Solving a Continent-Scale Inventory Routing Problem at Renault

Louis Bouvier, Guillaume Dalle, Axel Parmentier, Thibaut Vidal

▶ **To cite this version:**

Louis Bouvier, Guillaume Dalle, Axel Parmentier, Thibaut Vidal. Solving a Continent-Scale Inventory Routing Problem at Renault. 2022. hal-03766779

HAL Id: hal-03766779
https://hal.science/hal-03766779v1

Preprint submitted on 1 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Solving a Continent-Scale Inventory Routing Problem at Renault

Louis Bouvier*[1,2], Guillaume Dalle[1], Axel Parmentier[1], and Thibaut Vidal[3]

[1] *CERMICS, Ecole des Ponts, France*
[2] *Groupe Renault*
[3] *Polytechnique Montréal, Canada*

September 1, 2022

Abstract

This paper is the fruit of a partnership with Renault. Their backward logistic requires to solve a continent-scale multi-attribute inventory routing problem (IRP). With an average of 30 commodities, 16 depots, and 600 customers spread across a continent, our instances are orders of magnitude larger than those in the literature. Existing algorithms do not scale. We propose a large neighborhood search (LNS). To make it work, (1) we generalize existing split delivery vehicle routing problem and IRP neighborhoods to this context, (2) we turn a state-of-the art matheuristic for medium-scale IRP into a large neighborhood, and (3) we introduce two novel perturbations: the reinsertion of a customer and that of a commodity into the IRP solution. We also derive a new lower bound based on a flow relaxation. In order to stimulate the research on large-scale IRP, we introduce a library of industrial instances. We benchmark our algorithms on these instances and make our code open-source. Extensive numerical experiments highlight the relevance of each component of our LNS.

1 Introduction

The inventory routing problem (IRP) arises when a supplier manages the delivery of commodities to its customers on a multiple-day horizon in a centralized manner (Archetti and Speranza 2016). It consists in planning routes to deliver commodities from depots to customers with the objective of minimizing inventory and routing costs. This NP-hard problem has received significant attention in the operations research literature over the past 40 years.

The present paper is motivated by a partnership with Renault, a major European car manufacturer who must routinely solve IRP instances of unprecedented continental scale and complexity as part of their backward logistic problem. Indeed, they receive car parts from suppliers at their plants in packaging, and reuse the latter, which implies the need for backward packaging logistics. The goal of our partnership is to redesign their IRP algorithm. This is challenging because of (1) the size of the resulting instances, with 600 customers and 16 depots on average, (2) the 30 different commodities involved, and (3) the specific challenges that arise from the geography and timescale. When depots and customers are scattered across a whole continent, travel times can last up to ten days. Beyond requiring a long horizon, 21 days in our case, this makes the problem more difficult because classic decoupling results on the IRP, which were exploited in previous algorithms, are no longer valid. For instance, changing the order of the customers along a route impacts the arrival day at each customer and therefore the customer inventory levels. Hence, routes with suboptimal routing cost may be better because of inventory cost, which is not usually the case with the IRP. It can be compared with the continuous-time IRP discussed in M. Savelsbergh and Song (2008) or Lagos,

*Corresponding author: louis.bouvier@enpc.fr

Boland, and M. Savelsbergh (2020). Finally, our partner's supply chain process requires that (4) the solution algorithm should not take more than 90 minutes on our computing cluster.

State of the art exact algorithms rely on branch-and-cut (Archetti, Bertazzi, et al. 2007; Coelho and Laporte 2013; Manousakis et al. 2021) and branch-and-price-and-cut methods (Desaulniers, Rakke, and Coelho 2015) with dedicated valid inequalities. They can optimally solve single-commodity single-depot instances with up to 50 customers, but are not appropriate for our large-scale setting.

Typical heuristics include route-based matheuristics (Fischetti and Fischetti 2016), decomposition matheuristics, and metaheuristics.

In this field, Bertazzi et al. (2019) and Archetti, Boland, and Speranza (2017) are route-based matheuristics. The main idea is to reduce the size of the mixed-integer linear program (MILP) formulation of the IRP, by selecting promising routes heuristically. Although Bertazzi et al. (2019) is dedicated to the multi-depot case, neither of the two papers handles the multicommodity aspect we must face, and their largest instances have up to six days horizon, six depots and 50 customers. The methods cannot be applied directly in our context, because the MILP remains too large, even when we restrict ourselves to "promising routes". We instead adapt their principle to our setting, leading to the "reload fixed-path vehicles" subroutine.

Another common approach is to tackle the IRP through a decomposition (Campbell and M. W. P. Savelsbergh 2004; Cordeau et al. 2015). For instance, first set the quantities to be sent, and then create the routes to respect them. The largest instances solved with this two-step method have a single depot, up to five commodities, a six days horizon and 50 customers. We also adapt it to our setting and use it as an initialization heuristic.

Some metaheuristics have been designed for real world IRP. For instance, Benoist et al. (2011) is a randomized local search to address a large-scale single-commodity IRP with pickups, time windows, driver safety and other constraints that are specific to their use case, but less relevant to ours. Su et al. (2020) address a real-world IRP from the ROADEF IRP-Challenge 2016 available at `https://www.roadef.org/challenge/2016/fr/` with a large neighborhood search based on mathematical programming. As for Benoist et al. (2011), the single-commodity formulation with additional constraints is not adapted to our multicommodity context. Other large neighborhoods are introduced in Nolz, Absi, and Feillet (2014) for a single-depot single-commodity stochastic version of the IRP. They are based on perturbation ideas such as the removal of every customer visit on a particular day, followed by a best-insertion policy. This process of removal and insertion is at the core of our work, but we leverage MILP formulations for the insertion. A kernel search heuristic based on a preliminary tabu search is considered in Archetti, Guastaroba, et al. (2021). In this framework, smaller MILPs with increasing size are solved iteratively to improve an initial solution. Single-depot, single-commodity instances with up to six days horizon and 200 customers are solved. The study Coelho, De Maio, and Laganà (2020) fixes a part of the decision variables, this time based on the problem's main "axes" – that is to say the different types of sites involved in the deliveries, the routing and the inventory aspects. It solves reduced MILPs in a variable neighborhood search, and defines the multi-attribute IRP as a multicommodity, multi-depot and multi-vehicle IRP. This constraint structure is the closest to ours. Nonetheless, the instances are smaller, with up to six days horizon, 50 customers, six depots and three commodities. Besides, this study is restricted to one-day routes, whereas we explicitly deal with routes that last multiple days, which creates an additional combinatorial challenge. Another difference is that it incorporates a heterogeneous fleet with three distinct vehicle types, whereas we consider a homogeneous infinite fleet of vehicles.

Therefore, to the best of our knowledge, no algorithm is known to properly scale to our instances. In this context, our main contributions are the following:

1. We introduce two new large-scale perturbation neighborhoods designed for the multi-attribute IRP. They are based on well-solved MILP formulations, and enable to escape from local minima.

2. We design an efficient large neighborhood search (LNS) built upon these large neighborhoods. We generalize 12 Traveling Salesman Problem (TSP), and Split Delivery Vehicle Routing Problem (SDVRP) (see e.g. Dror and Trudeau (1990) and Archetti and Speranza (2008)) neighborhoods from the literature to our IRP context. We propose a new matheuristic inspired by Archetti, Boland, and Speranza (2017) and Bertazzi et al. (2019) and adapted to our large-scale setting.

3. We compute a new lower bound based on a linear program (LP) relaxation (one flow per commodity). To the best of our knowledge, this relaxation is not considered in the literature. We expect the bound not to be tight, which is a feature of every relaxation in the IRP literature. But it is useful to compare algorithm performance on instances with distinct scales.

4. We provide a publicly available library of realistic multi-attribute IRP instances of a continent scale, as an incentive for further research on the topic.

5. We give access to our open-source Julia (Bezanson et al. 2017) package that implements the ideas of the present paper.

6. We proceed to extensive numerical experiments. Since no algorithm is known to scale to our context, we compare the adapted route-based matheuristic and our large neighborhood search.

We precisely define the problem we consider in Section 2. We then provide an overview of the different solution processes in Section 3. The three next sections 4, 5 and 6 emphasize algorithms details. We eventually highlight our numerical experiments in Section 7.

