A heuristic branch-cut-and-price algorithm for the ROADEF/EURO challenge on Inventory Routing

Nabil Absi, Diego Cattaruzza, Dominique Feillet, Maxime Ogier, Frédéric Semet

September 21, 2017

A heuristic branch-cut-and-price algorithm for the ROADEF/EURO challenge on Inventory Routing

Nabil Absi[1], Diego Cattaruzza[2], Dominique Feillet[1], Maxime Ogier[2], Frédéric Semet[2]

[1]: Mines Saint-Etienne and LIMOS UMR CNRS 6158, CMP Georges Charpak, F-13541 Gardanne, France.
absi,feillet@emse.fr

[2]: Univ. Lille, CNRS, Centrale Lille, Inria
UMR 9189 - CRIStAL
Centre de Recherche en Informatique Signal et Automatique de Lille
F-59000 Lille, France
cattaruzza,ogier,semet@centralelille.fr

Abstract

This paper is part of the special section devoted to the ROADEF/EURO challenge on Inventory Routing. We propose an extended formulation that we address with a heuristic branch-price-and-cut method. Among the difficulties, that we had to face, are : a fractional objective function, the simultaneous generation of constraints and columns, and a complex pricing problem. We evaluate our approach on the benchmark instances proposed for the challenge.

1 Introduction

In this paper, we propose a heuristic solution method for the Inventory Routing Problem introduced during the 2016 ROADEF/EURO challenge, which we coin as REC-IRP. Inventory routing has attracted researchers for many years due to both its practical and theoretical issues (Coelho et al.; 2013). Many different variants have been investigated in the literature. The REC-IRP proposed for the challenge is original and particularly complex for several reasons. Among them, we can cite the logistic ratio optimization objective, the hourly time-granularity for inventory constraints or the driver/trailer allocation management. Designing an exact solution approach is out of reach for large size instances as those proposed during the challenge. However, we decided to address the REC-IRP with a branch-cut-and-price framework: a cut-and-column generation procedure is developed,

along with a heuristic pricing algorithm to generate new columns and a heuristic fixing procedure to generate integer solutions.

Several modeling opportunities were opened when deciding what should be a *column* in our approach. We decided that a column would be what we call a *td-shift*, namely, a trailer-driver-shift. A *td*-shift is an elementary piece of work for a driver (the driver is known), that starts with a specific trailer from the base (the trailer is known), visits a subset of customers and returns at the base, with possibly intermediate stops at one or several sources to refill the trailer. The *td*-shift is time-stamped, meaning that the timing of all operations is fixed. Conversely, the quantities delivered to the customers are unknown.

Let us define a trip as a sequence of customers starting from and ending at the base or a source. Because of the multi-trip dimension of shifts, one could have decomposed shifts in trips and defined columns as trips. These two options (shifts or trips) have been compared in Hernandez et al. (2016) for the Multi-trip Vehicle Routing Problem with Time Windows. The theoretical advantage of a column generation approach based on trips is to limit the combinatorial explosion at the pricing problem level (a trip contains fewer customers than a shift). The counterpart is a complex master problem where one has to select trips but also to combine them into shifts, and eventually a more complex implementation of the branch-and-price framework. Experimental results in Hernandez et al. (2016) do not permit to conclude definitely on the relative efficiency of the two approaches. For this reason, and given the challenging problem in hand, we decided for the easiest one.

A key feature of the problem concerns the quantities delivered during the shifts. In this respect, the problem is very similar to the Split Delivery Vehicle Routing Problem (SDVRP), where partial deliveries to customers are feasible. An important decision here is to decide whether the quantities delivered should be fixed in the column definition (pricing problem level) or decided at the master problem level. The two alternatives have been considered in Feillet et al. (2006) and Desaulniers (2010), respectively. The clear conclusion in the context of the SDVRP with Time Windows is that deciding on the quantities at the pricing problem level is better. However, the interaction between quantities delivered in different routes and for different customers seems stronger in inventory routing, and it is not clear whether such a conclusion can be generalized to this context. In the inventory routing literature, both options have been analyzed. Desaulniers et al. (2015) extends the approach developed in Desaulniers (2010). Bard and Nananukul (2010) follows a similar approach while solving the pricing problem as an integer program. Le et al. (2013) or André et al. (2016) decide of the quantities at the master problem level. We decided to follow this latest scheme. Note also that some authors address easier cases where the delivered quantities are limited to a few possible discrete values (Christiansen and Nygreen; 2005; Michel and Vanderbeck; 2012). This could also have been an option for us in a heuristic approach.

A last important characteristic of the problem, that may *a priori* significantly influence the solution scheme is the logistic ratio objective. This objective is naturally written as a fractional function of the decision variables. We are only aware of a few implementa-

tions of the branch-and-price methodology for mathematical formulations with fractional objectives. Garaix et al. (2011) implemented and compared two approaches, adapting to the column generation framework algorithms presented in Charnes and Cooper (1962) and Dinkelbach (1967). Initially they developed their research works in the context of mathematical programming with fractional objective functions. In short, Charnes and Cooper's approach relies on changes of variables. Dinkelbach's algorithm expresses the objective as a weighted linear combination of the numerator and the denominator, and solves the resulting program repeatedly while updating the weights (see more details in Section 3.4). Computational experiments in Garaix et al. (2011) show a similar efficiency for both methods. The authors concluded that having a fractional objective is not particularly detrimental to the solution of the problem. In this paper, we decided to use Dinkelbach's approach.

Recently, in Archetti et al. (2016) and André et al. (2016), the first column generation approaches for inventory routing problems with logistic ratio objectives were proposed. In both cases, the authors tackled a standard Inventory Routing Problem (apart from the objective) with the aim to develop an exact solution method. In particular, the additional difficulties described above are not present (even if André et al. (2016) progressively includes features of the REC-IRP). Archetti et al. (2016) apply a Dinkelbach's scheme. At each step, the linear program is solved using the method developed in Desaulniers et al. (2015). As mentioned above, delivered quantities are fixed in column definitions. Only small instances, including at most 15 customers and defined on a time horizon limited to at most 5 periods, are solved to optimality. Contrary to what could be observed in Garaix et al. (2011), the fractional objective highly complicates the problem. The reason identified by the authors is that the value of the objective function tends to be flat around the optimal solution. As a consequence, there exist many solutions with objective function values close to the optimal one. This creates difficulties in the solution algorithm as it is much harder to prune branch-and-bound nodes. Paradoxically, this might rather be helpful for a heuristic solution of the problem. André et al. (2016) use Charnes and Cooper's method. The column definition is very similar to ours. Their method allows finding good lower and upper bounds on the same instances as those used in Desaulniers et al. (2015) and on a new instance set including simplified Air Liquide instances of the same size.

The paper is organized as follows. Section 2 introduces the notation. In section 3, we present our mixed integer linear programming formulation. Section 4 explains why a simultaneous cut-and-column generation strategy is needed and how it is carried out. In Section 5, we propose what we call an implicit-relaxation based dynamic programming algorithm, that constructs a dual vector from an implicit linear relaxation of the master problem, and price columns according to this vector. Section 6 describes the increasing horizon heuristic, that we developed, based on the formulation we proposed. Computational results are presented in Section 7, before concluding with Section 8.

2 Problem description and notation

This paper being part of a special issue devoted to the REC-IRP, we omit the complete description of the problem. Readers are referred to the first paper of the special issue to find this description. However, the paper is organized so that it should be possible to read and understand it independently. In particular, the notation is entirely reintroduced, and all constraints are progressively introduced and explained.

Tables 1 and 2 provide a first round of notation. Note that (contrary to the convention taken in the problem description) we introduce a customer in $\mathcal{V}^2$ for each order placed by a call-in customer. In the following, we, however, call these orders *customers*. Especially, we call customer set the set $\mathcal{V} = \mathcal{V}^1 \cup \mathcal{V}^2$. Note also that the time horizon is discretized in one-hour periods, that starts at period 0 and ends at period H: $\mathcal{H} = \{0, \ldots, H\}$. The term period in the paper will always refer to this time granularity of one hour. When data or variables are expressed in minutes, it will be emphasized.

$\mathcal{H}$	set of one hour periods in the time horizon
$\mathcal{V}^1$	set of VMI customers
$\mathcal{V}^2$	set of call-in customer orders
$\mathcal{V}$	set of customers
$\mathcal{D}$	set of drivers
$\mathcal{T}$	set of trailers
$\mathcal{T}_d$	set of trailers that can be driven by driver $d \in \mathcal{D}$
$\mathcal{D}_t$	set of drivers that can drive trailer $t \in \mathcal{T}$
$\mathcal{TW}^d$	set of allowed time windows for driver $d \in \mathcal{D}$

Table 1: Notation (sets)

H	last period in set $\mathcal{H}$
$rest_d$	minimal resting time between two shifts for driver $d \in \mathcal{D}$
Q_t	vehicle capacity for trailer $t \in \mathcal{T}$
q_i^{min}	minimal quantity delivered to customer $i \in \mathcal{V}^1$ each time it is visited
q_i^{min}	accumulated minimal quantity delivered to customer $i \in \mathcal{V}^2$
q_i^{max}	accumulated maximal quantity delivered to customer $i \in \mathcal{V}^2$
$stock_0^i$	initial stock for customer $i \in \mathcal{V}^1$
$stock_{max}^i$	maximal stock for customer $i \in \mathcal{V}^1$
q_t^{init}	initial load in trailer $t \in \mathcal{T}$
d_h^i	demand of customer $i \in \mathcal{V}^1$ at period $h \in \mathcal{H}$

Table 2: Notation (additional data)

3 Mathematical formulation

We propose to formulate the REC-IRP with an extended formulation based on the selection of *td*-shifts. We introduce $\mathcal{R}$, the set of time-stamped shifts. By time-stamped shift, we mean that the physical itinerary followed during the shift and the time of all the events of the shift are known. Conversely, the quantities delivered to customers, the trailer used or the driver assigned to the trailer are not known. Note that the concept of *td*-shift, introduced in Section 1, will only be useful when describing the column generation procedure (see Section 4). However, it is noteworthy that a *td*-shift is obtained by assigning a trailer and a driver to a time-stamped shift. From now on, time-stamped shifts will be called shifts for short.

To limit the complexity of the model, we make some assumptions on set $\mathcal{R}$:

1. each customer is visited at most once in a shift;
2. a shift lasts at least the duration of a period, *i.e.*, it cannot start and end at the same period;
3. a trailer cannot have more than one visit to a source at a given period;
4. a shift is carried out at most once.

Assumption 1 makes sense in practice: it is certainly not desirable that a trailer would visit twice a customer in the same shift. This assumption should have a very small impact with respect to optimality on real-life instances. Assumption 2 and 3 are also expected to hold for most shifts, since a period is one-hour long. Actually, in the testbed used for the experiments, these assumptions are almost always satisfied (only three instances contain customers that can be delivered directly from the base within 54 minutes shift). Assumption 4 excludes the assignment of the same time-stamped shift to different drivers. This could be of interest in some specific situations, but, again, one can expect a very limited loss of optimality due to this assumption on real-life instances. We have to stress that Assumption 4 does not prevent from repeating the same tour at different times since a shift is time-stamped.

In the next two subsections, we introduce the notation associated with the shifts and the mathematical formulation.

3.1 Notation for shifts

Let us note $\mathcal{R}^d \subseteq \mathcal{R}$ the subset of shifts compatible with driver $d \in \mathcal{D}$. Shift r is compatible with driver d if: (1) shift r is included in one of the time windows associated with driver d; (2) the driving duration of shift r does not exceed the maximal driving duration allowed

for driver d; (3) if r is a layover shift, r respects the maximum layover duration associated with driver d.

We note $\mathcal{R}^t \subseteq \mathcal{R}$ the subset of shifts compatible with trailer $t \in \mathcal{T}$. A shift r is compatible with trailer t if: (1) all customers visited in shift r are compatible with trailer t; (2) all sources visited in shift r are compatible with trailer t; (3) for each trip in r, the sum of the minimal delivery quantities of the VMI customers served in the trip does not exceed the capacity of trailer t.

We note $\mathcal{R}^{dt}$ the subset of shifts that are simultaneously compatible with driver $d \in \mathcal{D}$ and with trailer $t \in \mathcal{T}$, *i.e.*, $\mathcal{R}^{dt} = \mathcal{R}^d \cap \mathcal{R}^t$.

We note c_{rdt} the cost of shift $r \in \mathcal{R}^{dt}$ when operated by driver d and trailer t. We note st_r the starting time of shift r, and et_r its ending time (in minutes). st_r and et_r are included in the time interval covered by the trip.

Notation $i \in r$ indicates that customer $i \in \mathcal{V}$ is visited by shift r. Given $i \in r$, we note st_{ri} the one-hour period during which the delivery of the commodity to customer i starts. Last, we define $\delta_{ihr} = 1$ if $i \in r$ and $st_{ri} = h$, 0 otherwise.

We introduce $\mathcal{T}rip_r$ the set of trips performed during in shift r. Recall that a trip is defined as a sequence of customer visits starting from/ending at a source and/or the base. We note $i \in s$ if customer $i \in \mathcal{V}$ is visited in trip s. We note $\mathcal{V}_s$ the subset of customers visited in trip s.

