Noname manuscript No.
(will be inserted by the editor)

Fleet Design Optimisation From Historical Data Using Constraint Programming and Large Neighbourhood Search

Philip Kilby · Tommaso Urli

Received: date / Accepted: date

Abstract We present an original approach to compute efficient mid-term fleet configurations, at the request of a Queensland-based long-haul trucking carrier. Our approach considers one year's worth of demand data, and employs a constraint programming (CP) model and an adaptive large neighbourhood search (LNS) scheme to solve the underlying multi-day multi-commodity split delivery capacitated vehicle routing problem. Our solver is able to provide the decision maker with a set of Pareto-equivalent fleet setups trading off fleet efficiency against the likelihood of requiring on-hire vehicles and drivers. Moreover, the same solver can be used to solve the daily loading and routing problem. We carry out an extensive experimental analysis, comparing our approach with an equivalent mixed integer programming (MIP) formulation, and we show that our approach is a sound methodology to provide decision support for the mid- and short-term decisions of a long-haul carrier.

Keywords Vehicle Routing Problem · Fleet Size and Mix · Large Neighbourhood Search · Mixed Integer Programming · Constraint Programming · Pre-processing

1 Introduction

Long-haul transportation is a fundamental component of every country's economy, supporting both the movement of raw materials between production facilities and the delivery of final products to their ultimate destinations [5]. Transportation also represents a significant fraction of the final price of goods, and it is broadly acknowledged that making an efficient use of the available transportation infrastructure and resources, e.g., trucks, railways, warehouses, etc., is an essential strategy to reduce costs and remain profitable in the open market. The *split delivery* vehicle routing problem (SDVRP) is a relaxation of the classic vehicle routing problem (VRP) [18], in which the demand of one customer can be cumulatively satisfied

Optimisation Research Group, NICTA Canberra Research Lab,
7 London Circuit, Canberra, 2601 ACT, Australia
E-mail: {philip.kilby,tommaso.urli}@nicta.com.au

by multiple vehicles. It has been shown [9] that split delivery vehicle routing has great potential for savings with respect to classic vehicle routing, in addition to being, in some circumstances, the only available option. Moreover, split delivery vehicle routing is a genuinely complex problem, belonging to the renowned class of NP-hard problems [10].

Both VRP and SDVRP assume a given fleet of (possibly diverse) vehicles. The problem of how to design such a fleet is called the *fleet mix* (or *fleet size and mix*) vehicle routing problem (FSMVRP), and is often studied as a generalisation of VRP where the fleet is not specified in advance but, instead, a collection of available vehicle *types* is provided. We argue that such an interpretation, while adequate for applications in which the *same exact* routing plan is repeated every day, e.g., mail delivery and collection, is insufficient to address mid- to long-term fleet configuration issues for delivery companies dealing with a demand that can vary significantly from day to day. To the best of our knowledge the problem of designing the best fleet for a long planning horizon, in such context, has not been addressed yet.

In this paper, we take an optimisation perspective on some of the mid- and short-term decisions involved with running a profitable long-haul split delivery transportation service, namely *i)* the design of a robust and efficient fleet, to support the customer demand over a mid-term, e.g., few months to one year, planning horizon, and *ii)* the efficient loading and routing of vehicles. Our study is carried out at the request of a long-haul trucking carrier operating in the Queensland area. The paper is organised as follows. In Section 2 we briefly summarise some of the most relevant literature on fleet mix optimisation, split delivery vehicle routing, and long-haul transportation. In Section 3 we formalise the problem formulation tackled in this work. In Section 4 we describe our approach, which can be decomposed into two phases: pre-processing and optimisation. Finally, in Section 5, we describe our experimental analysis and results. We conclude with some remarks on the strengths and limitations of the proposed approach, and we outline its future extensions.

2 Related work

In [5], Crainic surveys long-haul transportation as a whole, highlighting the various decisions that can be made at the strategic (long-term) level, e.g., demand modelling, location of facilities, physical network design, etc., tactical (mid-term) level, e.g., service network design, resource acquisition, etc., and operational (short-term) level, e.g., routing and loading of vehicles. An important insight in this regard, is that decisions at each level *constrain* the decisions at the following level, and *inform* the decisions at the previous level.

A number of approaches have been proposed in the literature to solve the fleet mix optimisation problem. Most of these contributions target rich extensions of the classical vehicle routing problem. As such, they are often limited to single-commodity and single-day scenarios. Dell'Amico *et al.* [6] present a heuristic solution to the fleet size and mix for the single-day single-commodity vehicle routing problem with time windows, based on a previous linear programming formulation of the same problem. The proposed heuristic approach comprises an initial phase in which unassigned customers are inserted into existing or new routes through

a constructive procedure, and a *ruin and recreate* improvement phase based on the same procedure. Although the problem being solved is different from the one we address, the proposed ruin and recreate search strategy shares many similarities with our adaptive large neighbourhood search scheme. In particular, in our approach, the construction heuristic is represented by a time-limited branch & bound tree-search based on a constraint programming model, and the ruin strategies are represented by our *destroy* (or *relaxation*) modes. Vidal *et al.* [20] propose a component-based framework to solve a variety real-world VRP variants, including fleet size and mix vehicle routing. The presented components address specific *problem attributes* (often referred to as *side constraints* in the CP literature), and consists of local search, route splitting, and genetic operators that can be used to configure the underlying solver for the particular VRP variant at hand. The approach doesn't handle split deliveries or multiple commodities.

A number of approaches in literature have explored the split delivery variant of the vehicle routing problem. In [1], Archetti and Speranza provide a survey of split delivery vehicle routing, along with a classic linear programming formulation and a thorough discussion of complexity aspects and lower bounds. Fleet mix aspects are, however, not considered. Belfiore *et al.* [2] describe a *scatter search* approach to solve the fleet mix problem for the daily split delivery vehicle routing problem. Again, the work shares some similarities with our approach, in that the demand can be served cumulatively by many vehicles; however the fleet mix problem is only considered in a single-day and single-commodity scenario. Bräysy *et al.* [4] introduce a *multi-phase local search* approach for the FSMVRP, which also considers acquisition costs. The approach assumes a single-day perspective, or, equivalently, a constant routing plan across all the days.

With respect to the reviewed literature, our approach introduces the following novelties. First, we consider the fleet mix problem in a mid-term perspective, i.e., we seek to find a *single* fleet design that can support the demand (and minimise the operation costs) across a long time span, e.g., months to years, rather than a different fleet setup for every day. This information is crucial to inform the mid- to long-term decisions such as which fleet to acquire. Second, we consider a multi-commodity split delivery vehicle routing problem, which is common in real-world industrial contexts, but often disregarded in classic formulations. Third, we propose a Pareto dominance-based technique to extract guaranteed representative subsets of days from the historical data, and we use them in place of the whole horizon, so to make the overall problem more tractable.

3 The problem

Our client serves a set of customers $c \in \mathbf{C}$ with locations $l_c \in \mathbf{L}$, and faces a daily multi-commodity, split delivery, long-haul vehicle routing problem, to be addressed using a heterogeneous fleet of trucks. On a daily basis, each customer $c \in \mathbf{C}$ may issue an order for a quantity $\mathbf{q}_{c,k} \in \mathbb{Z}^*$ of some commodity $k \in \mathbf{K}$. Since there is a one-to-one relation between customers and locations, in the following we will use the sets $\mathbf{C}$ and $\mathbf{L}$ interchangeably.