2 Problem Description

2.1 Notations and Data

Let $\mathbb{Z}^+$ be the set of non-negative integers. For $a \in \mathbb{Z}^+$, we denote by $[a]$ the set $\{1, ..., a\}$. Besides, for $x \in \mathbb{R}$, we define $x^+ := \max(x, 0)$. We denote by $|\mathcal{S}|$ the cardinal of a set or list $\mathcal{S}$. When we explicitly consider a vector $\mathbf{x} = (x_1, ..., x_p)$ of dimension $p \in \mathbb{Z}^+$, we use the notation $\mathbf{x} \in \mathbb{Z}$ or $\mathbf{x} \in \{0, 1\}$ instead of $\mathbf{x} \in \mathbb{Z}^{|p|}$ or $\mathbf{x} \in \{0, 1\}^{|p|}$ respectively. Let M be the set of *commodities*, D the set of *depots* and C the set of *customers*, that respectively *release* and *demand* commodities $m \in M$. The *time horizon* is $T \in \mathbb{Z}^+$ days. At the beginning, each *vertex* v (depot or customer) has an *initial inventory* of commodity m denoted by I^0_{mv}. On each day $t \in [T]$, a customer c demands a quantity b^-_{mct} of commodity m. A depot d releases a quantity b^+_{mdt} of commodity m. We say that a depot $d \in D$ *uses* a commodity $m \in M$ if it has a positive initial inventory or a positive release for m at least once over the horizon. We denote by M_d the set of *commodities used by depot* d. We similarly define M_c as the set of *commodities used by customer* $c \in C$, based on initial inventory and demand. A *maximum inventory capacity* κ_{mvt} is set on the night of each day t per vertex v and commodity m. Below this capacity, no inventory cost is paid. Above, a cost is set to $c^{\texttt{exc}}_{mv}$ per unit, where "`exc`" stands for excess. Besides, a price $c^{\texttt{short}}_{mc}$ is paid per unit of unsatisfied demand for commodity m of customer c, where "`short`" stands for shortage. It corresponds to a soft constraint of non-negativity for the customers' inventories. We approximate commodities and vehicles by one-dimensional objects. We associate a length ℓ_m to each commodity $m \in M$. We consider an infinite fleet of homogeneous vehicles of length L, to deliver the commodities from depots to customers. They are not assigned to a particular depot. A 1D bin packing problem must be solved for vehicle loading. The depots and customers are the vertices $\mathcal{V} = D \cup C$ of a directed graph $\mathcal{D} = (\mathcal{V}, \mathcal{A})$ that we name the *locations graph*. The directed aspect is used to model the fact that transport durations and distances depend on the trip direction. There is an arc $a = (u, v) \in \mathcal{A}$ for each vertex $u \in D \cup C$ and $v \in C$, $v \neq u$. Given a vertex v, we denote by $\delta^+(v)$ the set of arcs outgoing from v, and by $\delta^-(v)$ the set of arcs incoming to v. We associate a *distance* Δ_a (in kilometers) and a *transport duration* τ_a (in hours) to each arc. We assume that the distances satisfy the triangular inequality. When planning a route, a cost is paid per vehicle $c^{\texttt{veh}}$, per stop (customer visited) $c^{\texttt{stop}}$, and per kilometer travelled $c^{\texttt{km}}$. The number of stops must not exceed S_{max}, which is a practical requirement of the car manufacturer. The *limit of driving hours per day* is $\tau_{\max}$. The IRP consists in building a set of routes (see Section 2.2) to deliver commodities from depots to customers, minimizing the sum of the routing, inventory and shortage costs and respecting feasibility constraints detailed in Sections 2.3 and 2.2.

2.2 The Route Structure

An admissible path $P = (v_0, v_1, ..., v_k)$ in the locations graph is an elementary path, i.e. a path with pairwise distinct vertices. It starts from a depot $v_0 \in D$, and visits customers $(v_1, ..., v_k) \in C^k, v_i \neq v_j$. We have a limit S_{max} to the number of customers visited:

$$|P| \leq S_{max} + 1. \tag{1}$$

We highlight the fact that a path does not end at its starting depot. Let $\mathcal{P}$ be the set of admissible paths, and $A(P)$ the set of arcs in a path P. A *route* r is a "timed and loaded path". It is a tuple $r = (t^r, P^r, \mathbf{q}^r)$ where:

- $t^r \in [T]$ is the day of the departure.
- $P^r = (v_0^r, v_1^r, ..., v_k^r) \in \mathcal{P}$ is the admissible path followed.
- $\mathbf{q}^r = (q_{ms}^r)_{m \in M, s \in [|P^r|-1]} \in (\mathbb{Z}^+)^{|M| \times (|P^r|-1)}$ are the quantities delivered, for each commodity $m \in M$ and to each customer v_s^r for $s \in [|P^r| - 1]$.

The total load must not be larger than the vehicle capacity L, which can be written as:

$$\sum_{m \in M} \ell_m \left(\sum_{s \in [|P^r|-1]} q_{ms}^r \right) \leq L. \tag{2}$$

Given a route r, and the transport durations τ_a for $a \in \mathcal{A}$, we can compute the arrival day t_s^r at the customer v_s^r for $s \in [|P^r| - 1]$ as follows. We first compute the cumulated transport duration in hours up to customer v_s^r, with $\tau_0^r = 0$:

$$\tau_s^r = \tau_{s-1}^r + \tau_{(v_{s-1}^r, v_s^r)}, \quad \forall s \in [|P^r| - 1]. \tag{3}$$

Then, the actual day t_s^r of arrival at customer v_s^r takes pauses into account:

$$t_s^r = t^r + \left\lfloor \frac{\tau_s^r}{\tau_{\max}} \right\rfloor, \quad \forall s \in [|P^r| - 1]. \tag{4}$$

Equation (4) means that when a driver exceeds the driving time limit per day $\tau_{\max}$, a pause is made until the next day. The vehicle then goes on from the location of the pause. Since in practice routes start from depots in the morning and the deliveries are only available in the evening at the customers, it is indeed a floor and not a ceiling function we consider in Equation (4). A route must also visit every stop before the horizon T, which can be written as:

$$t_s^r \leq T, \quad \forall s \in [|P^r| - 1]. \tag{5}$$

We henceforth denote by $\mathcal{R}$ the set of admissible routes. A *direct route* follows a path $P = (d, c)$ from a depot $d \in D$ to a customer $c \in C$ in the locations graph. It has only one arc.

2.3 Inventory Routing Formulation

The variables we consider are the following. We denote by z_{mdt}^- the quantity of commodity m sent from depot d on day t, and by z_{mct}^+ the quantity of commodity m delivered to customer c on day t. Let I_{mvt} be the inventory of commodity m at vertex v on the evening of day t. We last denote by x_r the number of vehicles

following route r. We then consider the MILP formulation:

$$\min_{\mathbf{x},\mathbf{z},\mathbf{I}} \quad \sum_r x_r \left(c^{\texttt{veh}} + c^{\texttt{stop}}(|P^r| - 1) + c^{\texttt{km}} \sum_{a \in A(P^r)} \Delta_a \right) \qquad \text{(multi-attribute-IRP)}$$

$$+ \sum_{d,t,m} c^{\texttt{exc}}_{md} \left(I_{mdt} - \kappa_{mdt} \right)^+ \tag{6a}$$

$$+ \sum_{c,t,m} c^{\texttt{exc}}_{mc} \left(I_{mct} - \kappa_{mct} \right)^+ + c^{\texttt{short}}_{mc} (b^-_{mct} - I_{mc(t-1)})^+ \tag{6b}$$

$$\text{subject to} \quad z^-_{mdt} = \sum_{\substack{r, \\ v^r_0 = d, \\ t^r = t}} \sum_{s \in [|P^r|-1]} x_r q^r_{ms}, \quad \forall m \in M, \quad \forall d \in D, \quad \forall t \in [T] \tag{6c}$$

$$z^+_{mct} = \sum_r \sum_{s \in [|P^r|-1], t^r_s = t, v^r_s = c} x_r q^r_{ms}, \quad \forall m \in M, \quad \forall c \in C, \quad \forall t \in [T] \tag{6d}$$

$$I_{mdt} = I_{md(t-1)} + b^+_{mdt} - z^-_{mdt}, \quad \forall m \in M, \quad \forall d \in D, \quad \forall t \in [T] \tag{6e}$$

$$I_{md0} = I^0_{md}, \quad \forall m \in M, \quad \forall d \in D \tag{6f}$$

$$I_{mct} = \left(I_{mc(t-1)} - b^-_{mct} \right)^+ + z^+_{mct}, \quad \forall m \in M, \quad \forall c \in C, \quad \forall t \in [T] \tag{6g}$$

$$I_{mc0} = I^0_{mc}, \quad \forall m \in M, \quad \forall c \in C \tag{6h}$$

$$\mathbf{x} \geq 0, \ \mathbf{z} \geq 0, \ \mathbf{I} \geq 0 \tag{6i}$$

$$\mathbf{x} \in \mathbb{Z}, \ \mathbf{z} \in \mathbb{Z}, \ \mathbf{I} \in \mathbb{Z} \tag{6j}$$

We notice that, given an IRP instance and the route variables $\mathbf{x}$, we can deduce the quantities sent or received $\mathbf{z}$ and the inventory $\mathbf{I}$. The latter are useful to express the IRP as an MILP.