Notation introduced for shifts are reported in Table 3. To ease notation, we introduce additional subset notation reported in Table 4.

$\mathcal{R}^d$	subset of shifts compatible with driver d
$\mathcal{R}^t$	subset of shifts compatible with trailer t
c_{rdt}	cost of shift r when operated by driver d and trailer t
st_r	starting time of shift r (in minutes)
et_r	ending time of shift r (in minutes)
$\mathcal{V}_r$	subset of customers visited in shift r
st_{ri}	starting time of service for customer i in shift r (in hours)
δ_{ihr}	1 if $i \in r$ and $st_{ri} = h$, 0 otherwise
$\mathcal{T}rip_r$	set of trips in shift r
$\mathcal{V}_s$	subset of customers visited in trip s

Table 3: Notation introduced for shifts (part 1)

$\mathcal{R}^{dt}$	subset of shifts compatible with driver d and trailer t
$\mathcal{D}_r$	subset of drivers compatible with shift r
$\mathcal{T}_r$	subset of trailers compatible with shift r
$\mathcal{D}_{rt}$	subset of drivers compatible with shift r and trailer t
$\mathcal{T}_{rd}$	subset of trailers compatible with shift r and driver d
$\mathcal{V}_r^1$	subset of customer set $\mathcal{V}^1$ visited in shift r
$\mathcal{V}_r^2$	subset of customer set $\mathcal{V}^2$ visited in shift r
$\mathcal{V}_s^1$	subset of customer set $\mathcal{V}^1$ visited in trip s
$\mathcal{V}_s^2$	subset of customer set $\mathcal{V}^2$ visited in trip s
$\mathcal{R}_i$	subset of shifts that visit customer i

Table 4: Notation introduced for shifts (part 2)

3.2 Mathematical model

The REC-IRPcan be formulated as an integer linear program as follows. Let us first introduce four sets of decision variables:

$$\begin{cases} x_{rdt} \in \{0,1\} & (d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}^{dt}) \\ q_{irt} \geq 0 & (r \in \mathcal{R}, t \in \mathcal{T}_r, i \in \mathcal{V}_r) \\ q_{th} \geq 0 & (t \in \mathcal{T}, h \in \mathcal{H}) \\ S_h^i \geq 0 & (i \in \mathcal{V}_1, h \in \mathcal{H}) \end{cases}$$

x_{rdt} is a binary variable equal to 1 if shift r is assigned to driver d and trailer t, 0 otherwise. Variable q_{irt} indicates the quantity delivered to customer i in shift r using trailer t. Variable q_{th} is the load in trailer t at the end of period h. Variable S_h^i represents the inventory level at VMI customer $i \in \mathcal{V}^1$ at the end of period h.

We propose formulation (1)-(20), with new notation detailed next:

$$\text{minimize } \frac{\sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} \sum_{r\in\mathcal{R}^{dt}} c_{rdt} x_{rdt}}{\sum_{r\in\mathcal{R}} \sum_{t\in\mathcal{T}_r} \sum_{i\in\mathcal{V}_r} q_{irt}} \tag{1}$$

subject to compatibility constraints:

$$\sum_{r\in\mathcal{K}_m^d} \sum_{t\in\mathcal{T}_{rd}} x_{rdt} \leq 1 \qquad (d \in \mathcal{D}, m \in \mathcal{S}t(\mathcal{R}^d)), \quad \pi_m^d \tag{2}$$

$$\sum_{r\in\mathcal{K}_m^t} \sum_{d\in\mathcal{D}_{rt}} x_{rdt} \leq 1 \qquad (t \in \mathcal{T}, m \in \mathcal{S}t(\mathcal{R}^t)), \quad \pi_m^t \tag{3}$$

$$\sum_{d\in\mathcal{D}_r} \sum_{t\in\mathcal{T}_{rd}} x_{rdt} \leq 1 \qquad (r \in \mathcal{R}), \quad \pi_r \tag{4}$$

trailer capacity constraints:

$$q_{th} \leq q_{t,h-1} - \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta_{ihr} q_{irt} + Q_t \sum_{d\in\mathcal{D}}\sum_{r\in\mathcal{R}^{dt}} \delta^{src}_{hr} x_{rdt} \quad (t\in\mathcal{T}, h\in\mathcal{H}\setminus\{0\}), \quad \gamma_{th} \tag{5}$$

$$q_{t0} \leq q^{init}_t - \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta_{i0r} q_{irt} + Q_t \sum_{d\in\mathcal{D}}\sum_{r\in\mathcal{R}^{dt}} \delta^{src}_{0r} x_{rdt} \quad (t\in\mathcal{T}), \quad \gamma_{t0} \tag{6}$$

$$q_{th} \leq Q_t - \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta^{src+}_{ihr} q_{irt} - \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta^{start}_{hr}(1-\delta^{src}_{hr})\delta_{ihr} q_{irt} \quad (t\in\mathcal{T}, h\in\mathcal{H}), \quad \gamma^{up}_{th} \tag{7}$$

$$q_{t,h-1} \geq \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta^{src-}_{ihr} q_{irt} + \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta^{end}_{hr}(1-\delta^{src}_{hr})\delta_{ihr} q_{irt} \quad (t\in\mathcal{T}, h\in\mathcal{H}\setminus\{0\}), \quad \gamma^{down}_{th} \tag{8}$$

$$q^{init}_t \geq \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta^{src-}_{i0r} q_{irt} + \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r} \delta^{end}_{0r}(1-\delta^{src}_{0r})\delta_{i0r} q_{irt} \quad (t\in\mathcal{T}), \quad \gamma^{down}_{t0} \tag{9}$$

$$\sum_{i\in\mathcal{V}_s} q_{irt} \leq Q_t \sum_{d\in\mathcal{D}_{rt}} x_{rdt} \quad (t\in\mathcal{T}, r\in\mathcal{R}^t, s\in\mathcal{T}rip_r), \quad \gamma_{trs} \tag{10}$$

delivery constraints:

$$\sum_{t\in\mathcal{T}_r} q_{irt} \geq q^{min}_i \sum_{d\in\mathcal{D}_r}\sum_{t\in\mathcal{T}_{rd}} x_{rdt} \quad (r\in\mathcal{R}, i\in\mathcal{V}^1_r), \quad \beta_{ri} \tag{11}$$

$$\sum_{r\in\mathcal{R}_i}\sum_{t\in\mathcal{T}_r} q_{irt} \geq q^{min}_i \quad (i\in\mathcal{V}^2), \quad \beta^{min}_i \tag{12}$$

$$\sum_{r\in\mathcal{R}_i}\sum_{t\in\mathcal{T}_r} q_{irt} \leq q^{max}_i \quad (i\in\mathcal{V}^2), \quad \beta^{max}_i \tag{13}$$

customer inventory constraints:

$$S^i_{h-1} + \sum_{r\in\mathcal{R}_i}\sum_{t\in\mathcal{T}_r} \delta_{ihr} q_{irt} - d^i_h = S^i_h \quad (i\in\mathcal{V}^1, h\in\mathcal{H}\setminus\{0\}), \quad \sigma_{ih} \tag{14}$$

$$stock^i_0 + \sum_{r\in\mathcal{R}_i}\sum_{t\in\mathcal{T}_r} \delta_{i0r} q_{irt} - d^i_0 = S^i_0 \quad (i\in\mathcal{V}^1), \quad \sigma_{i0} \tag{15}$$

$$S^i_h \leq stock^i_{max} \quad (i\in\mathcal{V}^1, h\in\mathcal{H}), \quad \sigma^{max}_{ih} \tag{16}$$

and with decision variable definitions:

$$x_{rdt} \in \{0,1\} \quad (d\in\mathcal{D}, t\in\mathcal{T}, r\in\mathcal{R}^{dt}), \tag{17}$$

$$q_{irt} \geq 0 \quad (r\in\mathcal{R}, t\in\mathcal{T}_r, i\in\mathcal{V}_r), \tag{18}$$

$$q_{th} \geq 0 \quad (t\in\mathcal{T}, h\in\mathcal{H}), \tag{19}$$

$$S^i_h \geq 0 \quad (i\in\mathcal{V}^1, h\in\mathcal{H}). \tag{20}$$

We first describe equations that can quickly be explained. Other constraints are explained in details after, as well as new notation introduced for the model. On the extreme

right of the equations, we introduce, in red, notation for dual variables. These will be used later in the branch-cut-and-price algorithm.

Fractional objective function (1) is the logistics ratio. Constraints (4) state that every shift is selected at most once in a solution. These constraints stem from Assumption 4 and are useful for inventory constraints (11) below. Constraints (11) ensure that minimal quantities are delivered to VMI customers at each visit. Constraints (12) and (13) respectively impose a minimal and a maximal accumulated quantities delivered to each call-in customer. Constraints (14) and (15) avoid stockouts at VMI customers. These constraints remain easy to formulate thanks to Assumption 1. Constraints (16) limit the stock level at VMI customers. In these three sets of constraints, the inventory level is checked at the end of the period, once the deliveries have been made and the demand is satisfied.

In Constraints (2) we introduce notation $\mathcal{S}t(\mathcal{R}^d)$ and $\mathcal{K}_m^d$. Given a time instant (minute) m and a driver d, $\mathcal{K}_m^d = \{r \in \mathcal{R}^d : st_r \leq m, et_r + rest_d \geq m\}$. $\mathcal{K}_m^d$ represents a set of shifts that are not compatible at time m either because they overlap or because they would not satisfy the minimal duration $rest_d$ imposed between two shifts assigned to the same driver. Constraints (2) impose these incompatibilities. They are only needed at time instants (minutes) where a new shift comes into play, *i.e.*, at the starting times of the shifts. $\mathcal{S}t(\mathcal{R}^d)$ is the set of these starting times: $\mathcal{S}t(\mathcal{R}^d) = \{st_r, r \in \mathcal{R}^d\}$. Note that the time scale for these constraints is the minute. Notation introduced in these constraints, as well as in the next constraints, are recalled in Table 5.

$\mathcal{S}t(\mathcal{R}^d)$	set of starting times of shifts in $\mathcal{R}^d$
$\mathcal{K}_m^d$	set of shifts incompatible at time m for driver d
$\mathcal{S}t(\mathcal{R}^t)$	set of starting times of shifts in $\mathcal{R}^t$
$\mathcal{K}_m^t$	set of shifts incompatible at time m for trailer t

Table 5: Additional notation introduced for shift compatibility management

Constraints (3) are equivalent for trailers. We note $\mathcal{S}t(\mathcal{R}^t)$ the set of starting times of shifts in $\mathcal{R}^t$. We construct a set $\mathcal{K}_m^t$ for each $m \in \mathcal{S}t(\mathcal{R}^t)$ such that $\mathcal{K}_m^t = \{r \in \mathcal{R}^t : st_r \leq m, et_r \geq m\}$. Then constraints (3) enforce that two shifts requiring the same trailer cannot overlap. Again, the time scale for these constraints is the minute.

Constraints (5)-(10) are by far the less easily readable. They are introduced to manage trailer capacity. One important aspect of the REC-IRP is that the remaining load in a trailer at the end of a trip is available for the next trip. Due to this specific feature, we cannot model the trailer capacity constraint by simply bounding the quantity delivered on each trip. We proposed to trace the trailer loading levels with variables q_{th}. These constraints also imply introducing notation given in Table 6 and are at the origin of assumptions 2 and 3.

Constraints (5) are active when no source is attained by trailer t during period h. Then, the loading level of the trailer at the end of the period cannot exceed its level at

δ_{hr}^{src}	1 if shift r reaches a source at period h, 0 otherwise
δ_{ihr}^{src+}	1 if, during period h, shift r both visits, in this order, a source and customer i, 0 otherwise
δ_{ihr}^{src-}	1 if, during period h, shift r both visits, in this order, customer i and a source, 0 otherwise
δ_{hr}^{start}	1 if shift r starts at period h, 0 otherwise
δ_{hr}^{end}	1 if shift r ends at period h, 0 otherwise

Table 6: Additional notation introduced for the trailer capacity management

the beginning of the period minus the quantity delivered during the period. Note that when the source is visited, Constraints (5) are equivalent to the sum of Constraints (7) and Constraints (8). Constraints (6) consider the initial period $h = 0$.

When a source is reached by a trailer during period h, Constraints (7) limit the load of the trailer at the end of the period. The value subtracted to Q_t on the right-hand side of the constraint evaluates the quantity of the commodity delivered after the source in period h, both by the shift that visits the source and possibly successive shifts. We emphasize that a shift is assumed to last more than one period and that the source is assumed to be visited at most once in a given period. Therefore, at most two shifts can be associated with trailer t at a given period. In such a case, the first shift may include a visit to the source while the second shift includes visits to customers only. The latter starts in this period. If no source is reached, these constraints are dominated by Constraints (5).