The available vehicle types $t \in \mathbf{T}$ to carry out the deliveries differ in *i)* their total carrying capacity, denoted by $\mathbf{b}_t$, *ii)* the kind of commodities that can be transported, specified through a Boolean compatibility relation $\mathbf{comp}_{t,k}$ which is

true if vehicles of type $t \in \mathbf{T}$ can carry commodity $k \in \mathbf{K}$, and **false** otherwise, and *iii)* the operation cost per kilometre, denoted by $\mathbf{w}_t$. Table 1 reports the described parameters for each one of the available vehicle types.

According to our client's requirements, all the goods must be picked up from a central depot, henceforth denoted by **dep**, with location $l_{\mathbf{dep}}$ and all the vehicles must return to it once they have delivered their payload. We are given the distance in kilometres, $\mathbf{dist}_{l_i,l_j}$, between each pair of locations $l_i, l_j \in \mathbf{L} \cup \{l_{\mathbf{dep}}\}$. With respect to transported goods, our client delivers refrigerated (or *chilled*) and non-refrigerated (or *ambient*) products, but only some of the vehicle types are refrigerated. Ambient goods can be transported in refrigerated trucks along with chilled goods.

#	Model	Cost per km ($\mathbf{w}_t$)	Capacity ($\mathbf{b}_t$)	Refrigerated
0	B-Double	2.59	34	no
1	A-Double	2.67	40	no
2	B-Triple	2.86	48	no
3	B-Double Reefer	2.99	34	yes
4	A-Double Reefer	3.04	40	yes

Table 1: Vehicle data.

3.1 Operational and tactical perspectives

The above problem can be seen from two perspectives[1]. At the *operational* (day to day) level, we want, given a *fixed* fleet, to load and route the vehicles so that the daily demand of all the customers is met, the vehicle capacities and compatibilities are respected, and the total operation cost, computed as the sum of distances travelled by each vehicle weighted by the vehicle's cost per kilometre, is minimised. Since the total demand for some of the customers may be higher than the capacity of the largest truck, we allow more than one vehicle to visit a customer (split delivery) so that the demand can be met cumulatively. Moreover, on a given day, it is not necessary to use all the available vehicles, on the contrary, a good loading and routing strategy minimises the number of employed vehicles, since the long-haul nature of the problem causes a strong correlation between the number of vehicles used and the travelled distance.

At the *tactical* level, given an unlimited availability of each type of truck, we want to design a fleet to be used in the next mid-term period, which has enough capacity to address high-demand days (without the need for on-hire vehicles and drivers), but is lean enough that few vehicles are left unused on low-demand days. In order to accomplish this, we have access to one year of daily demand data.

[1] In the following, we employ the terms *operational* and *tactical* as they are used in the context of management science. Note, however, that in military contexts they are usually reversed.

4 Proposed approach

Following the principle, expressed in [5], that decisions at the tactical level must be informed by the decisions at the operational level, we propose to solve the fleet mix problem by extending the daily loading and routing problem to a multi-day perspective, and by relaxing[2] the fixed fleet constraint. This allows our method to find the most suitable fleet design across the whole planning horizon, instead of producing a different fleet design for each day.

In practice, our approach consists of an adaptive large neighbourhood search (LNS) scheme on top of a constraint programming (CP) model. In the following, we also present an equivalent mixed integer programming (MIP) formulation of the problem, which we consider as a possible alternative for solving the underlying routing problem. Both approaches are described in the following section (4.1), while an experimental comparison is presented Section 5.

Solving the extended problem for long planning horizons, e.g., one year, is a computationally challenging task. Therefore, we introduce a Pareto dominance-based approach to extract representative subset of days from the data, thus making the horizon shorter and the problem more tractable. Using this pre-processing phase, the obtained fleets are guaranteed to work on all the days represented by each subset. We describe this approach in Section 4.2.

4.1 Optimising a fleet in a multi-day scenario

In this section we describe a constraint programming formulation, an adaptive large neighbourhood search approach, and a mixed integer programming formulation for the multi-day multi-commodity split delivery capacitated vehicle routing problem.

The multi-day extension of the problem over a horizon $\mathbf{D} = \{0, \dots, D-1\}$ of D days, is obtained by augmenting the demand $\mathbf{q}_{c,k}$ with an additional index d indicating which day it belongs to ($\mathbf{q}_{d,c,k}$). Moreover, we assume a bound V on the maximum number of vehicles in the fleet, and we denote the set of all possible vehicles indices as $\mathbf{V} = \{0, \dots, V-1\}$. Finally, for model compactness, on each day $d \in \mathbf{D}$ we only consider the customers $\mathbf{C}_d \subseteq \mathbf{C}$ for which $\mathbf{q}_{d,c,k} > 0$.

4.1.1 Constraint programming formulation

Our CP model is inspired by the multi-day vehicle routing approach described by Di Gaspero *et al.* in [8], but it follows a step-based formulation (in which a solution directly encodes the routes as sequences of visits), rather than the classic vehicle routing formulation (in which routes are based on successor variables). The step-based formulation, similar to the one proposed by Di Gaspero *et al.* in [7], is especially suited for split delivery problems, unlike classic VRP formulations where introducing split deliveries requires replicating each customer by the number of vehicles. Moreover, unlike both [8, 7], our model can handle multiple commodities and a variable fleet, albeit bounded in size.

[2] By "relaxing" we mean that the fleet is not given *a priori*, but chosen by the solver. Additionally, we require that the vehicles used on each day belong to the fleet chosen by the solver.

Main variables. The variables in our model can be classified in two sets. The first set represents horizon-wide decisions, describing the overall composition of the fleet. In particular, for each vehicle $v \in \mathbf{V}$ we define a variable, $\texttt{v\_type}_v$, representing the type of the vehicle. The domain of such variables is $\{0, \ldots, |\mathbf{T}|\}$, where the vehicle type with index $|\mathbf{T}|$ is assigned to all unused vehicles. Moreover, for each $v \in \mathbf{V}$ we have a Boolean variable, $\texttt{used}_v$, which models whether a given vehicle is part of the fleet or not. Finally, the integer variable `n_veh`, with domain $\{1, \ldots, V\}$, represents the number of vehicles which are part of the fleet.

The second set of variables represents daily decisions. For each day $d \in \mathbf{D}$, and each vehicle $v \in \mathbf{V}$ a Boolean variable, $\texttt{used\_on}_{d,v}$, models whether vehicle $v \in \mathbf{V}$ is used on day d. Similarly, for each day $d \in \mathbf{D}$, the variable $\texttt{n\_veh\_on}_d$ models the number of vehicles used on day d. Note that the type and the general availability of each vehicle is decided at the horizon level, but we can decide not to use one or more vehicles on any given day.