Objective function. The first sum models a cost per vehicle, per stop and per kilometer travelled. The second one is related to the excess inventory during the nights at the depots. The third one has both an excess inventory and a shortage part. The quantity $\left(b^-_{mct} - I_{mc(t-1)}\right)^+$ is a substitute that is bought separately when a shortage appears.

Constraints. The $\mathbf{x}$ variable is used to count the number of vehicles that follow the admissible routes defined in Section 2.2. Equation (6c) is used to bind the total quantities that are sent from each depot to the route deliveries at each customer. Equation (6d) links the total quantities received per customer to the route deliveries. Constraints (6e)-(6f) define the inventory dynamics at the depots, and (6g)-(6h) at the customers. We highlight we cannot deliver a commodity to a customer that does not need it – in the sense of C_m – because the maximum inventory capacity is set to zero and the excess inventory cost to infinity. The MILP (multi-attribute-IRP) is intractable over our instances, we instead suggest several heuristic approaches in the next section.

3 Overview of the Algorithms and General Concepts

We emphasize the main principles of the algorithms in Section 3.1 before going into the details of each of their components. We also introduce generic flow graphs and formulations in Section 3.2, concepts useful in the rest of the present paper.

3.1 Overview of the Algorithms

We compare three algorithms with increasing degrees of sophistication and performance: an *initialization + local search* algorithm to quickly derive non-trivial IRP solutions, a *route-based matheuristic*, and our LNS. The two first algorithms are adapted from frameworks of the literature, the last one is our main contribution.

3.1.1 Subroutines

Our algorithms are illustrated on Figure 1 and combine five subroutines. We call inner iteration an iteration within any subroutine, and outer iteration a path through the four types of neighborhoods in the LNS (see the loop in Figure 1). The first subroutine builds an initial solution. 1) The *flow relaxation + bin packing* (`flow relaxation`, `bin packing`) subroutine solves a flow relaxation – thus an LP – per commodity and deduces direct routes by approximately solving bin packing problems to respect vehicle capacity. The other four subroutines improve or perturb an existing solution, and can be applied any number of times in any order. 2) The *routing local search* (`routing local search`) subroutine takes a random subset of routes and applies a local search with TSP and SDVRP neighborhoods (see Section 5.1). 3) The *reload fixed-path vehicles* (`reload fixed path vehicles`) subroutine solves an MILP per depot to re-optimize the load of the routes starting from it. The MILP is solved with a very low gap threshold in the route-based matheuristic, and up to a larger gap threshold in the LNS, adding a time limit. 4) The *customer reinsertion* (`customer reinsertion`) subroutine removes a customer from every delivery of a solution and solves an MILP to reinsert it in the existing routes, also creating new direct routes. 5) The *commodity reinsertion* (`commodity reinsertion`) subroutine removes a commodity from every delivery of a solution and solves an MILP to reinsert it. These MILPs are solved up to a gap and time limit. We see the last four subroutines as local search procedures, and call them with a customizable number of inner iterations per outer iteration. In Section 4 we detail the flow relaxation + bin packing subroutine. The routing local search subroutine is described in Section 5. The three large neighborhoods – reload fixed-path vehicles, customer and commodity reinsertion – are detailed in Sections 6.1, 6.2 and 6.3 respectively.

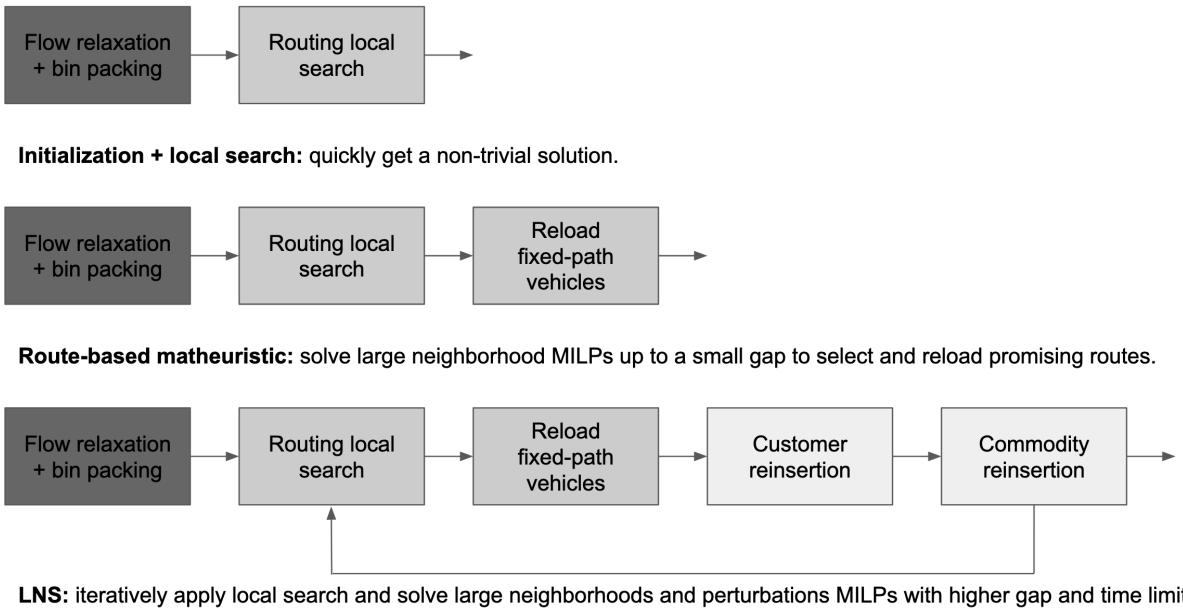


Figure 1: Different algorithms ordered by degree of sophistication. Dark gray corresponds to initialization, gray to descent subroutines, and light gray to perturbation subroutines.

3.1.2 Algorithms

Initialization + local search. This algorithm simply runs the initialization + bin packing subroutine to build an initial solution, and then applies the routing local search subroutine to improve it. It has the advantage of being fast (about four minutes on our large-scale instances on average) since it is based on an LP. It is detailed in Algorithm 1.

Algorithm 1: Initialization + local search

input : $\mathcal{I}$ an IRP instance.
output: A solution $\mathbf{r}$ to the IRP.
$\mathbf{y}$ = `flow relaxation` ($\mathcal{I}$);
$\mathbf{r}$ = `bin packing` ($\mathcal{I}$, $\mathbf{y}$);
$\mathbf{r}$ = `routing local search` ($\mathcal{I}$, $\mathbf{r}$);

Route-based matheuristic. As detailed in Section 1, Archetti, Boland, and Speranza (2017) and Bertazzi et al. (2019) solve the IRP given a subset of "promising routes" that are defined heuristically. This corresponds to reducing the set of feasible solutions to the IRP, which allows solving an MILP. The routes are either selected among those created during a tabu search and leading to cost improvements, or in a constructive manner. The underlying assumption is that they are likely to appear in a good IRP solution. We adapt this idea to our setting. The main difficulty is that our MILP (multi-attribute-IRP) is intractable, even when we restrict the set of "promising routes" to the set of an initial solution. This is due to the multicommodity aspect, the scale of our instances and the routes that last several days. We solve the restricted MILP heuristically, using a large neighborhood approach: 1) Apply the initialization + local search algorithm to get an initial solution. 2) Take the current solution as set of promising routes, and solve sequentially one MILP per depot, with the corresponding promising routes that start from it. It is detailed in Algorithm 2.