When a source is reached by a trailer during period h, Constraints (8) guarantee that the total quantity delivered does not exceed the quantity available before reaching a source. More precisely, the two terms on the right-hand side evaluate the quantity delivered before the source in period h, either by the shift that visits the source or by a shift scheduled before, i.e., that ends at period h. If no source is visited, these constraints are dominated by Constraints (5). Constraints (9) play the same role for the initial period ($h = 0$).

Constraints (10) are only active when shift r is not assigned to a trailer and a driver. It prevents from delivering commodities with shift r in this case. Otherwise, the maximal amount of commodity delivered is restricted by the previous constraints. Note that these constraints are tight.

3.3 Modified objective function

In the algorithm presented in Section 6, the objective function is actually slightly modified to favor early deliveries and identify feasible solutions more rapidly. We proceed as follows. We introduce a *delivery profit* vector c_h that indicates the profit raised when delivering a commodity unit to a customer at a specified time $h \in \mathcal{H}$ (see Section 6.2 for details on the values given to c_h). Note that the actual objective function of the REC-IRP is retrieved if

$c_h = 1$.

Objective function (1) then becomes:

$$\text{minimize } \frac{\sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}^{dt}} c_{rdt}x_{rdt}}{\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}_r}\sum_{i\in\mathcal{V}_r}(\sum_{h\in\mathcal{H}}\delta_{ihr}c_h)q_{irt}}$$

3.4 Linearized objective function

In the algorithm presented in Section 6, the objective function is linearized by following the principle of Dinkelbach's algorithm. Dinkelbach's method solves nonlinear problems with a fractional objective function, a concave numerator and a (non-zero) convex denominator. Since the linear relaxation of our model fits these conditions, its application to our problem is justified.

To briefly explain how Dinkelbach's algorithm works, let us consider a linear program with a linear fractional objective:

$$(P) \quad \min_{x\in X} \frac{N(x)}{D(x)},$$

with $N(x)$ and $D(x)$ linear and positive functions for all $x \in X$. We note $f(Z)$ the value of the optimal solution of the parametric linear program:

$$\min_{x\in X}(N(x) - Z \times D(x)).$$

One can easily prove that f has a unique root Z^* (*i.e.*, $f(Z^*) = 0$) and that $Z^* = \min_{x\in X} \frac{N(x)}{D(x)}$.

To solve the problem (P), Dinkelbach's algorithm computes Z^* applying Newton's algorithm: it starts from a reasonable value Z_0, and it iterates as follows:

$$Z_{k+1} = Z_k - \frac{f(Z_k)}{f'(Z_k)} = \frac{N(x_k^*)}{D(x_k^*)},$$

where x_k^* is the optimal solution of the parametric linear program for parameter value Z_k.

In our case, the objective function of the parametric linear program, for a given parameter value Z, is then:

$$\text{minimize } \sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}^{dt}} c_{rdt}x_{rdt} - Z\sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}_r}\sum_{i\in\mathcal{V}_r}(\sum_{h\in\mathcal{H}}\delta_{ihr}c_h)q_{irt},$$

To simplify notation, we introduce $Z_{ir} = Z\sum_{h\in\mathcal{H}}\delta_{ihr}c_h$. Then, the objective function becomes:

$$\text{minimize } \sum_{d\in\mathcal{D}}\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}^{dt}} c_{rdt}x_{rdt} - \sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}_r}\sum_{i\in\mathcal{V}_r} Z_{ir}q_{irt},$$

In Section 6.2, we precise how parameter Z varies in the course of the algorithm.

3.5 Valid inequalities

The linear relaxation of the model can be enriched by the introduction of the following three classes of valid inequalities related to visits to customers.

For call-in customers, at least one visit is needed:

$$\sum_{r\in\mathcal{R}_i}\sum_{d\in\mathcal{D}_r}\sum_{t\in\mathcal{T}_{rd}} x_{rdt} \geq 1 \qquad (i \in \mathcal{V}^2). \quad \alpha_i \tag{21}$$

For VMI customers, the minimal number of visits can be deduced from the accumulated demand of the customer on the horizon, the inventory capacity of the customer and the maximal capacity of a trailer. Let call n_i this value. For $h, h' \in \mathcal{H}$ with $h \leq h'$, we introduce $D^i_{hh'} = \sum_{l=h}^{h'} d^i_l$ the cumulated demand of customer $i \in \mathcal{V}^1$ between the beginning of period h and the end of period h'. Then,

$$n_i = \left\lceil \frac{D^i_{0H} - stock^i_0}{\min\{stock^{max}_i, \max_{r\in\mathcal{R}_i, t\in\mathcal{T}_r} Q_t\}} \right\rceil.$$

and the following constraints can be imposed:

$$\sum_{r\in\mathcal{R}_i}\sum_{d\in\mathcal{D}_r}\sum_{t\in\mathcal{T}_{rd}} x_{rdt} \geq n_i \qquad (i \in \mathcal{V}^1). \quad \alpha_i \tag{22}$$

Additional valid inequalities can be added to better manage the visit times at VMI customers. For these inequalities, we introduce functions $\pi_i : \mathcal{H} \to \mathcal{H} \cup \{-1\}$, $i \in \mathcal{V}^1$ defined as follows:

- if $D^i_{hH} > stock^{max}_i$, $\pi_i(h)$ is the smallest period $h' \geq h$ with $D^i_{hh'} > stock^{max}_i$;
- if $D^i_{hH} \leq stock^{max}_i$, $\pi_i(h) = -1$.

Then, the valid inequalities are as follows:

$$\sum_{r\in\mathcal{R}_i}\sum_{d\in\mathcal{D}_r}\sum_{t\in\mathcal{T}_{rd}}\sum_{l=h}^{l=\pi_i(h)} \delta_{ilr} x_{rdt} \geq 1 \qquad (i \in \mathcal{V}^1, h \in \mathcal{H} : \pi_i(h) > 0). \quad \alpha_{ih} \tag{23}$$

3.6 Artificial variables

In a branch-cut-and-price algorithm, the model presented in Section 3.2 will be solved over a restricted number of *td*-shifts. As a drawback, the restricted model may not admit a feasible solution even if the problem does. To prevent this situation we introduce some so-called artificial variables. In the following we detail the modifications on the objective function and on some constraints of the model due to the inclusion of these variables. We introduce four types of non-negative artificial variables. Basically, all these variables enable some deliveries to be skipped:

(1) y_i, to deactivate Valid Inequalities (21) or (22) for customers $i \in \mathcal{V}$;

(2) w_i, to deactivate Valid Inequalities (23) for customers $i \in \mathcal{V}^1$;

(3) oos_{ih}, to allow customer $i \in \mathcal{V}^1$ to get out of stock at period h;

(4) $drop_i$, to allow not to visit customer $i \in \mathcal{V}^2$.

Artificial variables are included in the objective function multiplied by large values M_i ($i \in \mathcal{V}$).

The objective function becomes:

$$\text{minimize} \sum_{d\in\mathcal{D}} \sum_{t\in\mathcal{T}} \sum_{r\in\mathcal{R}^{dt}} c_{rdt}x_{rdt} - \sum_{r\in\mathcal{R}} \sum_{t\in\mathcal{T}_r} \sum_{i\in\mathcal{V}_r} Z_{ir}q_{irt} + \sum_{i\in\mathcal{V}} M_i y_i + \sum_{i\in\mathcal{V}^1} M_i w_i$$

$$+ \sum_{i\in\mathcal{V}^1} \sum_{h\in\mathcal{H}} M_i oos_{ih} + \sum_{i\in\mathcal{V}^2} M_i drop_i.$$

Constraints (12), (14), (21), (22), (23) become:

$$\sum_{r\in\mathcal{R}_i} \sum_{t\in\mathcal{T}_r} q_{irt} \geq q_i^{min} - drop_i \qquad (i \in \mathcal{V}^2),$$

$$S_{h-1}^i + \sum_{r\in\mathcal{R}_i} \sum_{t\in\mathcal{T}_r} \delta_{ihr} q_{irt} - d_h^i = S_h^i - oos_{ih} \qquad (i \in \mathcal{V}^1, h \in \mathcal{H}),$$

$$\sum_{r\in\mathcal{R}_i} \sum_{d\in\mathcal{D}_r} \sum_{t\in\mathcal{T}_{rd}} x_{rdt} \geq 1 - y_i \qquad (i \in \mathcal{V}^2),$$

$$\sum_{r\in\mathcal{R}_i} \sum_{d\in\mathcal{D}_r} \sum_{t\in\mathcal{T}_{rd}} x_{rdt} \geq n_i(1 - y_i) \qquad (i \in \mathcal{V}^1),$$

$$\sum_{r\in\mathcal{R}_i} \sum_{d\in\mathcal{D}_r} \sum_{t\in\mathcal{T}_{rd}} \sum_{l=h}^{\pi_i(h)} \delta_{ilr} x_{rdt} \geq 1 - w_i \qquad (i \in \mathcal{V}^1, h \in \mathcal{H} : \pi_i(h) > 0).$$

In the following of the paper, we will say that a mathematical program is *feasible* when it admits a solution with all artificial variables equal to zero. Otherwise we will say that the mathematical program is *infeasible*.

3.7 Master problem

In the rest of the paper we will call Master Problem the linear relaxation of the mathematical model defined by (1)–(20) with the modifications described in the preceding subsections: modified and linearized objective function, valid inequalities and artificial variables added. The Master Problem will be denoted MP.

4 Column generation framework

Because of its number of variables, MP cannot be solved directly with a linear programming solver. In this section we describe how a column generation mechanism can be applied to progressively enrich a restricted version of MP.

Let us introduce some notation that will be used in this section. We note $\mathcal{TDR}$ the complete set of triplets (r, d, t), with $r \in \mathcal{R}$, $d \in \mathcal{D}_r$ and $t \in \mathcal{T}_{rd}$. $\mathcal{TDR}$ is the set of all the td-shifts. A variable x_{rdt} is present in MP for each triplet in $\mathcal{TDR}$. Similarly, we note $\mathcal{TR}$ the set of pairs (r, t), with $r \in \mathcal{R}$ and $t \in \mathcal{T}_r$, and $\mathcal{DR}$ the set of pairs (r, d), with $r \in \mathcal{R}$ and $d \in \mathcal{D}_r$.

We call RMP_k the restricted MP at a step k of the algorithm and we explain how RMP_k is constructed from a subset $\mathcal{TDR}_k \subseteq \mathcal{TDR}$ of td-shifts. Together, we explain how sets $\mathcal{TR}_k \subseteq \mathcal{TR}$, $\mathcal{DR}_k \subseteq \mathcal{DR}$ and $\mathcal{R}_k \subseteq \mathcal{R}$ are constructed. To this aim, we equivalently explain how we proceed when a td-shift is added to $\mathcal{TDR}_k$.

When a new td-shift (r, d, t) is added to $\mathcal{TDR}_k$:

- variable x_{rdt} is added to the model;
- if (r, t) is not in $\mathcal{TR}_k$, (r, t) is added to $\mathcal{TR}_k$, variables q_{irt} are added to the model for all $i \in r$, and Constraints (10) are added for pair (r, t) for every trip $s \in \mathcal{T}rip_r$. Furthermore, if the starting time of r is not in $\mathcal{S}t(\mathcal{R}_k^t)$, Constraint (3) is added for pair (t, st_r);
- if (r, d) is not in $\mathcal{DR}_k$, (r, d) is added to $\mathcal{DR}_k$ and, if the starting time of r is not in $\mathcal{S}t(\mathcal{R}_k^d)$, Constraint (2) is added for pair (d, st_r);
- if r is not in $\mathcal{R}_k$, r is added to $\mathcal{R}_k$, Constraint (4) is added to the model for r, as well as Constraint (11) for every $i \in \mathcal{V}_r^1$.

In our algorithm, see Section 6, we start with an initial set of promising td-shifts $\mathcal{TDR}_0$ and regularly insert new td-shifts to drive the search towards feasible/better solutions. Since the modifications described above impact RMP_k when new td-shifts are added, standard column generation cannot be applied. We recall in the next subsection how to adapt column generation for the simultaneous generation of columns and cuts. Then, we explain how it is applied to the REC-IRP.

4.1 About simultaneous cut and column generation

We note D the dual of MP and D_k the dual of RMP_k. When the addition of columns to a restricted master problem also implies the addition of cuts, it raises new difficulties that invalidate the standard column generation procedure (see Feillet et al. (2010)). Indeed, the

usual optimality criterion used in column generation relies on the fact that when no column of negative reduced cost exists, the current dual solution (obtained from the solution of the restricted master problem) is feasible for the dual of the master problem. In case of simultaneous cut and column generation it does not hold, since the cuts, that are missing in the restricted master problem, translate into missing variables into the dual of the master problem. Then, no guarantee of feasibility for the current dual solution with regards to the dual of the master problem is possible, without defining a value for these missing variables.

Feillet et al. (2010) propose to tackle this difficulty with a 3-step methodology:

1. from optimal solution X_k^* of RMP_k, construct a feasible solution X of MP, with cost $z_{MP} = z^*_{RMP_k}$;
2. from optimal solution Π_k^* of D_k, construct a (not necessarily feasible) solution Π of D, with cost $z_D = z^*_{D_k}$, *i.e.*, $z_D = z_{MP}$;
3. if Π is feasible, the optimality criterion is met: stop the algorithm; otherwise, at least one constraint of D is violated; add to $\mathcal{TDR}_k$ one td-shift from $\mathcal{TDR} \setminus \mathcal{TDR}_k$ for which a constraint is violated and pass to the next iteration.