Routes are constructed as sequences of steps (visits to customers). For the sake of clarity, we denote the set of steps in a day $d \in \mathbf{D}$ as $\mathbf{S}_d = \{0, \ldots, |\mathbf{C}_d| + 1\}$. The number of daily steps is the number of customers $|\mathbf{C}_d|$ with non-zero demand on day d plus two steps for the initial and final visits to the depot. The above bound on the route length is motivated by the fact that in long-haul contexts, because of the long distances being travelled, it is very unlikely that the same customer be visited twice by the same vehicle. For each day $d \in \mathbf{D}$, each vehicle $v \in \mathbf{V}$, and each step $s \in \mathbf{S}_d$, an integer variable $\texttt{vis}_{d,v,s}$, with domain $\{0, \ldots, |\mathbf{C}_d|\}$, represents the s-th visit of vehicle v on day d. The values 0 (or **null**) and 1 (or **dep**) in the domain of `vis` are reserved, respectively, for visits which do not happen, and for visits to the depot. Note that **null** visits can only appear at the end of a route, and exists to allow for routes shorter than $|\mathbf{S}_d|$. Regarding vehicle loading, for each day $d \in \mathbf{D}$, for each vehicle $v \in \mathbf{V}$, for each step $s \in \mathbf{S}_d$, and for each commodity $k \in \mathbf{K}$, an integer variable $\texttt{load}_{d,v,k,s}$ represents the amount of k present on v after step s is performed on day d. Finally, an additional variable $\texttt{act}_{d,v,k,s}$ represents the activity carried out at each node, where a negative activity denotes pick-up and positive activity denotes delivery. Activity variables do not represent decisions,

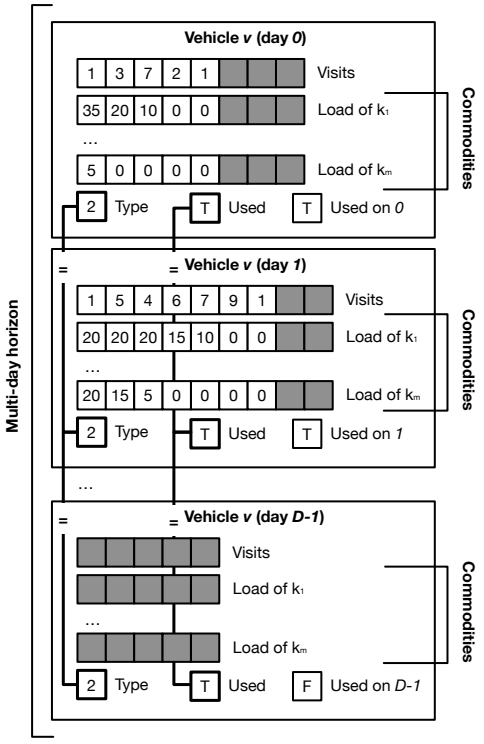

Fig. 1: Visual scheme of the daily variables.

and are computed as the variation in load at every step. They allow to express some of the constraints in a more convenient way.

The set of daily decision variables is represented schematically in Figure 1. In the figure, the lengths of the variable arrays reflect the number of steps, i.e., $|\mathbf{S}_d|$, and the grey steps mark the **null** visits.

Auxiliary variables. In order to define the cost measure, we also need some auxiliary variables, none of which represents a decision. For each day $d \in \mathbf{D}$, each vehicle $v \in \mathbf{V}$, and each step $s \in \mathbf{S}_d$, the total distance travelled up to step s by vehicle v on day d is represented with an integer variable $\texttt{tot\_dist}_{d,v,s}$. Such variable has a suitable domain, going from zero to the maximum distance between two locations multiplied by $max_{d\in\mathbf{D}}|\mathbf{C}_d|$. We call this measure the *maximum vehicle distance*. Similarly, for each $d \in \mathbf{D}$, and each vehicle $v \in \mathbf{V}$, the daily operation cost of vehicle v is modelled with a floating point variable $\texttt{tot\_cost}_{d,v}$ with a domain going from zero to the maximum vehicle distance multiplied by $max_{t\in\mathbf{T}}\mathbf{w}_t$. We call this the *maximum vehicle cost*. Finally, a `tot_cost` variable aggregates the daily vehicles costs through a sum constraint, providing the overall cost measure for a given solution s. During an optimisation run, the value of `tot_cost` is upper-bounded by the cost of the best solution $\hat{s}$ found so far (*incumbent solution*, or *incumbent*) by dynamically adding constraints of the form

$$\texttt{tot\_cost} \leq cost(\hat{s}). \tag{1}$$

Such constraints trigger a filtering of the domains of the $\texttt{tot\_cost}_{d,v}$ variables (through the propagator for the sum) and, in turn, of the $\texttt{vis}_{d,v,s}$ variables, which are decision variables.

Essential constraints. The model features essential constraints, which define the search space, and additional constraints, which are either aimed at eliminating symmetries, or at improving the propagation strength. In the following, we exhaustively explain the essential constraints. We only briefly address the additional constraints.

We start by defining constraints on the fleet design. In Constraint 2 we define a vehicle $v \in \mathbf{V}$ to be used on a day $d \in \mathbf{D}$ if and only if its first visit on day d is the depot. Constraint 3 states that a vehicle is part of the fleet if and only if it is used at least on one day. In Constraints 4 and 5 we establish the relation between the number of vehicles used on a day, and the total number of vehicles.

$$\texttt{vis}_{d,v,0} = \mathbf{dep} \Leftrightarrow \texttt{used\_on}_{d,v} \quad \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{2}$$

$$\texttt{used}_v = \vee_{d\in\mathbf{D}} \left(\texttt{used\_on}_{d,v}\right) \quad \forall v \in \mathbf{V} \tag{3}$$

$$\texttt{n\_veh} = \sum_{v\in\mathbf{V}} \texttt{used}_v \tag{4}$$

$$\texttt{n\_veh\_on}_d = \sum_{v\in\mathbf{V}} \texttt{used\_on}_{d,v} \quad \forall d \in \mathbf{D} \tag{5}$$

We proceed by stating the routing constraints for each day and each vehicle. Constraint 6 requires that the first step of each route be always either the depot

or the **null** visit. Together with Constraint 7, which forces each **null** visit to be followed only by **null** visits, this guarantees that either a vehicle is used on one day, or its whole route is **null** for that day. Additionally, we require that each route starts and ends with the depot. This is accomplished through Constraint 8, which states that the number of visits to the final depot must be equal to the number of visits to the initial depot. Similarly, Constraint 9 requires that all the visits following the final depot be **null** (i.e., the final depot is the last visit).

$$\texttt{vis}_{d,v,0} \in \{\textbf{null}, \textbf{dep}\} \;\; \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{6}$$

$$\texttt{vis}_{d,v,0} = \textbf{null} \Leftrightarrow \forall s \in \mathbf{S}_d \;\; \texttt{vis}_{d,v,s} = \textbf{null} \;\; \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{7}$$

$$count(\texttt{vis}_{d,v,2..|\mathbf{S}_d|} = \textbf{dep}) = count(\texttt{vis}_{d,v,0} = \textbf{dep}) \;\; \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{8}$$

$$\texttt{vis}_{d,v,s} \leq \textbf{dep} \Rightarrow \forall z \in s+1..max(\mathbf{S}_d) \;\; \texttt{vis}_{d,v,z} = \textbf{null}$$
$$\forall d \in \mathbf{D}, v \in \mathbf{V}, s \in 1..max(\mathbf{S}_d) \tag{9}$$

Constraint 10 requires that customers visited by a vehicle on a day be either different or **null**. This constraint makes sense in the context of long-haul transportation, since solutions where vehicles shuttle between customers and depots are unlikely to be optimal.

$$\textbf{distinct}_{\textbf{null}}(\texttt{vis}_{d,v,1..max(\mathbf{S}_d)}) \;\; \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{10}$$

Constraints 11 and 12 handle the initialisation and propagation of distances along a route. Distances to and from **null** visits are always zero, so that the distance at the last step is always the total travelled distance up to the final depot. The daily cost per vehicle is maintained in Constraint in 13 by multiplying the travelled distance up to each step by the cost per kilometre of the specific vehicle type being used.

$$\texttt{tot\_dist}_{d,v,0} = 0 \;\; \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{11}$$

$$\texttt{tot\_dist}_{d,v,s} = \texttt{tot\_dist}_{d,v,s-1} + \textbf{dist}_{\texttt{vis}_{d,v,s-1},\texttt{vis}_{d,v,s}}$$
$$\forall d \in \mathbf{D}, v \in \mathbf{V}, s \in 1..max(\mathbf{S}_d) \tag{12}$$

$$\texttt{tot\_cost}_{d,v} = \texttt{tot\_dist}_{d,v,max(\mathbf{S}_d)} \cdot \mathbf{w}_{\texttt{v\_type}_v} \;\; \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{13}$$