Algorithm 2: Route-based matheuristic

input : $\mathcal{I}$ an IRP instance, $\mathbf{r} = (r_k)_{1 \leq k \leq K}$ the current solution with $K \in \mathbb{Z}^+$ routes, `time limit` a time limit.
output: The solution $\mathbf{r}$ updated.
$\mathbf{r}$ = `initialization + local search`($\mathcal{I}$);
for $d \in D$ **do**
  $\mathbf{r}$ = `reload fixed path vehicles`($\mathcal{I}, \mathbf{r}, d$); `// small gap`
  if *time elapsed* $\geq$ *time limit* **then**
    break;

Large neighborhood search. The large neighborhood search Algorithm 3 first uses the initialization + local search approach to find a good initial solution. It then explores four kinds of neighborhoods. Two of them always improve the solution: the routing local search and reload fixed-path vehicles subroutines. Contrary to the route-based matheuristic, the latter is applied with a greater gap limit and an additional time limit, in order to avoid spending too much time within it and cycle over the neighborhoods instead. The two remaining ones are perturbations, which means they can deteriorate the solution. They fix a part of the current solution and optimize the quantities and routes involving a particular customer or commodity over the entire horizon. They both lead to substantial changes, allowing the search to escape from local minima. The LNS uses both iteratively, selecting 200 customers and 15 commodities at random per outer step. The outer LNS iterations are illustrated by the loop arc on Figure 1. The LNS returns the best solution found, comparing after each subroutine the current solution with the best one so far. The main idea behind this LNS is to consider the structure of the IRP, "decompose" it along its major axes, and solve smaller natural problems to explore the solution space. One difference with Coelho, De Maio, and Laganà (2020) or Archetti, Guastaroba, et al. (2021), is that our idea is not to fix a part of the variables of the MILP (multi-attribute-IRP) and optimize with respect to the remaining ones, but to define new smaller MILPs based on the structure of the IRP. Let us now introduce a concept that helps describing our subroutines.

Algorithm 3: Large neighborhood search

input : $\mathcal{I}$ an IRP instance, $\mathbf{r} = (r_k)_{1 \leq k \leq K}$ the current solution with $K \in \mathbb{Z}^+$ routes, `time limit` a time limit, $n_{\texttt{cust}}$ the number of customers to reinsert, $n_{\texttt{comm}}$ the number of commodities to reinsert.

output: The solution $\mathbf{r}$ updated.

$\mathbf{r} =$ `initialization + local search`$(\mathcal{I})$;

while *time elapsed* $<$ *time limit* **do**

  $\mathbf{r} =$ `routing local search`$(\mathcal{I}, \mathbf{r})$;

  for $d \in D$ **do**

    $\mathbf{r} =$ `reload fixed path vehicles`$(\mathcal{I}, \mathbf{r}, d)$; `// large gap`

  $C_{\texttt{sub}} = sample(C, n_{\texttt{cust}})$;

  for $c \in C_{sub}$ **do**

    $\mathbf{r} =$ `customer reinsertion`$(\mathcal{I}, \mathbf{r}, c)$; `// large gap`

  $M_{\texttt{sub}} = sample(M, n_{\texttt{comm}})$;

  for $m \in M_{sub}$ **do**

    $\mathbf{r} =$ `commodity reinsertion`$(\mathcal{I}, \mathbf{r}, m)$; `// large gap`

3.2 Flow Graphs and Formulations

Let us consider the generic MILP formulation:

$$\min_{\mathbf{y}, \mathbf{x}} \quad \sum_{m \in M} \mathbf{y}_m^\top \mathbf{c}_m + \sum_{r \in \mathbf{r}} x_r c_r \qquad \text{(generic-flow-MILP)}$$

$$\text{subject to} \quad \sum_{a \in \delta^+(v)} y_{ma} = \sum_{a \in \delta^-(v)} y_{ma}, \quad \forall m \in M, \quad \forall v \in \mathcal{V}^m \tag{7a}$$

$$\mathbf{y}_m^{\min} \leq \mathbf{y}_m \leq \mathbf{y}_m^{\max}, \quad \forall m \in M \tag{7b}$$

$$\sum_{m \in M} y_{ma} \ell_m \leq x_r L, \quad \forall a = (d \to (c, r)), \quad \forall r \in \mathbf{r} \tag{7c}$$

$$\mathbf{y} \in \mathbb{Z} \tag{7d}$$

$$\mathbf{x} \in \{0, 1\} \tag{7e}$$

The variable $\mathbf{y}_m$ encodes a flow on a given *commodity graph* thanks to Equations (7a)-(7b). This flow enables modelling the depot and customer inventory dynamics of commodity m defined by constraints (6e)-(6h), and the quantities sent and received. Variable $\mathbf{x}$ encodes the routes that are used to deliver the commodities. Constraint (7c) indeed enforces the flows to respect the vehicle capacity when a route is used. In order to obtain a specific MILP formulation from this generic one, we must specify which commodity graph is used, and which set of routes $\mathbf{r}$ is considered. We are going to use several distinct commodity graphs. However, they all share a common structure which we describe now:

Depot subgraphs. A subgraph per depot d (Figure 2 (b)), which models its inventory dynamics. It is shared by the distinct formulations we introduce in this paper and has the following vertices: $(t, d, \texttt{morning})$ for $t \in [T+1]$, and $(t, d, \texttt{evening})$ for $t \in [T]$.

Customer subgraphs. A subgraph per customer c (Figure 2 (c)), which models its inventory dynamics. It is also shared by the distinct formulations we introduce in this paper, and it has the following vertices: $(t, c, \texttt{morning})$ for $t \in [T+1]$, and $(t, c, \texttt{evening})$ for $t \in [T]$.

Table 1: Arcs of the commodity flow graph shared by our formulations.
When not stated in the table, **Min** is 0, **Cost** is 0 and **Max** is ∞.

Subgraph	Arc type	Arc description	Origin	Destination	Min	Max	Cost
Depot	Incoming	Initial inventory depot	`initial inventory`	$(1, d, \texttt{morning})$	I^0_{md}	I^0_{md}	
Depot	Outgoing	Final inventory depot	$(T+1, d, \texttt{morning})$	`final inventory`			
Depot	Incoming	Release depot	`release`	$(t, d, \texttt{morning})$	b^+_{mdt}	b^+_{mdt}	
Customer	Incoming	Initial inventory customer	`initial inventory`	$(1, c, \texttt{morning})$	I^0_{mc}	I^0_{mc}	
Customer	Outgoing	Final inventory customer	$(T+1, c, \texttt{morning})$	`final inventory`			
Customer	Outgoing	Demand customer	$(t, c, \texttt{morning})$	`demand`	b^-_{mct}	b^-_{mct}	
Customer	Incoming	Shortage customer	`shortage`	$(t, c, \texttt{morning})$			$c^{\texttt{short}}_{mc}$
Routes		Transport $d \to c$	Formulation specific	(see Sections 4.1, 6.1, 6.2, 6.3)			
Depot	Internal	Daily inventory depot	$(t, d, \texttt{morning})$	$(t, d, \texttt{evening})$			
Depot	Internal	Free night inventory depot	$(t, d, \texttt{evening})$	$(t+1, d, \texttt{morning})$		κ_{mdt}	
Depot	Internal	Excess night inventory depot	$(t, d, \texttt{evening})$	$(t+1, d, \texttt{morning})$			$c^{\texttt{exc}}_{md}$
Customer	Internal	Daily inventory customer	$(t, c, \texttt{morning})$	$(t, c, \texttt{evening})$			
Customer	Internal	Free night inventory customer	$(t, c, \texttt{evening})$	$(t+1, c, \texttt{morning})$		κ_{mct}	
Customer	Internal	Excess night inventory customer	$(t, c, \texttt{evening})$	$(t+1, c, \texttt{morning})$			$c^{\texttt{exc}}_{mc}$
Artificial		Circulation	`source`	`release`			
Artificial		Circulation	`source`	`initial inventory`			
Artificial		Circulation	`source`	`shortage`			
Artificial		Circulation	`demand`	`sink`			
Artificial		Circulation	`final inventory`	`sink`			
Artificial		Circulation	`sink`	`source`			

A route subgraph. Its specific structure depends on the formulation we consider. It contains paths between vertices of the form $(t, d, \texttt{morning})$ and vertices of the form $(\tilde{t}, c, \texttt{evening})$ as shown on Figure 2 (a). Flow variables $\mathbf{y}$ on those paths model the quantities sent from depots to customers.