Unfortunately, no simple general rule seems to exist to construct solutions X and Π. In the next section, we see how this scheme can be applied to the REC-IRP.

4.2 Application to the REC-IRP

Let us start with a solution X_k^* of RMP_k and the associated dual solution Π_k^* of D_k. The first step of the methodology consists in constructing a feasible solution X of MP, having the same cost as X_k^*. We simply fix to 0 all variables x_{rdt} for $(r, d, t) \in \mathcal{TDR} \setminus \mathcal{TDR}_k$ and q_{irt} for $(r, t) \in \mathcal{TR} \setminus \mathcal{TR}_k$, and keep other variables to their value in X_k^*. X is feasible for MP and $z_{MP} = z^*_{RMP_k}$.

The next step is then to construct a not necessarily feasible solution Π for the dual D, having the same cost as Π_k^*. For this step, we need to consider D carefully. D is composed of eight series of constraints, one for each type of primal variables (x_{rdt}, q_{irt}, q_{th}, S_h^i), plus one for each type of artificial variables (y_i, w_i, oos_{ih}, $drop_i$). The differences between D_k and D are the followings:

- the constraints derived from primal variables x_{rdt} with $(r, d, t) \in \mathcal{TDR} \setminus \mathcal{TDR}_k$ and primal variables q_{irt} with $(r, t) \in \mathcal{TR} \setminus \mathcal{TR}_k$ are missing in D_k
- some variables π_m^d, π_m^t, π_r, γ_{trs} and β_{ri}, respectively derived from primal constraints (2), (3), (4), (10) and (11) are missing in D_k.

We define Π as follows. First, all variables already defined in D_k keep the same value as in Π_k^*. Then, variables π_m^d, π_m^t and π_r not in D_k are set to 0. The remaining variables to be

fixed are variables γ_{trs} for $(r,t) \in \mathcal{TR} \setminus \mathcal{TR}_k$ and $s \in \mathcal{T}rip_r$, and β_{ri} for $r \in \mathcal{R} \setminus \mathcal{R}_k$ and $i \in \mathcal{V}_r^1$.

Before defining these variables, let us underline that, independently of the values to be assigned, several important conditions already hold. First, $z_D = z^*_{D_k}$. Indeed, variables γ_{trs} and β_{ri} derive from primal Constraints (10) and (11) that have no fixed term, and thus do not appear in the dual objective function. Second, Π satisfy all the dual constraints already present in D_k. Indeed, none of the remaining variables appear in these constraints. Thus, we now have to focus on finding values for the remaining variables and evaluating the feasibility of the missing constraints.

We write the missing constraints below. In order to facilitate the understanding of the constraints, we define by convention all dual variables nonnegative, except variables σ_{ih} $(i \in \mathcal{V}^1, h \in \mathcal{H})$ associated with Constraints (14) and (15) that are unsigned.

For $(r,d,t) \in \mathcal{TDR} \setminus \mathcal{TDR}_k$, the dual constraint in D derived from primal variable x_{rdt} is:

$$c_{rdt} + \sum_{i \in \mathcal{V}_r^1} q_i^{min} \beta_{ri} + \sum_{\{m \in \mathcal{S}t(\mathcal{R}^d): r \in \mathcal{K}_m^d\}} \pi_m^d + \sum_{\{m \in \mathcal{S}t(\mathcal{R}^t): r \in \mathcal{K}_m^t\}} \pi_m^t + \pi_r - \sum_{i \in \mathcal{V}_r} \alpha_i$$
$$- \sum_{i \in \mathcal{V}_r^1} \sum_{\{h \in \mathcal{H}: h \leq st_{ri} \leq \pi_i(h)\}} \alpha_{ih} - Q_t \sum_{h \in \mathcal{H}} \delta_{hr}^{src} \gamma_{th} - Q_t \sum_{s \in \mathcal{T}rip_r} \gamma_{trs} \geq 0. \tag{24}$$

For $(r,t) \in \mathcal{TR} \setminus \mathcal{TR}_k$, $i \in \mathcal{V}_r^1$, and s the trip of customer i in td-shift r, the dual constraint derived from primal variable q_{irt} is:

$$\sum_{h \in \mathcal{H}} (\delta_{ihr}^{src+} + \delta_{hr}^{start}(1 - \delta_{hr}^{src})\delta_{ihr})\gamma_{th}^{up} + \sum_{h \in \mathcal{H}} (\delta_{ihr}^{src-} + \delta_{hr}^{end}(1 - \delta_{hr}^{src})\delta_{ihr})\gamma_{th}^{down}$$
$$+ \sum_{h \in \mathcal{H}} \delta_{ihr}\gamma_{th} + \gamma_{trs} - \beta_{ri} - \sum_{h \in \mathcal{H}} \delta_{ihr}\sigma_{ih} \geq Z_{ir}.$$

Last, for $(r,t) \in \mathcal{TR} \setminus \mathcal{TR}_k$, $i \in \mathcal{V}_r^2$, and s the trip of customer i in td-shift r, the dual constraint derived from primal variable q_{irt} is:

$$\sum_{h \in \mathcal{H}} (\delta_{ihr}^{src+} + \delta_{hr}^{start}(1 - \delta_{hr}^{src})\delta_{ihr})\gamma_{th}^{up} + \sum_{h \in \mathcal{H}} (\delta_{ihr}^{src-} + \delta_{hr}^{end}(1 - \delta_{hr}^{src})\delta_{ihr})\gamma_{th}^{down}$$
$$+ \sum_{h \in \mathcal{H}} \delta_{ihr}\gamma_{th} + \gamma_{trs} + \beta_i^{max} - \beta_i^{min} \geq Z_{ir}.$$

For the sake of simplifying these constraints, we introduce :

$$\delta_{ihr}^{up} = \delta_{ihr}^{src+} + \delta_{hr}^{start}(1 - \delta_{hr}^{src})\delta_{ihr},$$

and

$$\delta_{ihr}^{down} = \delta_{ihr}^{src-} + \delta_{hr}^{end}(1 - \delta_{hr}^{src})\delta_{ihr}.$$

Note that $\delta^{up}_{ihr} = 1$ in two special cases: a source and customer i are visited in this order at period h in shift r, or shift r starts at period h and visits customer i at the same period but does not visit a source. Similarly, $\delta^{down}_{ihr} = 1$ if customer i and a source are visited in this order at period h in shift r, or shift r ends at period h and visits customer i at the same period but does not visit a source.

Given $(r,t) \in \mathcal{TR} \setminus \mathcal{TR}_k$, $i \in \mathcal{V}^1_r$, and s the trip of customer i in *td*-shift r, the dual constraint derived from primal variable q_{irt} is then:

$$\sum_{h\in\mathcal{H}} \delta^{up}_{ihr}\gamma^{up}_{th} + \sum_{h\in\mathcal{H}} \delta^{down}_{ihr}\gamma^{down}_{th} + \sum_{h\in\mathcal{H}} \delta_{ihr}\gamma_{th} + \gamma_{trs} - \beta_{ri} - \sum_{h\in\mathcal{H}} \delta_{ihr}\sigma_{ih} \geq Z_{ir}, \tag{25}$$

and given $(r,t) \in \mathcal{TR} \setminus \mathcal{TR}_k$, $i \in \mathcal{V}^2_r$, and s the trip of customer i in *td*-shift r, the dual constraint derived from primal variable q_{irt} is:

$$\sum_{h\in\mathcal{H}} \delta^{up}_{ihr}\gamma^{up}_{th} + \sum_{h\in\mathcal{H}} \delta^{down}_{ihr}\gamma^{down}_{th} + \sum_{h\in\mathcal{H}} \delta_{ihr}\gamma_{th} + \gamma_{trs} + \beta^{max}_i - \beta^{min}_i \geq Z_{ir}. \tag{26}$$

When defining the remaining dual variables in Π, we thus only have to consider the feasibility of Constraints (24), (25) and (26). Before providing a formal proof, we give some intuition on how we proceed. Focusing on Constraints (25) and (26), it is easy to see that choosing large values for variables γ_{trs} and small values for variables β_{ri} allows satisfying these constraints. However, having large values for variables γ_{trs} and small values for variables β_{ri} decreases the chances that Constraints (24) are satisfied. We propose to select values that ensure the feasibility of Constraints (25) and (26), but with variables γ_{trs} as small as possible and variables β_{ri} as large as possible, so that the chances, that Constraints (24) are feasible, are increased. However, clearly, looking at Constraints (25) a compromise has to be done between the values of variables γ_{trs} and β_{ri}. We state and solve this compromise with Property 1.

Property 1. *Let consider $r \in \mathcal{R} \setminus \mathcal{R}_k$, and assume that variables γ_{trs} are set for all $t \in \mathcal{T}_r, s \in \mathcal{T}rip_r$ and that variables β_{ri} are set for all $i \in \mathcal{V}^1_r$. Let γ_1 be the minimal value of all these variables. Let γ_2 be the minimal gap between the left-hand side and the right-hand side for all Constraints (26) associated with r. Let finally assume that $\gamma = \min(\gamma_1, \gamma_2) > 0$. Then, subtracting γ to all the variables mentioned above increases the chances that dual solution Π is feasible.*

Proof. First, the only constraints impacted by the values of the variables mentioned in the property are the constraints (24), (25) and (26) associated with r. It is easy to see that the subtraction of γ does not impact Constraints (25) and that Constraints (26) remain feasible. Given a *td*-shift (r,d,t), the impact on Constraint (24) is the addition to the left-hand side of term $-(\sum_{i\in\mathcal{V}^1_r} q^{min}_i)\gamma + |\mathcal{T}rip_r|Q_t\gamma$. $\sum_{i\in\mathcal{V}^1_r} q^{min}_i$ is a lower bound on the quantities delivered on shift r, while $|\mathcal{T}rip_r|Q_t$ is an upper bound. It proves that the term added to Constraint (24) is nonnegative, which concludes the proof. □

We now detail how values are fixed for γ_{trs} for $(r,t) \in \mathcal{TR} \setminus \mathcal{TR}_k$ and $s \in \mathcal{T}rip_r$, and β_{ri} for $r \in \mathcal{R} \setminus \mathcal{R}_k$ and $i \in \mathcal{V}_r^1$. Two cases are considered.

First, $r \in \mathcal{R}_k$. In this case, variables β_{ri} are already fixed and we need to define γ_{trs} for all t such that $(r,t) \in \mathcal{TR} \setminus \mathcal{TR}_k$ and for all $s \in \mathcal{T}rip_r$. Clearly, the smallest γ_{trs} is the best choice with respect to Constraint (24). So, we fix γ_{trs} to the minimal value leaving Constraints (25) and (26) feasible:

$$\begin{aligned}
\gamma_1 &= \max_{i \in \mathcal{V}_s^1} Z_{ir} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{up} \gamma_{th}^{up} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{down} \gamma_{th}^{down} - \sum_{h \in \mathcal{H}} \delta_{ihr} \gamma_{th} + \beta_{ri} + \sum_{h \in \mathcal{H}} \delta_{ihr} \sigma_{ih}, \\
\gamma_2 &= \max_{i \in \mathcal{V}_s^2} Z_{ir} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{up} \gamma_{th}^{up} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{down} \gamma_{th}^{down} - \sum_{h \in \mathcal{H}} \delta_{ihr} \gamma_{th} - \beta_i^{max} + \beta_i^{min}, \\
\gamma_{trs} &= \max(\gamma_1, \gamma_2, 0).
\end{aligned}$$

Second, $r \notin \mathcal{R}_k$. In this case, none of the variables γ_{trs} for $t \in \mathcal{T}_r$ and q_{irt} for $i \in \mathcal{V}_r^1$ are fixed. In view of Property 1, we favor small values for these variables:

$$\begin{aligned}
\gamma_1 &= \max_{i \in \mathcal{V}_s^1} Z_{ir} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{up} \gamma_{th}^{up} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{down} \gamma_{th}^{down} - \sum_{h \in \mathcal{H}} \delta_{ihr} \gamma_{th} + \sum_{h \in \mathcal{H}} \delta_{ihr} \sigma_{ih}, \\
\gamma_2 &= \max_{i \in \mathcal{V}_s^2} Z_{ir} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{up} \gamma_{th}^{up} - \sum_{h \in \mathcal{H}} \delta_{ihr}^{down} \gamma_{th}^{down} - \sum_{h \in \mathcal{H}} \delta_{ihr} \gamma_{th} - \beta_i^{max} + \beta_i^{min}, \\
\gamma_{trs} &= \max(\gamma_1, \gamma_2, 0).
\end{aligned}$$

Note that γ_{trs} is computed assuming β_{ri} variables set to 0.

$$\beta_{ri} = \min_{t \in \mathcal{T}} \sum_{h \in \mathcal{H}} \delta_{ihr}^{up} \gamma_{th}^{up} + \sum_{h \in \mathcal{H}} \delta_{ihr}^{down} \gamma_{th}^{down} + \sum_{h \in \mathcal{H}} \delta_{ihr} \gamma_{th} + \gamma_{trs} - \sum_{h \in \mathcal{H}} \delta_{ihr} \sigma_{ih} - Z_{ir}.$$

Note that with variables γ_{trs} fixed, variables β_{ri} are as large as possible.