We now present the constraints relative to loading. Constraints 37 and 38 ensure that, respectively, the capacity and the compatibility with commodities are respected along the whole route. Constraints 16 and 17 require that, if a depot or a customer is visited by a vehicle, at least some activity be carried out. Conversely, Constraints 18 and 19 disallow any activity at the final depot and the **null** visits. Constraint 20 ensures that, cumulatively, the vehicles pick up all the needed quantity of each commodity at the depot.

$$\sum_{k \in \mathbf{K}} \texttt{load}_{d,v,k,s} \leq \mathbf{b}_{\texttt{v\_type}_v} \;\; \forall d \in \mathbf{D}, v \in \mathbf{V}, s \in \mathbf{S}_d \tag{14}$$

$$comp_{\texttt{v\_type}_v,k} = \textbf{false} \Rightarrow \texttt{act}_{d,v,k,s} = 0 \;\; \forall d \in \mathbf{D}, v \in \mathbf{V}, k \in \mathbf{K}, s \in \mathbf{S}_d \tag{15}$$

$$\texttt{vis}_{d,v,0} \neq \textbf{null} \Rightarrow \sum_{k \in \mathbf{K}} \texttt{act}_{d,v,k,0} \neq 0 \quad \forall d \in \mathbf{D}, v \in \mathbf{V} \tag{16}$$

$$\texttt{vis}_{d,v,s} > \textbf{dep} \Rightarrow \sum_{k \in \mathbf{K}} \texttt{act}_{d,v,k,s} > 0$$

$$\forall d \in \mathbf{D}, v \in \mathbf{V}, s \in 1..max(\mathbf{S}_d) \tag{17}$$

$$\texttt{vis}_{d,v,0} = \textbf{null} \Rightarrow \texttt{act}_{d,v,k,0} = 0 \quad \forall d \in \mathbf{D}, v \in \mathbf{V}, k \in \mathbf{K} \tag{18}$$

$$\texttt{vis}_{d,v,s} \leq \textbf{dep} \Rightarrow \texttt{act}_{d,v,k,s} = 0$$

$$\forall d \in D, v \in \mathbf{V}, s \in 1..max(\mathbf{S}_d), k \in \mathbf{K} \tag{19}$$

$$\sum_{v \in \mathbf{V}} \texttt{load}_{d,v,k,0} = \sum_{c \in \mathbf{C}} \mathbf{q}_{d,c,k} \quad \forall d \in \mathbf{D}, k \in \mathbf{K} \tag{20}$$

Constraints 21 and 22 establish the relation between load and activity, stating that the activity at each customer is the the difference between the load before visiting the customer and the load after visiting it. Constraint 23 ensure that all the customers' demands are met every day. Finally, Constraint 24 guarantees that all the goods picked up at the depot are eventually delivered at the customers.

$$\texttt{act}_{d,v,k,0} = -\texttt{load}_{d,v,k,0} \tag{21}$$

$$\texttt{act}_{d,v,k,s} = \texttt{load}_{d,v,k,s-1} - \texttt{load}_{d,v,k,s}$$

$$\forall d \in \mathbf{D}, v \in \mathbf{V}, k \in \mathbf{K}, s \in 1..max(\mathbf{S}_d) \tag{22}$$

$$\sum_{v \in \mathbf{V}, s \in 1..max(\mathbf{S}_d) | \texttt{vis}_{d,v,s} = c} \texttt{act}_{d,v,k,s} = \mathbf{q}_{d,c,k} \quad \forall d \in \mathbf{D}, c \in \mathbf{C}, k \in \mathbf{K} \tag{23}$$

$$\texttt{act}_{d,v,k,0} = \sum_{s \in 1..max(\mathbf{S}_d)} \texttt{act}_{d,v,k,s} \quad \forall d \in \mathbf{D}, v \in \mathbf{V}, k \in \mathbf{K} \tag{24}$$

The last essential constraint, Constraint 25, aggregates the daily vehicle costs (see Constraint 13) into a $\mathbf{w}$ variable that serves as objective to minimise.

$$\texttt{tot\_cost} = \sum_{d \in \mathbf{D}, v \in \mathbf{V}} \texttt{tot\_cost}_{d,v} \tag{25}$$

Additional constraints. A few additional constraints are used to improve propagation and break symmetries. First, the customers with zero demand on some day $d \in \mathbf{D}$ are excluded from the domains of the $\texttt{vis}_d$ variables, and the minimum number of visits to each valid customer on that day is constrained to be at least the total demand divided by the vehicle with maximum capacity. Second, for each commodity, we constrain the minimum number of vehicles to be greater than the total amount of the commodity, divided by the capacity of the largest vehicles compatible with it. A similar constraint enforces the total capacity of the fleet to

be greater than the total demand of the customers. Finally, all unused vehicles are assigned the $|\mathbf{T}|$ vehicle type, so to prevent the solver from trying to optimise them.

Constraints for fixed fleets. The above model allows to solve the multi-day multi-commodity split delivery vehicle routing problem at the core of our fleet mix optimisation approach. Once a fleet is found, the same model (with $\mathbf{D} = \{0\}$) can be used to solve the daily loading and routing problems. To do this, the `n_veh` and `v_type` variables are fixed at the beginning of the optimisation run, fixing the fleet composition. The rest of the variables are free for the solver to optimise. Observe that, in general, solving the problem with a fixed fleet is a much easier task because of the fewer decisions to be made.

Variable and value selection strategies. We define two different variable and value selection strategies, or *branchings*. The first is used for branch & bound search and to find the initial solution for LNS. The second is used exclusively for re-optimisation in the *repair* step of large neighbourhood search.

Branching 1. First, the total number of vehicles and their types are chosen, prioritising small fleets and large vehicles[3]. After the overall fleet has been chosen, the branching proceeds day by day, first choosing how many and which vehicles are dispatched, and then sorting out the loading and routing aspects. Once more, we try to use the minimum number of vehicles, and we give priority to vehicles with low indices. Regarding the loading aspects, we try to fill the vehicles as much as possible at the depot. Routes are built vehicle-wise, starting from the first vehicle and adding visit after visit until the (final) depot is chosen. At each step, the next customer to be visited is chosen by preferring higher indices first, so to avoid going back to the depot (which has the lower index) too soon. After a vehicle reaches the final depot, the procedure jumps to to the next one.

Branching 2. The variable selection strategy is the same used for **Branching 1**, i.e., we first choose the overall fleet, then the daily fleet, then we handle the loading and routing. The value selection strategy chooses values uniformly at random within the current variable domains, except for the number of vehicles and the total initial load which follow the same heuristics as in **Branching 1**.

The way in which the number and type of vehicles is selected, combined with the auxiliary constraints on the capacity of the fleet, allows us to identify viable fleets very quickly, i.e., in a matter of seconds, for the considered problem instances, even in multi-day scenarios.

4.1.2 Large neighbourhood search scheme

Large neighbourhood search (LNS) [17,14] is a local search meta-heuristic based on the observation that exploring a large neighbourhood, i.e., looking for a better solution by modifying a large portion of an existing one, tends to yield solutions

[3] This is a sensible heuristic because, barring routing aspects, the operation costs are roughly proportional to the size of the fleet, and in order to minimise it, we need to use larger vehicles.

of higher quality than exploring a small neighbourhood, albeit at a higher computational cost. In order to make exploration efficient, LNS has often [12] been coupled with filtering techniques, such as constraint propagation, which allows us to contain the neighbourhood size by removing low quality or unfeasible solutions.