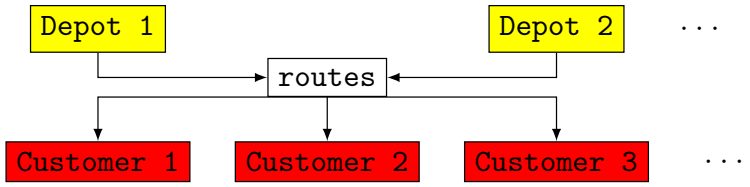
Artificial vertices. In order to model commodity flows as circulations over commodity graphs, we add artificial vertices connected to the subgraphs above: `source`, `sink`, `initial inventory`, `final inventory`, `release`, `shortage`, and `demand`.

The details of the arcs of the shared subgraphs defined above are in Table 1. For each arc, we give the following information. The subgraph it belongs to is first given. Then, we distinguish "incoming", "outgoing" and "internal" arcs with respect to the depots and customer subgraphs. A short description is stated to understand the meaning of the arcs. We specify the origin and destination vertices, as well as the minimum and maximum flow capacities associated to the flow variables on the arcs. Last, the cost corresponding to these variables are also given.

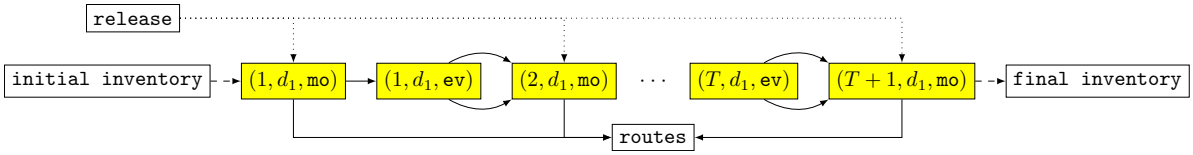
Given a graph $\tilde{\mathcal{D}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{A}})$ with capacities associated to its arcs $(\mathbf{y}^{\min}, \mathbf{y}^{\max})$, we define the *set of circulations* as $\mathcal{C}(\tilde{\mathcal{D}}, \mathbf{y}^{\min}, \mathbf{y}^{\max}) = \{\mathbf{y} \in \mathbb{R}^{\tilde{\mathcal{A}}}, \mathbf{y}_{\min} \leq \mathbf{y} \leq \mathbf{y}_{\max}, \forall v \in \tilde{\mathcal{V}}, \sum_{a \in \delta^+(v)} \mathbf{y}_a = \sum_{a \in \delta^-(v)} \mathbf{y}_a\}$. We use this notation instead of constraints (7a)-(7b) in the rest of the paper.

Remark 3.1 *In (generic-flow-MILP), routes are modelled with individual paths in the commodity graphs, bound with indicator variables* $\mathbf{x}$*. Sometimes, we define them as paths in another flow graph. In this case, some vertices and arcs are shared between routes. We detail this aspect in Sections 6.2 and 6.3.*

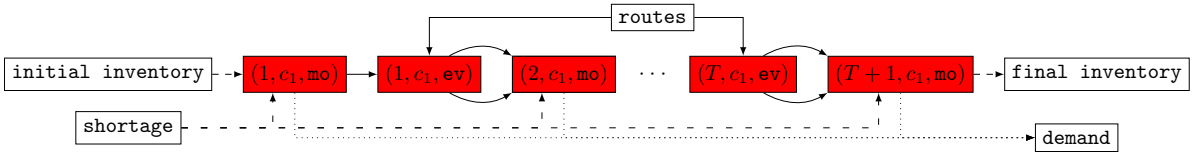
Remark 3.2 *We sparsify the commodity graph. Instead of considering the sets of depots* D *and customers* C *in the commodity graph* $\mathcal{D}^m$*, we define the subsets* $D_m = \{d \in D, m \in M_d\}$ *and* $C_m = \{c \in C, m \in M_c\}$ *the depots and customers that use commodity* m*. We then restrict the depots and customer subgraphs of* $\mathcal{D}^m$ *to the ones of* D_m *and* C_m*.*



(a) Global structure



(b) Depot subgraph



(c) Customer subgraph

Figure 2: Commodity flow graph. Overview of the global graph structure (a), and then details of the subgraphs "`Depot 1`" (b) and "`Customer 1`" (c). The abbreviations "`mo`" and "`ev`" stand for morning and evening respectively.

4 Flow Relaxation + Bin Packing Subroutine

The flow relaxation + bin packing subroutine is a fast heuristic to get an initial solution to (multi-attribute-IRP). It takes as input an IRP instance, and returns an initial IRP solution built from intermediate flow solutions that encode who sends what to whom and when.

4.1 Multiple Minimum Cost Flows and Relaxation

Minimum cost flow formulation. Let $\mathbf{y} = (y_{ma})_{m \in M, a \in \mathcal{A}^m}$ be a flow variable and $\mathbf{c}$ be the corresponding costs defined in Table 1 for the shared subgraphs and Figure 3 for the specific route subgraph. We consider the following LP:

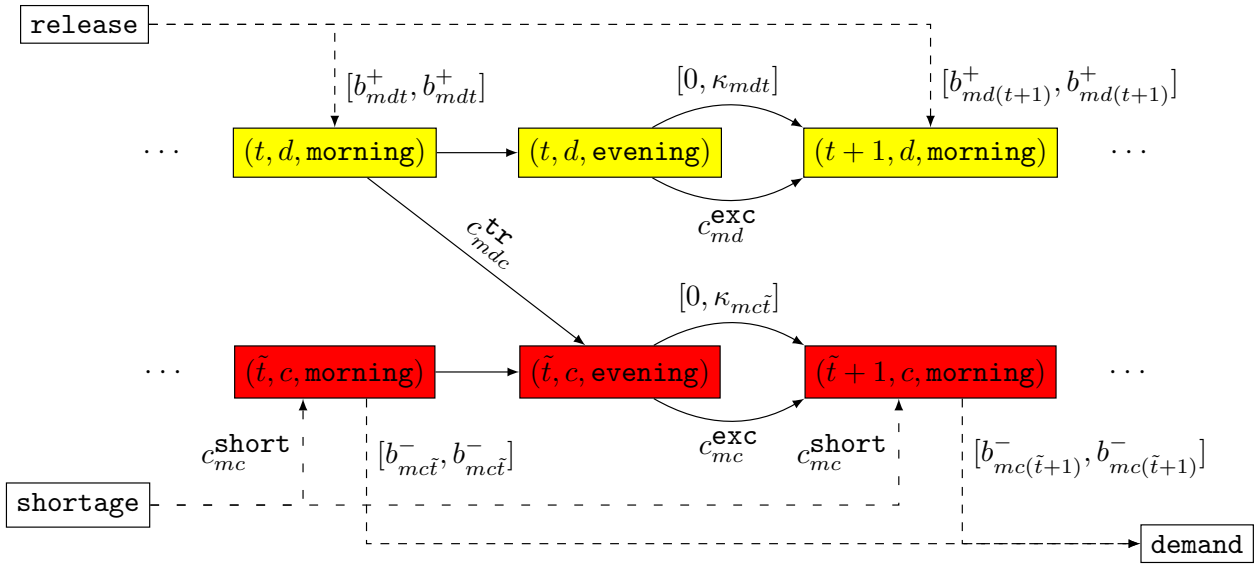
$$\min_{\mathbf{y}} \quad \sum_{m \in M} \mathbf{y}_m^\top \mathbf{c}_m \qquad \text{(flow-relaxation)}$$

$$\text{subject to} \quad \mathbf{y}_m \in \mathcal{C}(\mathcal{D}^m, \mathbf{y}_m^{\min}, \mathbf{y}_m^{\max}), \quad \forall m \in M \qquad \text{(8a)}$$

In this variant of the generic MILP (Section 3.2), we do not introduce a route variable $\mathbf{x}$ and the corresponding cost. We instead consider a soft version of constraint (7c) in the commodity cost $\mathbf{c}_m$. This leads to a separate flow LP for each commodity in (flow-relaxation). Let us now introduce the details of the commodity graph.