With this definition, the dual solution Π has the same cost as the dual solution Π_k^* of the restricted dual program and all constraints are feasible except possibly Constraints (24) for some *td*-shifts $(r,t,d) \in \mathcal{TDR} \setminus \mathcal{TDR}_k$. The goal of the pricing problem is to search for *td*-shifts (r,d,t) such that this constraint is not satisfied, *i.e.*:

$$\begin{aligned}
c_{rdt} + \sum_{i \in \mathcal{V}_r^1} q_i^{min} \beta_{ri} + \sum_{\{m \in \mathcal{S}t(\mathcal{R}_k^d) : r \in \mathcal{K}_m^d\}} \pi_m^d + \sum_{\{m \in \mathcal{S}t(\mathcal{R}_k^t) : r \in \mathcal{K}_m^t\}} \pi_m^t - \sum_{i \in \mathcal{V}_r} \alpha_i \\
- \sum_{i \in \mathcal{V}_r^1} \sum_{\{h \in \mathcal{H} : h \leq st_{ri} \leq \pi_i(h)\}} \alpha_{ih} - Q_t \sum_{h \in \mathcal{H}} \delta_{hr}^{src} \gamma_{th} - Q_t \sum_{s \in \mathcal{T}rip_r} \gamma_{trs} < 0. \qquad (27)
\end{aligned}$$

In the following, we will call the left-hand side of (27) the reduced cost of the *td*-shift.

In our algorithm, presented in Section 6, we developed two procedures to insert new *td*-shifts with negative reduced cost in RMP_k. The first procedure considers existing *td*-shifts and tries to generate new *td*-shifts by changing the trailer, the driver and/or the

time-stamp of these *td*-shifts. In this first procedure, the reduced cost is evaluated exactly. This procedure is detailed in Section 6.3.3.

The second procedure aims at generating completely new *td*-shifts. It is classically based on the solution of a shortest path problem with resource constraints by dynamic programming. However, because of the difficulty of the pricing, reduced costs are only evaluated heuristically in what we call an implicit-relaxation-based dynamic programming procedure. This procedure is described in the next section.

5 Implicit-relaxation-based dynamic programming procedure

The implicit-relaxation-based dynamic programming procedure is called with a few parameters that define the subset of *td*-shifts that is explored: the *td*-shifts start at time m_0 (in minutes), have to be finished at time m_1 (in minutes), use driver d_0 and trailer t_0. For short, the notation introduced below is not indexed by these parameters. In Section 6, we detail how these parameters are fixed and when the procedure is called.

The dynamic programming procedure is named implicit-relaxation-based because it is implicitly applied on a relaxation of MP. By implicitly, we mean that this relaxation is not implemented. Instead, when RMP_k is solved, a dual vector consistent with the dual space of this relaxation is constructed. Using this dual information, we derive a (heuristic) reduced-cost condition that is checked for the generation of new *td*-shifts. We call pricing problem, the problem of identifying *td*-shifts that do not satisfy this condition. This condition presents the advantage to be easier to consider in a dynamic programming scheme than condition (27). In particular, as it will be seen, the relaxation is constructed so that waiting is of no interest when *td*-shifts are generated in the pricing problem. However, contrary to standard column generation, there is no guarantee that the actual reduced cost of the *td*-shift in MP is negative. Hence, this reduced cost has to be computed exactly after the pricing problem is solved, which is done using the formula presented in Section 4.

We now describe the relaxation, the construction of the dual vector, the reduced-cost condition and the dynamic programming algorithm developed to identify *td*-shifts that do not satisfy this condition.

5.1 Implicit relaxation and dual vector

In this section, we consider the different families of constraints that are relaxed and explain how they are relaxed and with which consequences for the dual vector.

Constraints (23):
These constraints enforce visiting at least once a customer during a given period if its consumption during this period is greater than its inventory capacity. Let consider $i \in \mathcal{V}^1$. These valid inequalities are initially written:

$$\sum_{r\in\mathcal{R}_i}\sum_{d\in\mathcal{D}_r}\sum_{t\in\mathcal{T}_{rd}}\sum_{l=h}^{l=\pi_i(h)}\delta_{ilr}x_{rdt} \geq 1 - w_i \qquad (h \in \mathcal{H} : \pi_i(h) > 0)$$

We introduce set $\mathcal{H}_i = \{h \in \mathcal{H} : [m_0, m_1] \cap [h, \pi_i(h)] \neq \emptyset^1\}$ and we focus on the constraints associated with periods $h \in \mathcal{H}_i$. In these constraints, we change the term $\sum_{l=h}^{l=\pi_i(h)} \delta_{ilr}x_{rdt}$ to $\sum_{l\in[m_0,m_1]\cup[h,\pi_i(h)]} \delta_{ilr}x_{rdt}$ for all triplets (r, d, t) such that $d = d_0$, $t = t_0$. This new formulation is a relaxation because $[h, \pi_i(h)] \subseteq [m_0, m_1] \cup [h, \pi_i(h)]$.

Formally, the constraints become:

$$\sum_{r\in\mathcal{R}_i}\sum_{d\in\mathcal{D}_r}\sum_{t\in\mathcal{T}_{rd}}\sum_{l=h}^{l=\pi_i(h)}\delta_{ilr}x_{rdt} \geq 1 - w_i \qquad (h \in \mathcal{H} \setminus \mathcal{H}_i : \pi_i(h) > 0)$$

$$\sum_{r\in\mathcal{R}_i}\sum_{(d,t)\in\mathcal{D}_r\times\mathcal{T}_{rd}\setminus\{(d_0,t_0)\}}\sum_{l=h}^{l=\pi_i(h)}\delta_{ilr}x_{rdt}$$
$$+\sum_{r\in\mathcal{R}_i}\sum_{l\in[m_0,m_1]\cup[h,\pi_i(h)]}\delta_{ilr}x_{rd_0t_0} \geq 1 - w_i \qquad (h \in \mathcal{H}_i : \pi_i(h) > 0)$$

In the pricing problem, the contribution of these constraints to the reduced cost of td-shifts that visit i is $\sum_{h\in\mathcal{H}_i}\alpha_{ih}$. The interest of this relaxation relies on the fact that this term does not depend on the visiting time of i. For short, we note: $\sum_{h\in\mathcal{H}_i}\alpha_{ih} = \hat{\alpha}_i$.

Constraints (5) and (6):

These constraints are related to the trailer capacities. For these constraints, we apply a surrogate relaxation. We call h_0 and h_1 the periods (in hour) that contain m_0 and m_1, respectively. For $t = t_0$, we replace the constraints defined for $h \in \{h_0, \ldots, h_1\}$ with their sum:

$$q_{t_0h_1} \leq q_{t_0,h_0-1} - \sum_{h\in\{h_0,\ldots,h_1\}}\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{V}_r}\delta_{ihr}q_{irt_0} + Q_{t_0}\sum_{d\in\mathcal{D}}\sum_{r\in\mathcal{R}^{dt_0}} n_r^{src}x_{rdt_0}$$

with n_r^{src} equal to the number of visits to a source in r on interval $[h_0, h_1]$. If $h_0 = 0$, q_{t,h_0-1} is replaced by q_t^{init} in the above expression.

[1] note that these two intervals are not expressed with the same time unit: one has to understand that the two time intervals overlap

We note $\hat{\gamma}_{t_0}$ the dual variable associated with this constraint. In the pricing problem, the contribution of this constraint to the reduced cost of a *td*-shift is $\hat{\gamma}_{t_0}$ each time the *td*-shift visits a source. As the constraint is not in MP, we need to define a value for $\hat{\gamma}_{t_0}$. We arbitrarily set:

$$\hat{\gamma}_{t_0} = \frac{1}{h_1 + 1 - h_0} \sum_{h=h_0}^{h=h_1} \gamma_{t_0 h}.$$

The rationale is that every time the source is visited, the *td*-shift will receive a dual price equal to the average value of dual variables $\gamma_{t_0 h}$ instead of the value of the $\gamma_{t_0 h}$ variable associated with the specific period of the visit. Again, the main advantage is that the new dual variable $\hat{\gamma}_{t_0}$ is not indexed by time.

Constraints (7) to (9):

These constraints also concern trailer capacities. In these constraints, we set to 0 the coefficient of all q_{irt_0} variables. It defines a relaxation because the coefficients where initially all nonnegative on the left-hand side and nonpositive on the right-hand side of the constraints. The interest here is to get rid of the burden of carrying dual variables $\gamma^{up}_{t_0 h}$ and $\gamma^{down}_{t_0 h}$, that we suspect not being very influential.

Constraints (14) and (15):

These constraints prevent from stockouts at VMI customers. Again, these constraints are simplified using a surrogate relaxation. We reuse notation h_0 and h_1. Given $i \in \mathcal{V}^1$, we replace the constraints defined for $h \in \{h_0, \ldots, h_1\}$ with their sum:

$$S^i_{h_0-1} + \sum_{r \in \mathcal{R}_i} \sum_{t \in \mathcal{T}_r} q_{irt} - \sum_{h \in \{h_0, \ldots, h_1\}} d^i_h = S^i_{h_1} - \sum_{h \in \{h_0, \ldots, h_1\}} oos_{ih}$$

If $h_0 = 0$, $S^i_{h_0-1}$ is replaced by $stock^i_0$ in the above expression.

We note $\hat{\sigma}_i$ the dual variable associated with this constraint. As the constraint is not in MP, we need to define a value for this variable. Again, we define $\hat{\sigma}_i$ as the average value of variables σ_{ih} on the interval:

$$\hat{\sigma}_i = \frac{1}{h_1 + 1 - h_0} \sum_{h=h_0}^{h=h_1} \sigma_{ih}.$$

Intuitively, the dual price associated with a visit of customer i is averaged on the different periods of interval $\{h_0, \ldots, h_1\}$.

5.2 Evaluation of the reduced cost

We now explain how we evaluate the reduced cost of the *td*-shifts r searched in the pricing problem. Given $i \in \mathcal{V}^1$ visited by r and $s \in \mathcal{T}rip_r$ the trip of r in which i is visited, the

dual constraint derived from primal variable q_{irt_0} in the relaxed formulation is:

$$\hat{\gamma}_{t_0} + \gamma_{t_0rs} - \beta_{ri} - \hat{\sigma}_i \geq Z_{ir}$$

and given $i \in \mathcal{V}^2$ visited by r, the dual constraint derived from primal variable q_{irt_0} is:

$$\hat{\gamma}_{t_0} + \gamma_{t_0rs} + \beta_i^{max} - \beta_i^{min} \geq Z_{ir}$$

Because we are only interested in finding td-shifts not in $\mathcal{R}_k$, we assume that variable γ_{t_0rs} is not known. Following the approach presented in Section 4.2, we set:

$$\begin{aligned}
\gamma_1 &= \max_{i \in \mathcal{V}_r^1} Z_{ir} - \hat{\gamma}_{t_0} + \hat{\sigma}_i \\
\gamma_2 &= \max_{i \in \mathcal{V}_r^2} Z_{ir} - \hat{\gamma}_{t_0} - \beta_i^{max} + \beta_i^{min} \\
\gamma_{t_0rs} &= \max(\gamma_1, \gamma_2, 0)
\end{aligned}$$

Then, to avoid managing values γ_{trs} for $t \neq t_0$, we set:

$$\beta_{ri} = 0$$

Note that following the scheme developed in Section 4.2, we would have set $\beta_{ri} = \min_{t \in \mathcal{T}} \hat{\gamma}_t + \gamma_{trs} - \hat{\sigma}_i - Z$, but 0 also ensures the validity of dual constraints associated with dual variables q_{irt_0}.

The goal of the pricing problem is to find td-shifts (r, d_0, t_0), starting at time m_0 and ending not after time m_1, such that:

$$\begin{aligned}
c_{rd_0t_0} + \sum_{\{m \in \mathcal{S}t(\mathcal{R}_k^{d_0}) : r \in \mathcal{K}_m^{d_0}\}} \pi_m^{d_0} + \sum_{\{m \in \mathcal{S}t(\mathcal{R}_k^{t_0}) : r \in \mathcal{K}_m^{t_0}\}} \pi_m^{t_0} - \sum_{i \in \mathcal{V}_r} \alpha_i \\
- \sum_{i \in \mathcal{V}_r^1} \hat{\alpha}_i - Q_{t_0} n_r^{src} \hat{\gamma}_{t_0} - Q_{t_0} \sum_{s \in \mathcal{T}rip_r} \gamma_{t_0rs} < 0
\end{aligned} \tag{28}$$

It is noteworthy that this implicit relaxation can also be interpreted as a stabilization method. Indeed, when solving RMP_k with the simplex method, dual variables receive extreme values. An important drawback with dual variables indexed by time is that a dual variable can receive a large value while the dual variables in adjacent periods receive zero. Then, with the column generation mechanism, the dual value can be restricted for the specific period that received a large value, but this value can (repeatedly) be moved to a close period. Then many iterations are potentially required to attain a good evaluation of the dual prices. One advantage of our implicit relaxation scheme is to average the value of these dual variables.