In our approach, we employ the above CP model as a propagation engine, and we use a branch & bound tree-search strategy to explore the neighbourhood of the incumbent. Similar approaches have been applied successfully to tackle several vehicle routing problem variants [12,15,13].

Initial solution. In order to bootstrap the search, we need an initial solution. In our approach, we generate one using a branch & bound tree-search where the variable and value selection strategies are the ones of Branching 1.

Iterative improvement. Once the first solution is generated, the search enters a refinement loop, which alternates two steps. First, a *destroy* (or *relax*) procedure relaxes the incumbent solution by unassigning a subset of the decision variables, and propagating the constraints until a fix point is reached. After the relaxation, we are left with a smaller problem, which is expectedly easier to solve. Second, a *repair* procedure, consisting of a branch & bound tree-search using Branching 2 as variable and value selection strategy, re-optimises the newly freed variables yielding a new solution. Since the cost of the solution is bounded by the cost of the incumbent as described in Equation 1 only improving neighbouring solutions are explored. Since the re-optimisation is a branch & bound tree-search, and because constraints are propagated at each decision point, the search is able to keep the neighbourhood exploration small.

Our destroy phase employs two orthogonal relaxation schemes, which are chosen with coin-toss probability, and parametrised with a parameter δ, the *destruction rate*, representing the intensity of the relaxation

Relax δ days. all the decision variables (of all vehicles) concerning δ days chosen uniformly at random are set to their original domain, while the fleet is kept fixed,

Relax δ vehicles. all the decision variables (across all days) related to δ vehicles, chosen uniformly at random between the ones currently used in the solution plus the first unused vehicle, are set to their original domain.

The parameter δ represents the *adaptive* part of our LNS procedure. It is set to 1 at the beginning of each refinement phase, i.e., we try to relax either one day or one vehicle, and it is incremented by one as soon as a certain number of iterations, controlled by the parameter $iter_{max}$ have been spent on the same value of δ without finding any solution. Once an improving solution is finally found, the refinement loop restarts with the new solution, and δ is re-set to 1. If not improving solution are found, but δ has exceeded the maximum number of days or vehicles, the search is restarted from a new initial solution (but retaining the incumbent solution).

In principle, the repair step could be run until the optimal solution of the reduced problem is found. In practice, experiments showed that using a time limit is beneficial. In our setup, each repair step is given a time limit of $n_{free} \cdot t_{var}$ milliseconds, where n_{free} is the number of variables relaxed by the destroy step, and t_{var} (time per variable) is a parameter of the solver.

4.1.3 Mixed integer programming formulation

Our MIP formulation is a multi-day, and flexible fleet extension of the formulation presented by Archetti and Speranza in [1]. For presentation convenience we define the set of *legs* between the customers $c \in \mathbf{C}_d$ as $\mathbf{E}_d = \{(i,j) \mid i,j \in \mathbf{C}_d, i \neq j\}$.

Variables. A set of horizon-wide decision variables define the design of the fleet. First, vehicles are associated to types through the binary variables

$$z_{v,t} = \begin{cases} 1 & \text{if vehicle } v \text{ is of type } t \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in \mathbf{V}, t \in \mathbf{T}. \tag{26}$$

Moreover, the binary variables u_v (*used*) model whether a given $v \in \mathbf{V}$ is part of the *overall* fleet or not.

We now define the variables for the daily routing and loading problems. The binary variables $w_{d,v}$ model whether a vehicle $v \in \mathbf{V}$ is used on a given day $d \in D$ or not. Routes are defined, again, through binary variables

$$x_{d,v,i,j} = \begin{cases} 1 & \text{if } v \text{ travels over } (i,j) \text{ on } d \\ 0 & \text{otherwise} \end{cases} \quad \forall d \in D, v \in \mathbf{V}, (i,j) \in \mathbf{E}_d. \tag{27}$$

Finally, integer activity variables $y_{d,v,c,k}$ model how much commodity $k \in \mathbf{K}$ is dropped at each $c \in \mathbf{C}_d$ by a given vehicle $v \in \mathbf{V}$ on a day $d \in D$.

Constraints. We now describe the constraints of the MIP model. Constraint 28 ensures that each vehicle has one and only one type

$$\sum_{t \in \mathbf{T}} z_{v,t} = 1, \quad \forall v \in \mathbf{V}. \tag{28}$$

Constraints 29 and 30 establish the relation between u and w variables, stating that a vehicle is considered in the fleet if and only if it is used on some day

$$u_v \leq \sum_{d \in D} w_{d,v}, \quad \forall v \in \mathbf{V} \tag{29}$$

$$w_{d,v} \leq u_v, \quad \forall v \in \mathbf{V}, d \in D. \tag{30}$$

Constraint 31 states that a vehicle $v \in \mathbf{V}$ is considered used on some day $d \in D$ if it travels some edge on that day.

$$w_{d,v} \leq \sum_{(i,j) \in \mathbf{E_d}} x_{d,v,i,j}, \ \forall v \in \mathbf{V}, d \in D \tag{31}$$

Symmetrically, Constraints 32 and 33 ensure that a vehicle can travel on some edge on day $d \in D$ only if it is marked as used on that day.

$$\sum_{(i,j) \in \mathbf{E}_d} x_{d,v,i,j} \leq w_{d,v}, \quad \forall i \in \mathbf{C}_d, v \in \mathbf{V}, d \in D \tag{32}$$

$$\sum_{(j,i)\in\mathbf{E}_d} x_{d,v,j,i} \leq w_{d,v}, \quad \forall i \in \mathbf{C}_d, v \in \mathbf{V}, d \in D. \tag{33}$$

Constraint 34 is a flow conservation constraint: a vehicle reaching a location must always leave the location.

$$\sum_{(i,j)\in\mathbf{E}_d} x_{d,v,i,j} = \sum_{(j,i)\in\mathbf{E}_d} x_{d,v,j,i}, \quad \forall i \in \mathbf{C}_d, v \in \mathbf{V}, d \in D. \tag{34}$$

Constraints in 35 are *subtour elimination constraints*. Since the number of such constraints grows exponentially in the size of customers, we do not consider them from the beginning. Instead, we perform the search in the under-constrained search space, and look for subtours in the solutions as they are generated. When a subtour is found, we dynamically add the relative subtour elimination constraint as a cut. This is a standard technique to deal with large scale vehicle routing problems.

$$\sum_{i,j\in\mathbf{S},\, i\neq j} x_{d,v,i,j} \leq |\mathbf{S}| - 1, \quad \mathbf{S} \subseteq \mathbf{C}_d, |\mathbf{S}| \geq 2, \forall v \in \mathbf{V}, d \in D. \tag{35}$$

Constraint 36 states that a vehicle $v \in \mathbf{V}$ can move some commodity $k \in \mathbf{K}$ on day $d \in D$ at a customer $i \in \mathbf{C}_d$ (represented by activity variables $y_{d,v,i,k}$) only if it visits it.

$$y_{d,v,i,k} \leq \mathbf{q}_{d,i,k} \sum_{(j,i)\in\mathbf{E}_d} x_{d,v,j,i}, \quad \forall i \in \mathbf{C}_d, v \in \mathbf{V}, d \in D, k \in \mathbf{K}. \tag{36}$$

Constraints 37 and 38 ensure that the capacity of the vehicles and the compatibility with the commodity being transported are always respected.