Details of the commodity graph. We define one graph per commodity m, named $\mathcal{D}^m = (\mathcal{V}^m, \mathcal{A}^m)$. The vertices $\mathcal{V}^m$ are exactly the ones defined in Section 3.2. The arcs detailed in Table 1 are included. We specify the route subgraph in the table of Figure 3. It is made of direct transport and delayed transport $d \to c$

arcs. These arcs are added when the date of arrival at the customer is smaller than the horizon T. For each tuple $(t, d, c) \in [T] \times D_m \times C_m$, we add one delayed arc $\big((t, d) \to (\tilde{t}, c)\big)$ per possible delayed arrival day $\tilde{t}$ induced by an indirect path from d to c (thus visiting any set of other customers before c) that respects the route constraints defined by $\mathcal{R}$. Those possible delays are pre-computed, using a breadth-first search algorithm over the locations graph, with maximum depth set to $S_{\max}$. Indeed, we can browse the locations graph starting from depots, saving the cumulative delay at any vertex and any depth smaller than $S_{\max}$. The flow on these arcs models the quantity of commodity sent from depot d on day t to customer c with arrival on day $\tilde{t}$. The precise structure of the graph is illustrated on Figure 3. In arc annotations, capacities are given between brackets (e.g. $[0, \kappa_{mdt}]$), and costs without (e.g. $c^{\texttt{exc}}_{md}$). Dotted arrows are related to the shared artificial vertices, continuous ones to depots, customers and route subgraphs. When not stated in the table, **Min** is 0, **Cost** is 0 and **Max** is ∞. On this figure, only two days, one depot and one customer are shown. Besides, some artificial vertices are omitted for simplicity. The cost $c^{\texttt{tr}}_{mdc}$ is detailed below.



(a) Graph details

Subgraph	Arc description	Origin	Destination	Min	Max	Cost
Routes	Transport $d \to c$	$(t, d, \texttt{morning})$	$(t + \lfloor \frac{\tau_{dc}}{\tau_{\max}} \rfloor, c, \texttt{evening})$			$c^{\texttt{tr}}_{mdc}$
Routes	Delayed transport $d \to c$	$(t, d, \texttt{morning})$	$(\tilde{t}, c, \texttt{evening})$			$c^{\texttt{tr}}_{mdc}$

(b) Additional commodity graph arcs compared with Table 1

Figure 3: Details of the commodity graph $\mathcal{D}^m$ for the flow relaxation problem.

Since the routing price is paid at the vehicle level, we cannot derive a minimum cost commodity flow that takes it into account exactly without adding variables for each individual vehicle. Instead, we want to approximate this cost with transportation arcs between depots and customers naturally involving commodity flow variables. A way to do so is to use a "vehicle fraction" unit per commodity, leading to:

$$c^{\texttt{tr}}_{mdc} = \frac{\ell_m}{L}(c^{\texttt{veh}} + c^{\texttt{stop}} + c^{\texttt{km}}\Delta_{dc}), \quad \forall d \in D_m, \quad \forall c \in C_m. \tag{9}$$

In Equation (9) the factor $\frac{\ell_m}{L}$ is a way to scale the price paid for the delivery of a unit of commodity m based on the percentage of a vehicle it occupies, hence the "vehicle fraction".

Proposition 4.1 *The optimization problem (flow-relaxation) based on $|M|$ flows is a relaxation of (multi-*

attribute-IRP). The optimal value of (flow-relaxation) is a lower bound to the cost of an optimal solution to our initial problem.

We now sketch the proof. Given a feasible solution of (multi-attribute-IRP), we can deduce a feasible solution of (flow-relaxation) by fixing the quantities sent by each depot to each customer per day, delay and commodity. The inventory costs are modelled exactly with (flow-relaxation) thanks to the delayed arcs, thus equal to the ones of (multi-attribute-IRP). The transportation costs are lower bounded with Equation (9). The structure of the graph detailed on Figure 3 only allows geographically direct routes between depots and customers, whereas the route constraints allow up to $S_{\max}$ stops. Nonetheless, considering additional arcs $c_1 \to c_2$ with fraction costs $c^{\texttt{tr}}_{mc_1c_2} = \frac{\ell_m}{L}(c^{\texttt{stop}} + c^{\texttt{km}}\Delta_{c_1c_2})$ and delay $\tau_{c_1c_2}$ in an extended flow graph would also produce solutions with only direct routes. Indeed, by triangular inequality, it would always be cheaper to send commodities through geographically direct $(d \to c)$ (possibly delayed) arcs in this framework of "vehicle fraction" costs, rather than sending quantities to intermediate customer c_1 before reaching the destination c_2.

4.2 Bin Packing

We highlight here how the minimum cost flows can be used to derive an IRP solution, a step further from the lower bound computation. The $|M|$ minimum cost flow solutions resulting from (flow-relaxation) enable us to set the quantities sent by each depot to each customer per day and commodity, but do not directly lead to a set of routes. Indeed, for now, we do not know how quantities are loaded in various vehicles. We highlight the fact that the delayed arcs are not used to build an initial solution. They are only introduced to compute a lower bound. To deduce a set of direct routes, we approximately solve one bin packing problem per tuple $(d, c, t) \in D \times C \times [T]$, using the first-fit-decreasing heuristic. The instance of the bin packing is given by the set of commodities to be sent on day t from depot d to customer c, their respective lengths, and the length of one vehicle. The solution to the bin packing problem leads to a low number of vehicles each of length L, with corresponding loading made of possibly $|M|$ distinct commodities. At this point, we get a set of direct routes as a first feasible solution to (multi-attribute-IRP).

5 Routing Local Search Subroutine

Our solution processes emphasized on Figure 1 rely on the routing local search subroutine Algorithm 4. We detail here both the local search procedure, and the neighborhoods listed in Table 2.

5.1 The Neighborhoods

Before introducing the local search procedure, we focus on the TSP and SDVRP neighborhoods in an IRP framework. The routing local search subroutine combines those neighborhoods.

Routes impact inventories. We highlight the fact that the 12 neighborhoods detailed below in Table 2 alter the routes of a solution and the inventories at the depots or at the customers (contrary to the SDVRP framework). Therefore, whenever a neighborhood is considered, we evaluate the effects on inventories and routes so as to check feasibility and to estimate the cost change. For instance, the optimal order of a route not only depends on the distances Δ, but also on the delays introduced in the inventory dynamics of the customers involved. Therefore, even elementary neighborhoods require calculation. They can be seen as a generalization of the TSP and SDVRP concepts to the continuous-time IRP.

5.2 Routing Local Search

From the list of neighborhoods emphasized in Table 2, we design the routing local search, Algorithm 4. During one iteration, it starts from the largest neighborhoods involving multiple depots, then considers the

Table 2: Routing local neighborhoods: single-depot and multi-depot variants are considered.

Type	Name	Description
TSP	`relocate`	change the position of one stop in a route.
	`swap`	exchange the positions of two stops in a route.
	`2-opt*`	cut a route into three parts and revert the order of the middle one.
SDVRP single-depot	`insert`	give a stop s from route r_1 to route r_2.
	`swap single depot`	`swap` adapted to two routes with same depot.
	`merge`	merge two routes r_1 and r_2 on the same day.
	`merge multi day`	`merge` extended to routes with different start dates.
	`delete route`	delete a route.
	`change day`	move a route in time, one day before or after.
SDVRP multi-depot	`insert multi depot`	`insert` extended to routes with distinct depots.
	`swap multi depot`	`swap` extended to routes with distinct depots.
	`2-opt* multi depot`	cut two routes each into two parts and exchange their end parts.

single-depot neighborhoods, and finishes by the single route TSP neighborhoods. Since the neighborhoods tend to reduce the number of routes, the routing local search algorithm deletes routes at several stages of its iteration. It only explores neighborhoods in feasible directions that improve the cost.