5.3 Dynamic programming algorithm

The pricing problem is addressed through dynamic programming as an elementary shortest path problem with resource constraints. The algorithm follows the approach described in Feillet et al. (2004), except for the label definition, resource extension functions and dominance rule. Before detailing these features of the algorithm, we introduce additional notation (see Table 7) that completes the notation already reported in Tables 1 and 2.

dc_d	working cost (per working time unit) for driver $d \in \mathcal{D}$
tc_t	distance cost (per distance unit) for trailer $t \in \mathcal{T}$
s_i	setup time for location i (0 if i is the base)
t_{ij}	travel time from location i to location j
d_{ij}	distance from location i to location j
$t_d^{layover}$	duration of a layover for driver $d \in \mathcal{D}$
$c_d^{layover}$	cost of a layover for driver $d \in \mathcal{D}$
T_d	maximum driving duration for driver $d \in \mathcal{D}$
tw_i	number of time windows for customer $i \in \mathcal{V}$
$[a_{ik}, b_{ik}]$	k^{th} time window for customer $i \in \mathcal{V}$ $(1 \leq k \leq tw_i)$

Table 7: Notation (other data)

Labels are defined with ten attributes plus a vector of size $|\mathcal{V}|$, $L =$ *(ending vertex, cost, time, driving time, minimal quantity, gamma, sources, trips, layover, layover customer,visited[])*, that can be interpreted as shown in Table 8. Denoting *base* the index of the base in the location set, the algorithm is initialized with label:

$$L = (base, \sum_{m=m_0}^{m_0+rest_{d_0}-1} \pi_m^{d_0}, m_0, T_{d_0}, 0, 0, 0, 0, 0, 0, [0, \ldots, 0])$$

.

Resource extension functions are defined in Tables 9 and 10. In these tables, label $L = (i, c, t, dt, minq, gamma, nbs, nbt, lay, laycust, visited[])$ is extended to vertex $j \in \mathcal{V}$ to form label $L' = (i', c', t', dt', minq', gamma', nbs', nbt', lay', laycust', visited'[])$. The last column in Table 9 indicates if the formula applies to the source (s), the base (b) or both (s+b). The last column in Table 10 is empty when the formula always applies. (lay.) indicates that it applies when a layover is carried out, and (no lay.) stands for the case with no layover.

The inclusion of a layover is somewhat complex. Three cases are considered. When the waiting time before the opening of a time window exceeds the layover duration, a layover is imposed (formally when $t + s_i + t_{ij} + t_{d_0}^{layover} \leq a_{jk}$ with $k = min\{l : 1 \leq l \leq tw_j$ and $t + s_i + t_{ij} \leq b_{jl}\}$, that is, k indicates the earliest time window allowing to reach j from L). Inserting a layover in this case is imposed by the REC-IRP definition. The two

ending vertex	last visited vertex
cost	evaluation of the reduced cost
time	starting time of the service at *ending vertex* if it is a source or a customer, arrival time otherwise (base), expressed in minutes
driving time	remaining allowed driving time, expressed in minutes
minimal quantity	minimal quantity to be delivered in the current trip, according to values q_i^{min}
gamma	current value for dual variable γ_{t_0rs}, accordingly with the vertices already visited in the current trip
sources	number of visits to a source
trips	number of trips (started or finished)
layover	boolean value indicating if a layover has been planned
layover customer	boolean indicating if a layover customer has been visited
visited[u]	boolean indicating if customer $u \in \mathcal{V}$ has been visited or not

Table 8: Label definition

other cases result from a simplification. When j is a layover customer and L contains no layover ($lay = false$), we decided to insert a layover before (second case) or after (third case) j. First, the earliest time window $k = min\{l : 1 \leq l \leq tw_j \text{ and } t + s_i + t_{ij} \leq b_{jl}\}$ in j is computed. If the layover can be inserted before the closing time of the time window ($t + s_i + t_{ij} + t_{d_0}^{layover} \leq b_{jk}$), it is inserted before, otherwise it will be inserted when L' is extended, after j.

The extension of label L to j is also subject to some few feasibility tests: i) If j is a customer, the extension is not allowed if $visited[j] = 1$; ii) the extension with a layover is not allowed when $lay = 1$ (note that only the first case above is concerned); iii) a time window k computed as explained in Table 10 must exist ($t+s_i+t_{ij} \leq b_{jtw_j}$). If j is the base or a source, condition $t' \leq m_2$ is required. If j is the base the combination $lay = 1$ and $laycust = 0$ is not allowed. In all cases (customer, source or base), the remaining driving time should be nonnegative: $dt' \geq 0$, and the minimal delivered quantity mq' should not exceed Q_{t_0}.

Regarding the dominance rule, we propose very simplified (non-exact) conditions. Given two labels L_1 and L_2, $L_1 = (i_1, c_1, t_1, dt_1, minq_1, gamma_1, nbs_1, nbt_1, lay_1, laycust_1, visited_1[])$ dominates $L_2 = (i_2, c_2, t_2, dt_2, minq_2, gamma_2, nbs_2, nbt_2, lay_2, laycust_2, visited_2[])$ if:

1. $i_1 = i_2$
2. $c_1 \leq c_2$
3. $t_1 \leq t_2$
4. $dt_1 \geq dt_2$

end. ver.	$i' = j$	(s+b)
cost	$c' = c + dc_{d_0} \times (s_i + t_{ij}) + tc_{t_0} \times d_{ij} + \sum_{m=t}^{t'-1} \pi_m^{t_0} + \sum_{m=t+rest_{d_0}}^{t'+rest_{d_0}-1} \pi_m^{d_0} - Q_{t_0}\hat{\gamma}_{t_0}$	(s)
	$c' = c + dc_{d_0} \times (s_i + t_{ij}) + tc_{t_0} \times d_{ij} + \sum_{m=t}^{t'} \pi_m^{t_0} + \sum_{m=t+rest_{d_0}}^{t'+rest_{d_0}} \pi_m^{d_0}$	(b)
time	$t' = t + s_i + t_{ij}$	(s+b)
dr. time	$dt' = dt - t_{ij}$	(s+b)
min. qtty	$mq' = 0$	(s+b)
gamma	$gamma' = 0$	(s+b)
sources	$nbs' = nbs + 1$	(s)
	$nbs' = nbs$	(b)
trips	$nbt' = nbt$	(s+b)
layover	$lay' = lay$	(s+b)
lay. cust.	$laycust' = laycust$	(s+b)
visited[]	$visited'[u] = visited[u]$ for all $u \in \mathcal{V}$	

Table 9: Resource extension functions to a source or to the base

end. ver.	$i' = j$	
cost	$c' = c + dc_{d_0} \times (t' - t) + tc_{t_0} \times d_{ij} + \sum_{m=t}^{t'-1} \pi_m^{t_0} + \sum_{m=t+rest_{d_0}}^{t'+rest_{d_0}-1} \pi_m^{d_0} - \alpha_j - \hat{\alpha}_j + gamma \times Q_{t_0} - gamma' \times Q_{t_0}$	(no lay.)
	$c' = c + dc_{d_0} \times (t' - t) + tc_{t_0} \times d_{ij} + \sum_{m=t}^{t'-1} \pi_m^{t_0} + \sum_{m=t+rest_{d_0}}^{t'+rest_{d_0}-1} \pi_m^{d_0} - \alpha_j - \hat{\alpha}_j + gamma \times Q_{t_0} - gamma' \times Q_{t_0} - dc_{d_0} \times t_{d_0}^{layover} + c_{d_0}^{layover}$	(lay.)
time	$t' = \max(t + s_i + t_{ij}, a_{jk})$ with $k = min\{l : 1 \leq l \leq tw_j$ and $t + s_i + t_{ij} \leq b_{jl}\}$	(no lay.)
	$t' = \max(t + s_i + t_{ij} + t_{d_0}^{layover}, a_{jk})$ with $k = min\{l : 1 \leq l \leq tw_j$ and $t + s_i + t_{ij} + t_{d_0}^{layover} \leq b_{jl}\}$	(lay.)
dr. time	$dt' = dt - t_{ij}$	(no lay.)
	$dt' = \min(T_{d_0}, T_{d_0} + dt - t_{ij})$	(lay.)
min. qtty	$mq' = mq + q_j^{min}$	
gamma	$gamma' = \max(gamma, Z - \hat{\gamma}_{t_0} + \hat{\sigma}_j)$ if $j \in \mathcal{V}^1$, $gamma' = \max(gamma, Z - \hat{\gamma}_{t_0} - \beta_j^{max} + \beta_j^{min})$ if $j \in \mathcal{V}^2$	
sources	$nbs' = nbs$	
trips	$nbt' = nbt + 1$ if $i \notin \mathcal{V}$, $nbt' = nbt$ otherwise	
layover	$lay' = lay$	(no lay.)
	$lay' = 1$	(lay.)
lay. cust.	$laycust' = laycust$ if j is not a layover customer, $laycust' = 1$ otherwise	
visited[]	$visited'[u] = visited[u]$ for all $u \in \mathcal{V} \setminus \{j\}$, $visited'[j] = 1$	

Table 10: Resource extension functions to a customer $j \in \mathcal{V}$

5. $c_1 + (gamma_1 - gamma_2) \times Q_{t_0} \leq c_2$

The last condition can be explained as follows. When $gamma_1 \leq gamma_2$, the condition is necessary satisfied from the second one. When $gamma_1 > gamma_2$ the reduced cost c_1 benefits from this difference compared to c_2. Indeed, the term $Q_{t_0} \sum_{s \in \mathcal{T}rip_r} \gamma_{t_0 rs}$ is subtracted in the reduced cost evaluation. The maximum future benefit that could be obtained from this term for label L_2 and not be obtained to L_1 is $(gamma_1 - gamma_2) \times Q_{t_0}$.

6 Increasing horizon heuristic

In order to deal with the large time horizon of the instances provided for the ROADEF/EURO challenge, we designed an *iterative* increasing horizon heuristic. The principle of this heuristic is to progressively construct a solution from the beginning to the end of the planning horizon.

We divide the time horizon in three parts. The first part goes from the beginning of the horizon to, let say, period H_1. This is the *active* part where the algorithm takes decisions, *i.e.,* determines the *td*-shifts that will be part of the solution. In order not to make blind choices, constraints (consumption of VMI customers, call-in orders, time windows, etc.) are taken into account until period $H_2 > H_1$. Finally, to reduce the complexity of the problem, consumption of customers from $H_2 + 1$ to H is not considered.

When the algorithm has computed the solution in $[0, H_1]$, the values of H_1 and H_2 are increased and the procedure is repeated. The active part of the horizon is still $[0, H_1]$, but now it is wider. That is why we call this approach a *increasing horizon* heuristic instead of a more classic but misleading *rolling horizon* algorithm.

The column generation framework presented in the previous section should be integrated in a branching scheme in order to be able to solve the REC-IRP. However, the large size of instances proposed for the ROADEF/EURO challenge makes this option irrelevant. As a consequence, we select *td*-shifts,that will be part of the solution, using a variable fixing procedure. This heuristic will be described in Section 6.4.

In order to reduce the computation times, we introduced in our procedure what we called the *catalog* $\mathcal{CAT}$. $\mathcal{CAT}$ is a long-term memory which stores promising *templates* identified so far by the algorithm. Templates are sequences of loading and delivery operations such that the time between each pair of operations is determined, but the starting time has not been set yet. More precisely, when we assign a starting time to a *template* it becomes a *shift*, while assigning a driver and a trailer to a *shift* makes it a *td-shift*.

The determination of a *td*-shift from a template and the exact evaluation of its reduced cost can be done very efficiently and in a much quicker way than calling the implicit-relaxation-based dynamic programming procedure presented in Section 5. As soon as a *td*-shift is generated (with constructive heuristics explained in Section 6.3.1, or with

the dynamic programming pricing procedure presented in Section 4), the corresponding template is added to $\mathcal{CAT}$. It will be considered to generate new *td*-shifts by assigning to the template an effective starting time, a trailer and a driver. Note than we can obtain different shifts from the same template considering different starting times. Similarly, we can obtain different *td*-shifts from the same shift taking into account different trailer-driver pairs.