$$\sum_{i\in\mathbf{C}_d, k\in\mathbf{K}} y_{d,v,i,k} \leq \sum_{t\in\mathbf{T}} \mathbf{b}_t \cdot z_{v,t}, \quad \forall v \in \mathbf{V}, d \in D \tag{37}$$

$$\sum_{i\in\mathbf{C}_d} y_{d,v,i,k} \leq \sum_{t\in\mathbf{T}} \mathbf{b}_t \cdot z_{v,t} \cdot \mathbf{comp}_{t,k}, \quad \forall k \in \mathbf{K}, v \in \mathbf{V}, d \in D \tag{38}$$

In Constraint 39, we require the demand of each customer to be satisfied (solutions that don't completely satisfy the demand are considered unfeasible).

$$\sum_{v\in\mathbf{V}} y_{d,v,i,k} \cdot w_{d,v} = \mathbf{q}_{d,i,k}, \quad \forall k \in \mathbf{K}, i \in \mathbf{C}_d, d \in D \tag{39}$$

Finally, consistently with the problem formulation, the objective function minimises the total operation costs of the fleet.

$$\textbf{minimise} \sum_{d\in D, v\in\mathbf{V}, (i,j)\in\mathbf{E}_d, t\in\mathbf{T}} x_{d,v,i,j} \cdot z_{v,t} \cdot \mathbf{w}_t \cdot \mathbf{dist}_{l_i,l_j}$$

Some of the constraints in this formulation contain non-linear terms, e.g., $y_{d,v,l,k} \cdot u_{d,v}$. However, since one of the variables involved in such terms is always binary, we can decompose such constraints so that the model only contains linear terms.

When using the model to solve a daily problem with a fixed fleet, we introduce additional constraints to fix the values of the z and the w variables so that they match the existing fleet. Since these constraints are rather trivial we do not report them here.

4.2 Pre-processing

While our models can tackle the above multi-day formulation, running it for very large horizons, e.g., one year, can be impractical. The particular nature of our problem, allows us to do better. Using a Pareto dominance-based approach, we identify representative subsets of the historical data, and use those in place of the whole horizon. To accomplish this, we represent each day d in the horizon with the total amount of ambient and chilled demand, i.e., $(\mathbf{q}_{d,amb}, \mathbf{q}_{d,chi})$, and consider these as objectives to be maximised. For instance, a day with a total demand vector of $(30, 40)$ Pareto-dominates a day with demand $(20, 30)$, but it is incomparable to a day with demand $(40, 30)$. Figure 2 shows the annual distribution of the total daily demand.

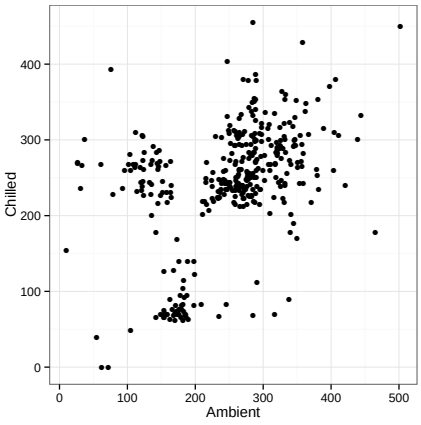

Fig. 2: Distribution of daily ambient and chilled demand over one year.

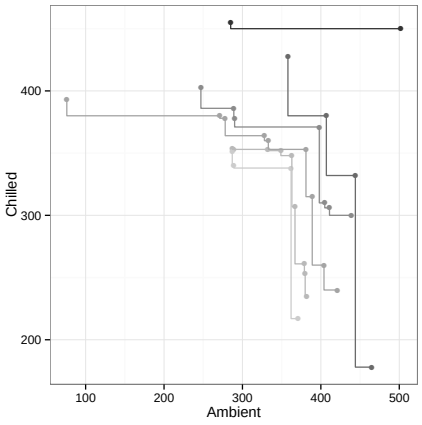

Fig. 3: The first six Pareto fronts of the ambient and chilled demand.

The idea behind our approach is that, if the demand on a day p dominates the demand on a day q, then the optimal fleet for p is also guaranteed to be able to serve q without requiring additional vehicles[4]. By extending this reasoning to the multi-day perspective, we can state that a fleet which solves the multi-day problem based on the Pareto front is also guaranteed to be able to solve all the other days in the horizon.

We tested this approach by computing the Pareto front based on 364 days of historical order data. The identified front is composed by only two days, which greatly dominate the whole horizon. We identified the optimal fleet for such front, and then we tested the found fleet on each day of the horizon[5]. These preliminary

[4] This is only true because the considered problem does not include time constraints or limits on the length of the routes; the carrying capacity of a fleet determines completely the set of days that can be covered.

[5] In our setup, we always run our experiments until a given time limit, which depends on the task at hand. To optimise a fleet we let the solver run for 1 hour, since the problem is quite large, and the quality of the solution has a long lasting effect. For the optimisation of the daily

experiments revealed that, while the identified fleet guarantees a coverage of 100% of the instances (we call this the *conservative fleet*), almost half of the vehicles were unused, on average, during a typical day, suggesting that the conservative fleet had a poor efficiency. For this reason we decided to adopt a *layered* Pareto dominance approach, in which multiple Pareto fronts, with decreasing degrees of coverage (but increasing levels of efficiency), are generated. Algorithm 1 outlines this procedure.

Algorithm 1 Pre-processing step

```
procedure LayeredPareto(I)
    F ← ∅; P ← ∅; D ← ∅
    while P ≠ ∅ do
        C ← ∅
        for d = (d, q_{d,amb}, q_{d,chi}) ∈ P do
            for p = (p, q_{p,amb}, q_{p,chi}) ∈ C do
                if p ≻ d then
                    D ← D ∪ {d}
                else if p ≺ d then
                    D ← D ∪ {p}
                end if
            end for
            C ← C/D
            if d ∉ D then
                C ← C ∪ {d}
            end if
        end for
        P ← D; D ← ∅, F ← F ∪ {C}
    end while
    return F
end procedure
```

The algorithm accepts a set **I** of instances, and returns a set of layered Pareto fronts. The procedure relies on a number of queues and sets: **F** represents the set of Pareto fronts generated so far, **P** represents the set of days that need to be processed, **C** is the current Pareto front being built, and **D** is the set of days that are dominated by some of the days in the current Pareto front (and that will constitute the set of days to be processed in the next iteration). The procedure iteratively splits the dominated days from the non-dominated days, and builds a Pareto front with the non-dominated ones. Then it proceeds by identifying the next Pareto front using only the dominated ones. The procedure stops when there are no more days to process, i.e., all days have been added to some front.

Obviously, this is not necessarily the only way to generate different levels of coverage and efficiency. For instance, one could start from a Pareto front, remove one of its points from the set, and recompute a new "weaker" Pareto front, thus obtaining all possible coverage levels. For our purposes, however, the layered Pareto approach is convenient, as it provides us with a tractable number of alternatives within the range of coverage values we are interested to (90% to 100%), without generating a potentially large number of alternative fronts.

problems, which are much smaller, we allot 15 minutes, after which little improvement typically occurs. A more exhaustive discussion of our experimental setup is presented in Section 5.