Some features of this local search differ from the common SDVRP local search algorithms. Because of the computations involved for `insert`, `swap single depot`, and for the multi-depot neighborhoods, and the number of pairs of routes, the routing local search subroutine only samples a fraction of them. To do so, it explicitly samples a subset of the pairs of routes of the solution according to a uniform distribution. We tune this approach with parameter p in Algorithm 4 to find a good cost gain per CPU time ratio. We do so instead of restricting the routes candidates with geographic criteria, because the inventory costs cannot be neglected. Two routes that visit customers that are far from each other may still be suitable candidates for a `swap` for instance, due to the change in inventory cost. The `change day` function is applied per route one day forward or backward, until no improvement is found.

Algorithm 4: Routing local search

input : $\mathcal{I}$ an IRP instance, $\mathbf{r} = (r_k)_{1 \leq k \leq K}$ the current solution with $K \in \mathbb{Z}^+$ routes, $n_{it} \in \mathbb{Z}^+$ a number of iterations, p a percentage.
output: The solution $\mathbf{r}$ updated.
for $i = 1 : n_{it}$ **do**
  perform one pass of `insert multi depot` over p% of the pairs of routes at random;
  perform one pass of `swap multi depot` over p% of the pairs of routes at random;
  perform `delete route` per day and depot until no improvement;
  perform one pass of `2-opt* multi depot` over p% of the pairs of routes at random;
  perform `delete route` per day and depot until no improvement;
  perform best `merge` per day and depot until no improvement;
  perform best `merge multi day` per day and depot until no improvement;
  perform `delete route` per day and depot until no improvement;
  perform `insert` over p% of the pairs of routes per day and depot;
  perform `swap single depot` over p% of the pairs of routes per day and depot;
  perform one pass of `relocate`, `swap` and `2-opt*` over the routes until no improvement;
  perform `change day` until no improvement;

6 MILP-Based Neighborhoods and Perturbations

We now introduce three subroutines that are based on optimization problems written as MILPs. They all leverage the commodity graph structure emphasized in Section 3.2.

6.1 Reload Fixed-Path Vehicles Subroutine

Reload neighborhood problem. Let us define the problem behind this large neighborhood. We consider a subset of routes $\mathbf{r}_{\texttt{reload}}$ of the current IRP solution $\mathbf{r}$, in our case the routes that start from a given depot d. We solve the following problem: choose the routes to keep in the solution among $\mathbf{r}_{\texttt{reload}}$, and re-estimate the delivered quantities (for the whole set of commodities) of the routes kept, to minimize the total cost. In our large neighborhood setting, we fix the remaining routes of the current solution $\mathbf{r}$.

We denote by x_r for $r \in \mathbf{r}_{\texttt{reload}}$ the indicator variable for keeping route r, and by $\mathbf{y} = (\mathbf{y}_m)_{m\in M}$ the set of commodity flow variables. The commodity flow graphs $\big(\mathcal{D}^m(\mathbf{r}, \mathbf{r}_{\texttt{reload}})\big)_{m\in M}$ involved are defined below. We model the problem with the following formulation:

$$\min_{\mathbf{y},\mathbf{x}} \quad \sum_{m\in M} \mathbf{y}_m^\top \mathbf{c}_m + \sum_{r\in\mathbf{r}_{\texttt{reload}}} x_r \left(c^{\texttt{veh}} + c^{\texttt{stop}}(|P^r| - 1) + c^{\texttt{km}} \sum_{a\in A(P^r)} \Delta_a \right) \tag{Reload-MILP}$$

$$\text{subject to} \quad \mathbf{y}_m \in \mathcal{C}\big(\mathcal{D}^m(\mathbf{r}, \mathbf{r}_{\texttt{reload}}), \mathbf{y}_m^{\min}(\mathbf{r}, \mathbf{r}_{\texttt{reload}}), \mathbf{y}_m^{\max}(\mathbf{r}, \mathbf{r}_{\texttt{reload}})\big), \quad \forall m \in M \tag{10a}$$

$$\sum_{m\in M} y_{ma}\ell_m \le x_r L, \quad \forall a = (d \to (c, r)), \quad \forall r \in \mathbf{r}_{\texttt{reload}} \tag{10b}$$

$$\mathbf{y} \in \mathbb{Z} \tag{10c}$$

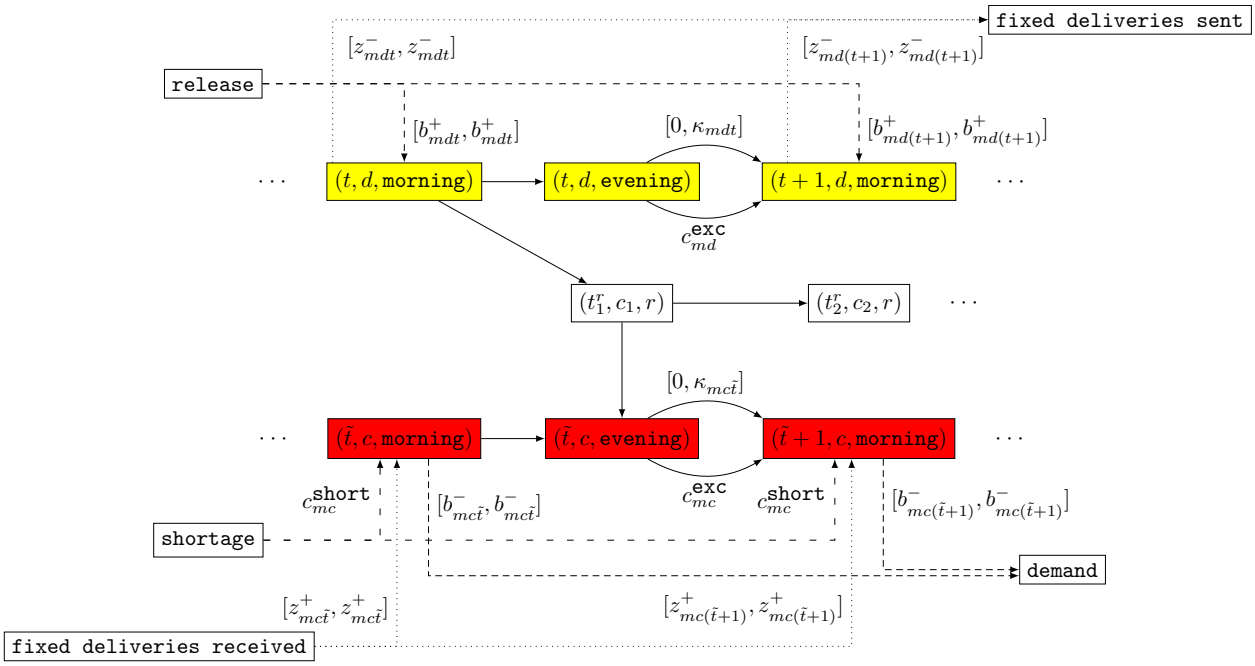
$$\mathbf{x} \in \{0, 1\} \tag{10d}$$

This formulation is very close to the generic MILP introduced in Section 3.2. The objective function is composed of one flow cost per commodity $m \in M$ (inventory and shortage costs), and of the routing cost of each route kept among $\mathbf{r}_{\texttt{reload}}$. Constraint (10b) ensures that the commodity flows from depots to customers only exist along routes that are kept, and that the capacity of the vehicles is respected. The last two constraints define integer and binary variables. We highlight that this MILP exactly formulates the reloading of a given subset of routes. Solving the problem (Reload-MILP) leads to a new feasible solution with lower cost.

The commodity graph $\mathcal{D}^m(\mathbf{r}, \mathbf{r}_{\texttt{reload}})$ for $m \in M$ depends both on the current solution $\mathbf{r}$, and on the routes to potentially keep and reload $\mathbf{r}_{\texttt{reload}}$. As previously, the backbone structure is the same as in Section 3.2: one subgraph per customer, one per depot, one for the routes, and additional vertices to create circulations. The special route subgraph is detailed on Figure 4. In arc annotations, capacities are given between brackets (e.g. $[0, \kappa_{mdt}]$), and costs without (e.g. $c^{\texttt{exc}}_{md}$). Dotted arrows are related to the shared artificial vertices, continuous ones to depots, customers and route subgraphs. When not stated in the table, **Min** is 0, **Cost** is 0 and **Max** is ∞. We explicitly create individual route paths, with vertices of the form (t^r_s, c_s, r). It models the fact that when using route r, commodities are delivered to customer c_s on day t^r_s at position s of the route. The details of the arcs can be found in the table of Figure 4. Besides, since we fix the quantities sent by the routes in $\mathbf{r}\backslash\mathbf{r}_{\texttt{reload}}$, we need two additional vertices in the route subgraph that we name `fixed deliveries sent`, and `fixed deliveries received`. The former is connected to depot vertices in order to take other quantities sent into account. The second is connected to customer vertices to model other quantities received.