Algorithm 1 Increasing horizon heuristic

1: Initialize catalog $\mathcal{CAT}$
2: $H_1 \leftarrow P^{rh}_{short}$, $H_2 \leftarrow P^{rh}_{long}$
3: Initialize RMP with parameters $(\mathcal{CAT}, H_1, H_2)$
4: **while** $H_1 \leq H$ **do**
5: Solve RMP and apply dynamic programming (Section 5) to add new templates to $\mathcal{CAT}$
6: Solve RMP and, if needed, recover feasibility (Section 6.5)
7: **repeat**
8: Solve RMP and add new *td*-shifts to RMP from $\mathcal{CAT}$ (Section 6.3.4)
9: **until** $P^{cat}_{nbSearch}$ repetitions has been performed
10: Over the horizon $[0, H_1]$ fix x_{rdt} variables to 1 in RMP (Section 6.4)
11: Solve RMP and, if needed, recover feasibility (Section 6.5)
12: Reduce RMP if it contains too many variables (Section 6.6)
13: Increment H_1 and H_2
14: Repopulate $\mathcal{CAT}$
15: **end while**
16: Improve solution

The framework of the solution procedure is presented in Algorithm 1. First, $\mathcal{CAT}$ is initialized with a constructive heuristic presented in Section 6.3.1 (Line 1) and the values of H_1 and H_2 are set to P^{rh}_{short} and P^{rh}_{long} respectively (Line 2). The first RMP is initialized with a set of *td*-shifts (Line 3). Modifications generated to RMP in order to deal with the increasing horizon are described in Section 6.1.

The goal of the main loop of Algorithm 1 (Lines 4 to 15) is to generate *td*-shifts over horizon $[0, H_2]$ and to fix *td*-shifts during the current active part of the horizon, namely $[0, H_1]$. New templates are added to $\mathcal{CAT}$ with the dynamic programming pricing procedure presented in Section 5 (Line 5). This is detailed in Section 6.3.2.

Whenever RMP is not feasible (*i.e.,* based on definition introduced in Section 3.6, the optimal solution of RMP has non-zero artificial variables), the procedure presented in Section 6.5, is applied to recover feasibility, namely an attempt is made to obtain a solution with all artificial variables set to zero. Then, RMP is solved $P^{cat}_{nbSearch}$ times. At each step, new *td*-shifts are generated from $\mathcal{CAT}$ using the procedure presented in Section 6.3.4 to improve the quality of the *td*-shifts considered in RMP.

To generate integer solutions we apply the variable-fixing procedure presented in Section

6.4. It sets x_{rdt} variables to 1 and determine the td-shifts that are included the solution. More precisely, x_{rdt} variables of the current RMP associated with td-shifts starting in $[0, H_1]$ are considered as binary. A mixed integer version of RMP is then solved. Then, variables x_{rdt} that equal to 1 are fixed and the corresponding td-shifts are part of the final solution.

It may happen that once variables have been fixed, RMP becomes infeasible. In such a case, the procedure described in Section 6.5 is applied again to recover feasibility.

Then, the horizon is increased and values H_1 and H_2 are incremented by a value P_+^{rh} (or set to H if the increment would exceed H). At this point we evaluate the size of RMP, *i.e.,* the td-shifts considered by the current RMP. If it is too large, some variables are excluded, as detailed in Section 6.6. Finally, the catalog $\mathcal{CAT}$ is repopulated calling the implicit-relaxation-based dynamic programming procedure.

Once the whole horizon has been considered, the algorithm provides with a solution for the REC-IRP. A final step, presented in Section 6.7, is then applied to try to improve its quality.

6.1 Restricted master problem on a limited time horizon

Since we consider a restricted time horizon, the RMP has to be defined only on a part of the whole horizon. Here we present the modifications we apply to RMP.

- for all $i \in \mathcal{V}^1$, for all periods $h > H_2$:
  - variable S_h^i is unbounded;
  - Constraint (16) is relaxed;
  - if $\pi_i(h) \geq H_2$, the corresponding Valid inequality (23) is relaxed;
- for all $i \in \mathcal{V}^1$:
  - the value of n_i involved in Valid inequalities (22), is computed over $[0, H_2]$;
- for all $i \in \mathcal{V}^2$ such that earliest time for delivery is in $[H_2 + 1, H]$:
  - Constraints (12) and (13) are relaxed;
  - Valid inequality (21) is relaxed.

6.2 Management of the objective function

One of the specific feature of the REC-IRP is the fractional objective function that we manage using the Dinkelbach's algorithm (Section 3.4). In this section we explain the implementation details related to our procedure and how we modify the objective function to help find feasible solutions.

6.2.1 Application to the increasing horizon heuristic

During the resolution of the REC-IRP with the increasing horizon heuristic we initially set the value of Z, the parameter used to linearize the objective function, to $Z_0 = 0.01$.

Then, the value of Z is updated at each iteration of the algorithm, before calling the fixing procedure described in Section 6.4. The ratio of the objective function is updated $P^{ratio}_{iterMax}$ times according to the Dinkelbach's rule (3.4) and, after each update, the RMP is solved again. Whenever the current value of $Z - \frac{N(x)}{D(x)}$ is lower than a parameter P^{ratio}_{eps}, the procedure stops.

Note that this approach is consistent with the implementation proposed in Garaix et al. (2011), while Archetti et al. (2016) only update Z when the parametric linear program is completely solved (by column generation).

6.2.2 Early delivery

During our preliminary experiments, we noticed that, on some instances, the algorithm worked correctly until a certain time period in the horizon. Then it encounters difficulties to find *td*-shifts to be part of the solution and to build a solution over the whole horizon. We guessed that this problem was due to a lack of anticipation in serving customers that would eventually run out of stock. Once such a problem appears, the procedure tries to insert a particular *td*-shift in the horizon unsuccessfully since it is blocked by other already fixed *td*-shifts.

To circumvent this difficulty, we slightly modify the objective function introducing the *delivery profits* c_h. c_h indicates the profit raised when delivering a unit of goods to a customer at a specified time $h \in \mathcal{H}$. The expected benefit was to favor early delivery, and so, help finding feasible solutions. The values of c_h are as follows:

- $c_h = 1$ for $h \leq H_1$;
- $c_h = 0$ for $h \geq H_2$;
- c_h decreases linearly between H_1 and H_2.

6.3 Management of the catalog $\mathcal{CAT}$

6.3.1 Initialization of the catalog $\mathcal{CAT}$

$\mathcal{CAT}$ is first initialized with basic templates, *i.e.,* templates including a visit to a source and a visit to only one customer. Obviously, only *feasible* template are kept into $\mathcal{CAT}$. By feasible, we mean a template from which feasible *td*-shifts can be obtained.

Successively, two templates generation procedures, one based on driver's time windows and one based on customer requirements are executed. These procedures are detailed thereafter. The procedure based on customer requirements is first executed once, then the one based on driver's time windows is executed at most $P_{nbIter}^{initCat}$ times, and stops when $\mathcal{CAT}$ contains more than $P_{sizeMax}^{initCat}$ templates.

Initialization procedure based on driver's time windows

This procedure is based on the exploration of the set of driver time windows. It may happen that a driver time window is as wide as the whole horizon. This represents the availability of the driver during the entire planning horizon. Since two shifts assigned to the same driver d must be separated by at least $rest_d$ units of time, the driver time window is divided into smaller time windows in order to: (1) allow to deliver a reasonable number of customers together with a layover (more precisely the time window length is set to the average maximum driving time, plus the layover duration, plus 10 times the setup time); (2) satisfy the minimal time that has to take place between two consecutive time windows.

For each time window, considered in chronological order, and all trailers compatible with the driver, the following steps are carried out:

1. identify *critical* customers, *i.e.,* a set of customers compatible with the trailer that must be visited in the time window in order to avoid stockouts;
2. estimate delivery quantities for these customers based on their consumption;
3. generate a set of *td*-shifts that cover all the selected customers, by means of a best-insertion-like procedure as follows:
   - first, the customer is tried to be inserted into a trip of an existing shift;
   - if this is not possible, a new trip with a visit to the source and then to the customer is created;
4. update inventory levels accordingly. As a consequence, the set of critical customers may change;
5. for each *td*-shift the corresponding template is obtained by removing the association to the driver and to the trailer and by unfixing the starting time. The template is then inserted into $\mathcal{CAT}$.

In order to generate several templates with the same procedure, the following steps are randomized:

- for each time window, the order of the trailers compatible with the driver is shuffled;

- the order of the customers in the best-insertion-like procedure is shuffled;
- in the selection of the customers, it is possible to randomly add customers that do not need to be visited to avoid stockouts;
- when the set of shifts has been generated, it is possible to try to add other customers using the best-insertion-like procedure.

Initialization based on customer requirements

The procedure described here considers only the customers that must be delivered during the horizon in order to avoid stockouts. For each of them, we consider a delivery-time window. It starts when the minimal quantity can be delivered and ends when the customer runs out of stock. The customers are then sorted based on the closing date of the delivery time windows.

The best-insertion-like procedure is applied on this set of customers in order to obtain a set of *td*-shifts. Among all *td*-shifts that have been produced by the procedure, only one is kept. Delivery time windows are then re-computed for the customers served by this *td*-shift. If some customers do not require additional deliveries, they are not anymore considered. The procedure is repeated until sufficient quantities have been delivered to all customers to avoid stockouts over the time horizon. For each of the *td*-shifts that are obtained the corresponding template is inserted into $\mathcal{CAT}$.

6.3.2 Insertion of new templates with the impicit-relaxation-based dynamic programming procedure

In order to populate $\mathcal{CAT}$ with new templates, the dynamic programming pricing procedure is called. This procedure depends on a few parameters that define the subset of *td*-shifts that is explored: a driver d_0, a trailer t_0, a starting time m_0 and a maximum ending time m_1. To diversify the templates generated, the procedure is applied for all the feasible trailer-driver pairs. For each pair, several values of m_0 and m_1 are considered in order to cover the horizon $[0, H_2]$. We consider values of m_0 and m_1 such that $m_1 - m_0$ is large enough to make a layover shift. As a consequence, we set $m_1 - m_0$ to twice the maximum driving time, plus the layover duration, plus 10 times the setup time to allow enough time for the operation. Moreover, we take into account the *td*-shifts that have already been fixed. For each trailer-driver pair, we compute the availability time windows, i.e., where they do not operate fixed *td*-shifts. Then, we ensure that m_0 and m_1 values are inside these time windows. Algorithm 2 details the search for negative reduced-cost shifts using the implicit-relaxation-based dynamic programming procedure.

Algorithm 2 Generate shifts with negative RC

$\mathcal{S} \leftarrow \emptyset$
for all $d \in \mathcal{D}$ **do**
  for all $t \in \mathcal{T}_d$ **do**
    $maxDuration \leftarrow$ Twice the maximum driving time, plus the layover duration, plus 10 setup times
    $\mathcal{TW} \leftarrow$ time windows where d and t are available (based on the current td-shifts fixed into the RMP)
    for all $tw \in \mathcal{TW}$ **do**
      $m_0 \leftarrow st(tw)$
      $m_1 \leftarrow \min\{m_0 + maxDuration, end(tw)\}$
      while $m_0 < H_2$ **and** $m_0 < m_1$ **do**
        $\mathcal{S}_{DYN} \leftarrow td$-shifts from dynamic programming (m_0, m_1, d, t)
        $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{DYN}$
        $end \leftarrow \max_{(r,d,t)\in\mathcal{S}_{DYN}}\{et_r\}$
        $m_0 \leftarrow end + gap(d)$
        $m_1 \leftarrow \min\{m_0 + maxDuration; end(tw)\}$
      end while
    end for
  end for
end for
return $\mathcal{S}$

6.3.3 Pricing of the columns in the catalog $\mathcal{CAT}$

For each template $\tilde{r}$ stored in $\mathcal{CAT}$, we evaluate the potential benefit of including it into the model, namely, to create a td-shift r based on template $\tilde{r}$. We proceed as follows. A shift $\bar{r}$ is obtained by postponing by an integer number of periods over $[0, H_2]$, the template $\tilde{r}$. The scheduling of the operations is not modified. The procedure considers all time-shiftings that satisfy time constraints, i.e., for every customer the delivery occurs during one of its time window).

Then, for each of these shifts $\bar{r}$, compatible trailer-driver pairs are determined in order to generate a td-shift r. To this aim, we take into account the current partial solution. When a variable $x_{r'dt}$ is fixed to 1, it means that trailer t is no longer available between $st_{r'}$ and $et_{r'}$. Similarly, driver d is no longer available between $st_{r'} - rest_d$ and $et_{r'} + rest_d$. Hence, driver d and trailer t are compatible with shift $\bar{r}$ if:

- d and t are compatible ($t \in \mathcal{T}_d$);
- t can operate shift $\bar{r}$;
- d can operate shift $\bar{r}$;
- t is available during $[st_{\bar{r}}, et_{\bar{r}}]$;
- d is available during $[st_{\bar{r}}, et_{\bar{r}}]$.

Then, for each feasible td-shift r derived from template $\tilde{r}$, the corresponding exact reduced cost is computed using Expression (27). Only the td-shift with the lowest reduced cost is considered for inclusion into RMP. Note that certain terms of Expression (27) depend only on the sequence of the customers and need to be computed only once.

A parameter P^{eval}_{impr} is used to determine how much improvement in the reduced cost is needed to consider updating the best td-shift generated from template $\tilde{r}$ (the td-shift considered for inclusion into RMP). Since starting times are ordered, this avoid to postpone the starting time of a shift when the impact on the reduced cost is limited.