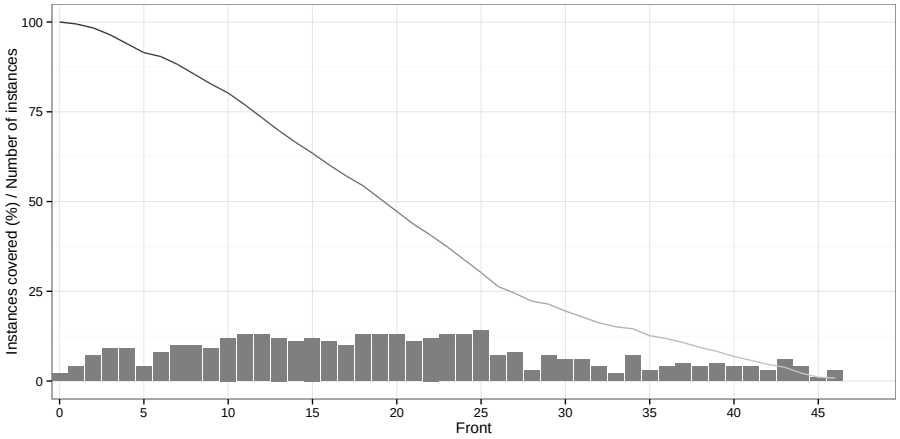

Fig. 4: Instance coverage and number of days per front.

Starting from the 364 days, the layered approach identifies 46 Pareto fronts with decreasing instance coverage. The number of days in each fronts has a median of 7 and a maximum of 14. This, of course, makes the problem much more tractable than considering the whole horizon. In Figure 3, we highlight the first six fronts, which contain less than 10 days each, and dominate 100% (front 0, upper right) to 90% (front 5, lower left) of the days. An instance coverage of 90% means that a fleet that can solve the front is *guaranteed* to solve at least 90% of the instances, but it could *possibly* solve more (see Table 3 for the actual coverage obtained by the fronts on the full data set). In all the other days some vehicles need to be hired. Note that there are economical reasons for choosing a smaller fleet over a large fleet, even taking into account the cost of hiring on-demand vehicles which, however, we do not consider in this context. The plot in Figure 4 reports, for every front, the percentage of days its fleet is guaranteed to cover, and (as bar plots) the number of days in the front.

5 Experimental analysis

We implemented our CP model and LNS scheme in Gecode 4.4.0 [16], and our MIP model in Gurobi 6.0.4 [11]. We then carried out an experimental analysis to compare the quality of the presented approaches, and to identify the best fleet for each of the first six fronts.

In the following, all the experiments on the multi-day problems (aimed at identifying a fleet) are run with a time limit of 1 hour, validation experiments on the single-day problems for 15 minutes, and parameter tuning experiments for 1 minute. Each experiment is run in single-threaded mode; this does not advantage any of the three approaches, since each one of them supports, and can benefit from, parallelism.

5.1 Problem instances

The problem instances considered in our experimental analysis represent the same 364 days used to identify the Pareto fronts. The instances have a median of 19 customers per day, and a median total demand of 518 units of product, of which about 49% chilled. We call these instance *large*, and all the reported results are based on these, as they represent the real data.

To analyse the scaling properties of our approach, and to perform a more principled parameter tuning, we also considered two *clustered* variants of the above instances, in which customers close to each other are collapsed, and their demand is aggregated. The instances dubbed *medium* have a median of 16 customers per day, while the *small* instances have a median of 5 customers per day.

5.2 Preparatory steps

The behaviour of LNS described in 4.1.2 is controlled by two parameters, $iter_{max}$ and t_{var}, that have to be set by the user. In order to find a suitable setup for these parameters we ran an automated F-Race [3] with confidence level 95% using JSON2RUN [19]. The tuning involved the *small* class of instances, and all the 18 combinations of the parameters $iter_{max} \in \{10, 20, 50\} \times t_{var} \in \{1, 2, 5, 10, 20, 50\}$; each experiment was run on a 2.9 GHz Intel Core i7 machine with 8 GB of RAM. The parameter ranges were selected through a preliminary screening of the parameter space. F-Race identified $iter_{max} = 20$ and $t_{var} = 1$ millisecond as the ideal setup for the considered training set.

We then tested each approach, i.e., CP (branch & bound), LNS, and MIP, on the full set of single-day instances. The goal of this analysis was to find a method with suitable scaling properties, that could be used to effectively tackle the multi-day problem. Table 2 reports the average solution costs aggregated by instance size and yearly quarter. Note that here we are not necessarily reporting optimal solutions, just the best solution found within the time limit. Moreover, the average solution costs are computed by ignoring all the instances for which not all of the solvers could find a solution. This clearly favours the MIP solver, which was unable to find a solution on several medium and large instance. For this reason, we also report the percentage of solved instances, which must be considered for fair comparison.

From the results, it is easy to see that both CP (branch & bound) and LNS achieve complete coverage, even on the *large* instances, which represent the actual historical data. In general LNS reports better performance than pure branch & bound, sometimes obtaining daily costs a few thousands dollar lower. The MIP solver scores better than both CP and LNS, at least with respect to quality, on the *small* instances, however fails to generate even a single feasible solution in some cases. As the size of the instances grows, the instance coverage of the MIP solver drops significantly ($< 30\%$), making it unsuitable to be used in the next step.

A second set of experiments, not reported here, reveals that the MIP solver can become competitive, and even outperform LNS, when a fixed fleet is used. This suggests that, while this specific mathematical programming formulation is not suitable for finding good fleet designs, it represents a good choice for solving the daily problem once the fleet is fixed. Furthermore, separating the fleet mix aspects

Size	Qtr.	CP		LNS		MIP	
		f	%	f	%	f	%
Small	Q1	53384.21	**100.00**	49511.26	**100.00**	**49272.46**	97.80
	Q2	52994.07	**100.00**	49046.25	**100.00**	**48807.93**	98.91
	Q3	52370.42	**100.00**	48615.01	**100.00**	**48400.14**	98.88
	Q4	53198.69	**100.00**	49244.21	**100.00**	**48979.94**	**100.00**
Medium	Q1	45652.74	**100.00**	**41864.71**	**100.00**	44835.24	32.97
	Q2	46023.11	**100.00**	**41747.70**	**100.00**	45444.73	29.35
	Q3	46271.52	**100.00**	**41391.28**	**100.00**	44454.63	29.21
	Q4	46904.61	**100.00**	**42532.34**	**100.00**	45804.38	23.91
Large	Q1	45221.61	**100.00**	**40769.25**	**100.00**	43690.42	26.37
	Q2	47271.52	**100.00**	**42676.57**	**100.00**	45785.29	17.39
	Q3	43128.69	**100.00**	**38226.30**	**100.00**	42026.75	15.73
	Q4	48154.71	**100.00**	**43561.53**	**100.00**	46209.25	17.39

Table 2: Summary of approach quality and instance coverage.

of the problem from the routing and loading aspects, e.g., by using a Benders' decomposition approach, is a compelling research direction which could allow a MIP formulation to tackle the overall problem. Other than performance, however, there are other important aspects in the choice of an optimisation technique, first and foremost maintainability and extensibility. In this regard, the combination between CP and LNS offers a trade-off between extensibility and performance, which is hard to challenge.

All in all, considering instance coverage and solution quality, we decided to tackle the multi-day extension of the problem using the large neighbourhood search solver.

5.3 Fleet generation

We ran the large neighbourhood solver on the multi-day instances obtained by aggregating the *large* instances according to the Pareto fronts identified in the pre-processing phase. In these experiments, run in flexible fleet mode, we used a maximum number of vehicles of $V = 25$. All the fleet generation experiments were run on a cluster of 3.1 GHz AMD Opteron nodes with 64 GB of RAM each and allotted 1 hour of time.

The identified fleets differ in size (see Table 3), however share the same composition: the only employed vehicles are the largest non-refrigerated and refrigerated trucks, respectively B-Triples and A-Double Reefers.

Once identified the best fleets, we validated each one by using it to solve all the daily instances of the *large* family. All the validation experiments were run on a cluster of 3.1 GHz AMD Opteron nodes with 64 GB of RAM each and allotted 15 minutes of time.

