicle routing problem: state of the art classification and review. Comput. Ind. Eng. 99, 300–313. doi:10.1016/j.cie.2015.12.007.

Campbell, A.M., Savelsbergh, M.W.P., 2004. Efficient insertion heuristics for vehicle routing and scheduling problems. Transp. Sci. 38, 369–378. doi:10.1287/trsc.1030.0046.

Gehring, H., Homberger, J., 1999. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Proceedings of the EUROGEN99: Short Course on Evolutionary Algorithms in Engineering and Computer Science. University of Jyväskylä, Finland

Ghilas, V., Demir, E., Van Woensel, T., 2016. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. Comput. Oper. Res. 72, 12–30. doi:10.1016/j.cor.2016.01.018.

Goel, A., 2009. Vehicle scheduling and routing with drivers' working hours. Transp. Sci. 43 (1), 17–26. doi:10.1287/trsc.1070.0226.

Goel, A., 2010. Truck driver scheduling in the European Union. Transp. Sci. 44 (4), 429–441. doi:10.1287/trsc.1100.0330.

Goel, A., 2018. Legal aspects in road transport optimization in Europe. Transp. Res. Part E 114, 144–162. doi:10.1016/j.tre.2018.02.011.

Goel, A., Gruhn, V., 2006. Drivers' working hours in vehicle routing and scheduling. In: IEEE Intelligent Transportation Systems Conference, ITSC 2006, Toronto, Ontario, Canada, 17-20 September 2006. IEEE, pp. 1280–1285. doi:10.1109/ITSC.2006.1707399.

Goel, A., Irnich, S., 2016. An exact method for vehicle routing and truck driver scheduling problems. Transp. Sci. 51, 395–789. doi:10.1287/trsc.2016.0678.

Goel, A., Vidal, T., 2013. Hours of service regulations in road freight transport: an optimization-based international assessment. Transp. Sci. 48, 313–463. doi:10.1287/trsc.2013.0477.

Grimault, A., Bostel, N., Lehuédé, F., 2017. An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization. Comput. Oper. Res. 88, 1–14. doi:10.1016/j.cor.2017.06.012.

Kara, I., 2011. Arc based integer programming formulations for the distance constrained vehicle routing problem. In: 3rd IEEE International Symposium on Logistics and Industrial Informatics, pp. 33–38. doi:10.1109/LINDI.2011.6031159.

Lahyani, R., Gouguenheim, A.-L., Coelho, L.C., 2019. A hybrid adaptive large neighbourhood search for multi-depot open vehicle routing problems. Int. J. Prod. Res. 57 (22), 6963–6976. doi:10.1080/00207543.2019.1572929.

Luxen, D., Vetter, C., 2011. Real-time routing with openstreetmap data. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, New York, NY, USA, pp. 513–516. doi:10.1145/2093973.2094062.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435. doi:10.1016/j.cor.2005.09.012.

Potvin, J.-Y., Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Eur. J. Oper. Res. 66 (3), 331–340. doi:10.1016/0377-2217(93)90221-8.

Repoussis, P.P., Tarantilis, C.D., Ioannou, G., 2007. The open vehicle routing problem with time windows. J. Oper. Res. Soc. 58 (3), 355–367. doi:10.1057/palgrave.jors.2602143.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40, 455–472. doi:10.1287/trsc.1050.0135.

Savelsbergh, M.W.P., 1990. An efficient implementation of local search algorithms for constrained routing problems. Eur. J. Oper. Res. 47 (1), 75–85. doi:10.1016/0377-2217(90)90091-O.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M.J., Puget, J. (Eds.), Principles and Practice of Constraint Programming – CP98, 4th International Conference, Pisa, Italy, October 26-30, 1998, Proceedings. Springer, pp. 417–431. doi:10.1007/3-540-49481-2_30.

Soares, R., Marques, A.F., Amorim, P., Rasinmäki, J., 2019. Multiple vehicle synchronisation in a full truck-load pickup and delivery problem: A case-study in the biomass supply chain. Eur. J. Oper. Res. 277 (1), 174–194. doi:10.1016/j.ejor.2019.02.025.

Sol, M., Savelsbergh, M.W.P., 1995. The general pickup and delivery problem. Transp. Sci. 29, 17–29.

Van Steenwegen, P., Gunawan, A., 2019. Orienteering Problems: Models and Algorithms for Vehicle Routing Problems with Profits. Springer.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. Eur. J. Oper. Res. 231 (1), 1–21. doi:10.1016/j.ejor.2013.02.053.

Wieberneit, N., 2008. Service network design for freight transportation: areview. OR Spectrum 30, 77–112. doi:10.1007/s00291-007-0079-2.

Xue, N., Bai, R., Qu, R., Aickelin, U., 2021. A hybrid pricing and cutting approach for the multi-shift full truckload vehicle routing problem. Eur. J. Oper. Res. 292 (2), 500–514. doi:10.1016/j.ejor.2020.10.037.