6.3.4 Selection of td-shifts to enter RMP

To construct shifts from templates in $\mathcal{CAT}$, initially a template is randomly selected. The reduced cost of associated td-shifts is calculated as explained in Section 6.3.3. If the resulting reduced cost is lower than a negative value P^{eval}_{rc}, the td-shift is memorized and will be considered to enter RMP. The procedure terminates when $P^{cat}_{nbTemplates}$ have been memorized or $\mathcal{CAT}$ has been completely scanned. Finally, the $P^{cat}_{nbAddBySearch}$ td-shifts with the lowest reduced cost are inserted into RMP.

6.4 Variable-fixing heuristic

The solution of the current RMP is by nature fractional. In order to obtain a solution for the REC-IRP, we need to retrieve an integer solution with respect to variables x_{rdt}. To this aim, we propose a variable-fixing heuristic. The purpose of this procedure is to select the td-shifts included in the REC-IRP solution by fixing the corresponding x_{rdt} variables to 1.

This variable-fixing heuristic solves a mixed integer programming problem $MIX\text{-}RMP$ derived from the current RMP. To do so, all the x_{rdt} associated with td-shifts that start before H_1 are considered as binary variables. The other variables remains continuous. After solving $MIX\text{-}RMP$, binary variables x_{rdt} with value 1 are fixed to 1 in RM. Note that the mixed integer program is solved with a time limit of P_{tl}^{MIP} seconds, and a relative gap of P_{gap}^{MIP}.

It is noteworthy that each time the variable-fixing heuristic is called, the previous fixing decisions can be updated. Hence a variable previously fixed can be unfixed. Moreover, the reduction of the size of RMP presented in Section 6.6 permits to remove the x_{rdt} with value 0. This allows to maintain a reasonable size of x_{rdt} variables over the horizon $[0, H_1]$, hence to maintain a reasonable number of integer variables when solving the mixed integer program.

After fixing the x_{rdt} variables over the horizon $[0, H_1]$, it could happen, even after applying the recovery procedure presented in Section 6.5, that RMP does not admit a solution without non-zero artificial variables. In this case we detect an incorrect fixing strategy and another mixed integer program $MIX\text{-}RMP'$ obtained is solved. $MIX\text{-}RMP'$ is obtained from RMP by considering as binary variables all variables in $\mathcal{X}_{fixed}$, where $\mathcal{X}_{fixed}$ contains all the x_{rdt} variables currently fixed. Then, the two following constraints are added:

$$\sum_{x \in \mathcal{X}_{fixed}} x \leq |\mathcal{X}_{fixed}| - 1; \tag{29}$$

$$\sum_{x \in \mathcal{X}_{fixed}} x \geq |\mathcal{X}_{fixed}| - P_{uf}^{MIP}. \tag{30}$$

Constraint (29) forces to unfix at least one variable while Constraint (30) limits the number of unfixed variables. Note that if RMP turns to be still unfeasible, the horizon is increased by incrementing the values of H_1 and H_2.

6.5 Recovery procedure

It may occur that the current RMP only admits a feasible solution where at least one artificial variable is non-zero. That means that the td-shifts together with the variables that have been fixed (see Section 6.4) in the current RMP cannot provide a feasible solution for the REC-IRP. In this case, we propose the following recovery procedure.

The procedure detects the source of infeasibility by generating the set $\mathcal{U}$ of customers associated with a non-zero artificial variable (we recall from Section 3.6 that all the artificial variables are indexed by a customer).

Then $\mathcal{CAT}$ is first scanned focusing only on templates that contain at least a customer in $\mathcal{U}$. The associated $P^{cat}_{nbAddBySearch}$ *td*-shifts with the most negative reduced cost are then added to RMP.

A second set of $P^{cat}_{nbAddBySearch}$ *td*-shifts is added to RMP using the procedure detailed in Section 6.3.4.

Those two searches into $\mathcal{CAT}$ are repeated until RMP is feasible or it has been executed $P^{recover}_{nbIter}$ times. In case RMP is still unfeasible after the $P^{recover}_{nbIter}$ iterations:

- $\mathcal{CAT}$ is repopulated calling the implicit-relaxation-based dynamic programming procedure;
- the whole procedure is applied using the updated catalog.

If RMP is still unfeasible after $P^{recover}_{nbUpdateCat}$ updates of the $\mathcal{CAT}$, at most P^{MIP}_{uf} x_{rdt} variables are unfixed as described in Section 6.4.

6.6 Reduction of the size of the RMP

Each time the current values of H_1 and H_2 are incremented we check if it is possible to reduce the size of RMP. The procedure simply checks if the current solution of RMP admits more than P^{reduc}_{maxNul} variables x_{rdt} set to zero or more than $P^{reduc}_{maxUnfix}$ unfixed variables. In this case, the procedure removes from RMP all variables x_{rdt} set to zero. For each variable x_{rdt} removed from RMP the associated variables q_{irt} and the constraints related to *td*-shift r are also removed.

6.7 Solution improvement

Once a feasible solution has been found, we try to improve the current best solution. We create a set $\mathcal{S}$ containing each *td*-shift included in the current solution. Then, for each *td*-shift in $\mathcal{S}$, we add to $\mathcal{S}$ additional *td*-shifts built as follows:

(1) customers receiving less than 1% of the total quantity delivered in the *td*-shifts are removed;

(2) visits, for which only the minimal quantities are delivered, are removed from the *td*-shift;

(3) the possible initial stop at the source is removed;

P^{rh}_{short}	24
P^{rh}_{long}	120
P^{rh}_{+}	12 or 24
$P^{initCat}_{sizeMax}$	5000
$P^{initCat}_{nbIter}$	50
$P^{cat}_{nbSearch}$	3
$P^{cat}_{nbAddBySearch}$	10
$P^{recover}_{nbUpdateCat}$	2
$P^{recover}_{nbIter}$	10
$P^{cat}_{nbTemplates}$	1000
P^{reduc}_{maxNul}	$\max\left\{100; \frac{1}{2}\text{nb of } x_{rdt}\text{variables}\right\}$
$P^{reduc}_{maxUnfix}$	10
$P^{improve}_{maxSize}$	250
$P^{ratio}_{iterMax}$	10
P^{ratio}_{eps}	10^{-6}
P^{eval}_{impr}	1.01
P^{MIP}_{tl}	300 if $\lvert\mathcal{V}\rvert \leq 100$ and $\mathcal{H} \geq 35$ 1200 if $\lvert\mathcal{V}\rvert \geq 300$ and $\mathcal{H} \geq 20$ 600 otherwise
P^{MIP}_{gap}	0.1
P^{MIP}_{uf}	5

Table 11: Parameters.

(4) a final stop at the source is added (before going back to the base);

(5) new visits are considered by applying a best-insertion like procedure (this step is applied as well to *td*-shifts generated in previous points).

When $\mathcal{S}$ contains more than $P^{improve}_{maxSize}$ *td*-shifts, the process is stopped. The formulation proposed in Section 3 restricted to $\mathcal{S}$ is then solved in the hope of improving the solution quality. In case of improvement, the procedure is repeated.

7 Computational experiments

The algorithm presented in Section 6 was run on a Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz. Cplex 12.6 was used to solve the (mix-integer) linear programs considered in the procedure and described in the previous sections. The parameters used in the algorithm are set as presented in Table 11.

A maximum time is allocated to solve each instance. We distinguish two sets of instances based on their size. Mid-size instances include less than 100 customers, or less

Instance	$\|\mathcal{V}\|$	$\|\mathcal{H}\|$ (in days)	$\|\mathcal{D}\|$	$\|\mathcal{T}\|$	P_+^{rh}	Ratio ($\times$100)	Best known ratio ($\times$100)
V2.13	53	10	5	5	12	3.4803	2.8875
V2.14	53	35	5	5	24	4.7637	3.4971
V2.15	134	10	4	3	12	3.4528	2.4993
V2.16	184	10	7	4	24	1.6826	1.1783
V2.19	53	35	5	5	12	4.1487	3.4022
V2.24	32	10	5	6	24	1.4175	1.1219
V2.25	32	35	5	6	12	1.3194	1.1451
V2.26	32	35	5	6	12	1.3580	1.1281
X.2	184	10	7	4	12	1.6886	1.1799

Table 12: Results on mid-size instances.

than 200 customers with a planning horizon of 10 days. Namely these are instances 13, 14, 15, 16, 19, 24, 25, 26 and X2. The remaining instances are large instances, namely 12, 17, 18, 20, 21, 22, 23, X1, X3, X4 and X5. The CPU time limits were 3 hours for mid-size instances, 6 hours for large instances with less than 20 periods or with less than 10 drivers, and 12 hours for instances with more than 20 periods and more than 10 drivers. In Tables 12 and 13, we report the results obtained with our algorithm. The column *Instance* contains the name of the instance, while the column *Ratio* contains the corresponding value of the logistic ratio (multiplied by 100). Since tests were performed with two different values of P_+^{rh}, the column P_+^{rh} gives the value of this parameter that allows to obtain the corresponding ratio.

The branch-cut-and-price heuristic provides solutions on all small and medium-size instances, and on 9 out of 11 large instances. The solution quality is reasonable for most instances with respect to the best known solution values. However, this is at the expense of large computation times.

8 Conclusions and perspectives

This paper deals with the problem proposed in the context of the ROADEF/EURO challenge. The problem is a rich variant of the so-called Inventory Routing Problem encountered by Air-Liquide.

Even if the designing of exact solution approaches would not be suitable for large size instances as those proposed by the organizers of the ROADEF/EURO challenge, we proposed a branch-cut-and-price framework to tackle the REC-IRP. To deal with real-life instances the cut-and-column generation procedure is developed along with a heuristic pricing algorithm. To produce new columns, the pricing procedure relies on a surrogate relaxation of the constraints. Moreover, a heuristic fixing procedure has been integrated

Instance	$\|\mathcal{V}\|$	$\|\mathcal{H}\|$ (in days)	$\|\mathcal{D}\|$	$\|\mathcal{T}\|$	P_+^{rh}	Ratio ($\times$100)	Best known ratio ($\times$100)
V2.12	324	10	13	15	24	1.6290	1.0024
V2.17	134	35	4	3	24	4.9813	3.2130
V2.18	134	35	4	3	24	4.7547	3.1882
V2.20	184	35	7	4	24	2.6095	1.7486
V2.21	184	35	7	4	24	2.5374	1.6806
V2.22	324	21	13	15	12	1.8819	1.2667
V2.23	324	21	13	15	-	-	1.2603
X.1	324	10	13	15	12	1.5974	1.0042
X.3	134	35	4	3	-	-	3.0760
X.4	324	21	13	15	24	1.8325	1.2633
X.5	324	21	13	15	12	1.8101	1.2965

Table 13: Results on large instances.

into the framework to generate feasible solutions.

The proposed method provides solutions of reasonable quality on all but two instances. However, this is at the expense of large computation times.

References

André, J., Baldacci, R., Fokouop, R., Létocart, L., Traversi, E. and Wolfler-Calvo, R. (2016). An approach based on column generation for solving routing problems with fractional objective function, *ROUTE workshop, Rambouillet, June 1–4*.

Archetti, C., Desaulniers, G. and Speranza, M. G. (2016). Minimizing the logistic ratio in the inventory routing problem, *EURO Journal on Transportation and Logistics* p. to appear.

Bard, J. F. and Nananukul, N. (2010). A branch-and-price algorithm for an integrated production and inventory routing problem, *Computers & Operations Research* **37**(12): 2202–2217.

Charnes, A. and Cooper, W. W. (1962). Programming with linear fractional functionals, *Naval Research logistics quarterly* **9**(3-4): 181–186.

Christiansen, M. and Nygreen, B. (2005). Robust inventory ship routing by column generation, *Column generation*, Springer, pp. 197–224.

Coelho, L. C., Cordeau, J.-F. and Laporte, G. (2013). Thirty years of inventory routing, *Transportation Science* **48**(1): 1–19.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows, *Operations research* **58**(1): 179–192.

Desaulniers, G., Rakke, J. G. and Coelho, L. C. (2015). A branch-price-and-cut algorithm for the inventory-routing problem, *Transportation Science* .

Dinkelbach, W. (1967). On nonlinear fractional programming, *Management Science* **13**(7): 492–498.

Feillet, D., Dejax, P., Gendreau, M. and Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems, *Networks* **44**(3): 216–229.

Feillet, D., Dejax, P., Gendreau, M. and Gueguen, C. (2006). Vehicle routing with time windows and split deliveries, *Technical Paper* **851**.

Feillet, D., Gendreau, M., Medaglia, A. L. and Walteros, J. L. (2010). A note on branch-and-cut-and-price, *Operations Research Letters* **38**(5): 346–353.

Garaix, T., Artigues, C., Feillet, D. and Josselin, D. (2011). Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation, *Computers & Operations Research* **38**(10): 1435–1442.

Hernandez, F., Feillet, D., Giroudeau, R. and Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows, *European Journal of Operational Research* **249**(2): 551–559.

Le, T., Diabat, A., Richard, J.-P. and Yih, Y. (2013). A column generation-based heuristic algorithm for an inventory routing problem with perishable goods, *Optimization Letters* **7**(7): 1481–1502.

Michel, S. and Vanderbeck, F. (2012). A column-generation based tactical planning method for inventory routing, *Operations research* **60**(2): 382–397.