The results of our analysis are summarised in Table 3, which reports, for every Pareto-based fleet, *i)* the number of vehicles (V), *ii)* the average daily operations cost (Cost_d), *iii)* the predicted instance coverage, the actual instance coverage, *iv)* the difference in daily cost between using the Pareto-based fleet and the *ideal fleet* for each day ($\text{Cost}_\%$), and *v)* the average utilisation of vehicles on each day

F	V	Cost_d	Pred. coverage	Actual coverage	$\text{Cost}_\%$	Usage
0	22	51.71k	100.00%	100.00%	8.60%	52.92%
1	19	52.24k	99.45%	99.73%	9.88%	62.37%
2	18	51.71k	98.35%	98.90%	8.69%	64.74%
3	17	51.79k	96.43%	97.80%	8.78%	68.55%
4	17	52.22k	93.96%	97.25%	9.86%	69.70%
5	17	52.66k	91.48%	96.98%	10.92%	70.50%

Table 3: Fleets obtained for the first six fronts.

(Usage). Note that the cost of the ideal fleet for each day is a measure of the absolute best we can do, and represents an goal rather than a viable alternative. In other words, obtaining a 0% cost increase over the ideal fleet might not be possible using a single fleet to cover the whole horizon. Again, as we did for Table 2, we compute the average costs based exclusively on those instances for which both the Pareto-based fleet and the ideal fleet for each day obtained a feasible solution. This measure, considered together with the instance coverage, allows us to draw a fair comparison.

From our results emerges that, by optimising the usage of resources, it is possible to maintain a good instance coverage despite the reduction in fleet size. A metric that provides insight on the quality of a fleet, in this regard, is the average daily utilisation of vehicles. Ideally, good fleets tend to have a high daily utilisation rate, i.e., on average they leave few vehicles unused. Conversely, fleets with a smaller difference in cost with respect to the ideal fleet are better, because they are less affected by the fact that they need to work consistently across the whole horizon. For instance, with an average operation cost of about 9% above the one of the ideal fleet, the fleet based on the 2^{nd} Pareto front achieves an instance coverage of almost 99% using 4 vehicles less than the conservative fleet. Clearly, this comparison does not tell the whole story. Acquisition costs, together with depreciation, and the costs for hiring vehicles to cover the remaining demand on the 1.1% of the days which cannot be fully covered, are not considered. While it is reasonable to believe that the additional costs can be easily absorbed by the savings in the maintenance and acquisition costs obtained by using a more streamlined fleet, in order to provide a quantitative picture, these costs must be explicitly included in the objective function. For a variety of reasons, these aspects are currently not being considered in our approach, however they represent a top priority for the next stages of the project.

6 Conclusions

We proposed *i)* a new constraint programming model for the multi-day, multi-commodity, fleet mix and routing problem, *ii)* a large neighbourhood scheme to produce daily plans and efficient fleet designs using the proposed model as a propagation engine, *iii)* a mixed integer programming formulation of the same problem, and *iv)* a pre-processing phase that allows us to use representative subsets of instances to predict the best fleet based on the historical demand data. The principal novelty of our approach lies in how we use a solver for short term operational deci-

sions (routing and loading) to inform the mid-term tactical decisions, and on our pre-processing technique.

We have shown experimentally that our approach is a suitable methodology to inform fleet-related mid-term decisions. Our experimental analysis also considered an alternative MIP formulation. Our experiments revealed that the LNS has better scaling properties with respect to the proposed MIP formulation, however the block structure of the problem suggests that using an appropriate decomposition might significantly change the picture.

Our approach has also a number of limitations, which must be addressed in order to be able to apply it outside of our specific case study. First, our pre-processing phase relies on the fact that instance coverage, and thus representative subsets of instances, can be deterministically calculated based on the total demand. This is not necessarily true if side constraints, and in particular time constraints, are added to the mix. While, thanks to the flexibility of constraint programming, it is relatively easy to add such constraints to our model, the lack of an effective pre-processing phase might lead to arbitrarily long horizons, which are supported but computationally challenging. Second, our approach does not take into account the costs of modifying the fleet, i.e., acquiring and decommissioning vehicles, and hiring on-demand vehicles. All these aspects must be considered in order to provide a holistic view to the client. Third, we have not explored alternative meta-heuristic techniques, which are well known to be successful on large-scale optimisation problems. Among these local-search is a particularly compelling research avenue. We plan to address these aspects in future iterations of this project.

References

1. Archetti, C., Speranza, M.: The split delivery vehicle routing problem: A survey. In: B. Golden, S. Raghavan, E. Wasil (eds.) The Vehicle Routing Problem: Latest Advances and New Challenges, *Operations Research/Computer Science Interfaces*, vol. 43, pp. 103–122. Springer US (2008)
2. Belfiore, P., Yoshizaki, H.T.: Heuristic methods for the fleet size and mix vehicle routing problem with time windows and split deliveries. Computers & Industrial Engineering **64**(2), 589–601 (2013)
3. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-Race: An overview. In: Experimental methods for the analysis of optimization algorithms, pp. 311–336. Springer (2010)
4. Bräysy, O., Dullaert, W., Hasle, G., Mester, D., Gendreau, M.: An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows. Transportation Science **42**(3), 371–386 (2008)
5. Crainic, T.G.: Long-haul freight transportation. In: R.W. Hall (ed.) Handbook of Transportation Science, *International Series in Operations Research & Management Science*, vol. 56, pp. 451–516. Springer US (2003)
6. Dell'Amico, M., Monaci, M., Pagani, C., Vigo, D.: Heuristic approaches for the fleet size and mix vehicle routing problem with time windows. Transportation Science **41**(4), 516–526 (2007)
7. Di Gaspero, L., Rendl, A., Urli, T.: Balancing bike sharing systems with constraint programming. Constraints pp. 1–31 (2015)
8. Di Gaspero, L., Urli, T.: A cp/lns approach for multi-day homecare scheduling problems. In: Hybrid Metaheuristics, pp. 1–15. Springer (2014)
9. Dror, M., Trudeau, P.: Savings by split delivery routing. Transportation Science **23**(2), 141–145 (1989)
10. Dror, M., Trudeau, P.: Split delivery routing. Naval Research Logistics (NRL) **37**(3), 383–402 (1990)
11. Gurobi Optimization, I.: Gurobi optimizer reference manual (2015). URL `http://www.gurobi.com`
12. Kilby, P., Shaw, P.: Vehicle routing. Handbook of constraint programming pp. 799–834 (2006)
13. Kytöjoki, J., Nuortio, T., Bräysy, O., Gendreau, M.: An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Computers & Operations Research **34**(9), 2743–2757 (2007)
14. Pisinger, D., Ropke, S.: Large neighborhood search. In: Handbook of Metaheuristics, pp. 399–419. Springer (2010)
15. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation science **40**(4), 455–472 (2006)
16. Schulte, C., Tack, G., Lagerkvist, M.Z.: Modeling and programming with gecode. In: C. Schulte, G. Tack, M.Z. Lagerkvist (eds.) Modeling and Programming with Gecode (2015)
17. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: M. J. Maher, J.F. Puget (eds.) CP'98: the 4th International Conference on Principles and Practice of Constraint Programming, 1998, Proceedings of, *Lecture Notes in Computer Science*, vol. 1520, pp. 417–431. Springer (1998)
18. Toth, P., Vigo, D.: The Vehicle Routing Problem. Society for Industrial and Applied Mathematics (2002)
19. Urli, T.: json2run: a tool for experiment design & analysis. arXiv preprint arXiv:1305.1112 (2013)
20. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research **234**(3), 658–673 (2014)