6.2 Customer Reinsertion Subroutine

As mentioned in the overview Section 3, our neighborhoods are based on a decomposition of the IRP along its main axes. Previous sections focus on the routes. Here, we design a perturbation based on the customers. We call it perturbation because it may slightly increase the cost of the IRP solution.



(a) Graph details

Subgraph	Arc description	Origin	Destination	Min	Max	Cost
Routes	Fixed deliveries sent	$(t, d, \texttt{morning})$	`fixed deliveries sent`	z^-_{mdt}	z^-_{mdt}	
Routes	Fixed deliveries received	`fixed deliveries received`	$(t, c, \texttt{evening})$	z^+_{mct}	z^+_{mct}	
Routes	Transport $d \to (c_1, r)$	$(t^r, d, \texttt{morning})$	(t^r_1, c_1, r)			
Routes	Transport $(c_s, r) \to (c_{s+1}, r)$	(t^r_s, c_s, r)	(t^r_{s+1}, c_{s+1}, r)			
Routes	Transport $(c_s, r) \to c$	(t^r_s, c, r)	$(t^r_s, c, \texttt{evening})$			
Artificial	Circulation	`source`	`fixed deliveries received`			
Artificial	Circulation	`fixed deliveries sent`	`sink`			

(b) Additional commodity graph arcs compared with Table 1

Figure 4: Details of the commodity graph $\mathcal{D}^m(\mathbf{r}, \mathbf{r}_{\texttt{reload}})$ for the reload fixed-path vehicle neighborhood.

Customer reinsertion problem. Let us define the customer reinsertion problem. Once the customer $c \in C$ is removed from the solution – that is to say, removed from the routes that deliver to it, leading to zero delivery $\mathbf{z}^+$ in the inventory dynamics (6g) – we need to reinsert it in the solution, using only former routes and new direct routes. This means choosing: 1) The insertion position of customer c in each route of the solution in which it is inserted, keeping the relative order of the other stops unchanged. 2) The quantity of each commodity to be delivered by those former routes where c is inserted. 3) New direct routes (path, timing and quantities) to deliver c. It can be formulated as a MILP akin to the generic one defined in Section 3.2.

The customer insertion MILP. Let $\mathbf{y} = (\mathbf{y}_m)_{m \in M_c}$ be the commodity flow variable. Instead of the indicator route variables in Section 3.2, we use another type of graph to model the vehicles with a flow variable $\mathbf{x}$. Indeed, a flow is a convenient tool to model the fact that we have multiple insertion positions in a given route for customer c, and we can choose at most one of them. We link the flow variables on the

route subgraphs of the commodity graphs with this vehicle flow. It leads to the MILP:

$$\min_{\mathbf{y},\mathbf{x}} \quad \sum_{m\in M_c} \mathbf{y}_m^\top \mathbf{c}_m + c_x^\top \mathbf{x} \tag{Cust-MILP}$$

$$\text{subject to} \quad \mathbf{y}_m \in \mathcal{C}(\mathcal{D}_c^m, \mathbf{y}_m^{\min}, \mathbf{y}_m^{\max}), \quad \forall m \in M_c \tag{11a}$$

$$\mathbf{x} \in \mathcal{C}(\mathcal{D}_c^{\texttt{veh}}, \mathbf{x}^{\min}, \mathbf{x}^{\max}) \tag{11b}$$

$$\sum_{m\in M_c} y_{ma}\ell_m \leq x_a L, \quad \forall a = (d \to c), \quad \forall d \in D \tag{11c}$$

$$\sum_{m\in M_c} y_{ma}\ell_m \leq x_a L^r_{\texttt{free}}, \quad \forall a = (r \to (r,s)), \quad \forall r \in \mathbf{r}, \quad \forall s \in [|P^r|] \tag{11d}$$

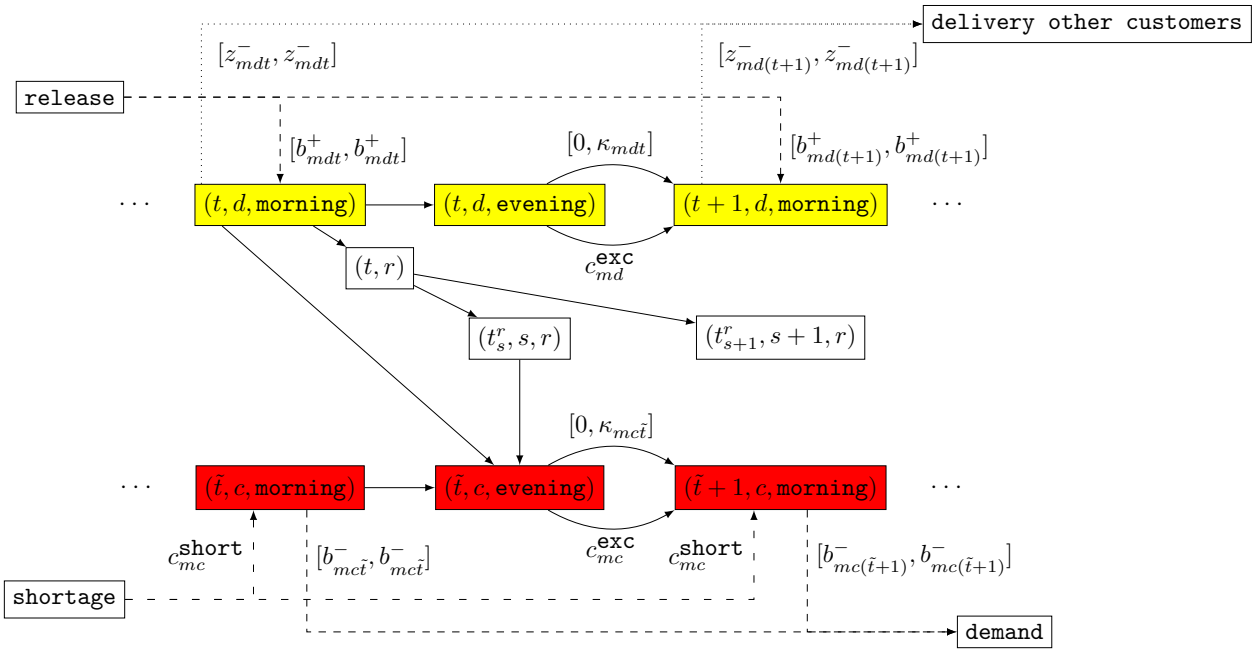
$$\mathbf{x} \in \mathbb{Z},\ \mathbf{y} \in \mathbb{Z} \tag{11e}$$

The vehicle flow as well as each commodity flow must respect circulation constraints. In Equation (11c), we force the amount of commodities to be sent through new direct routes from the depots to the customer c not to exceed in length the total content size of the vehicles involved. Indeed, we consider the total vehicle capacity with this constraint, and not individual vehicles each of capacity L. Similarly, Equation (11d) does so for the former routes $r \in \mathbf{r}$ at position s having remaining loading space $L^r_{\texttt{free}}$.

Details of the commodity flow graphs. Based on the generic graph structure Section 3.2, we define new commodity graphs $\mathcal{D}_c^m = (\mathcal{V}_c^m, \mathcal{A}_c^m)$ for each commodity $m \in M_c$ for the reinsertion of customer $c \in C$. We highlight the fact that here only one customer is involved in the problem, so only one customer subgraph is present. The depots and customer subgraphs are introduced in Section 3.2. We now define the route subgraph specific to this customer reinsertion MILP. First, we have a vertex `delivery other customers` in the route subgraph. It is connected to the depot subgraphs to model the quantities sent to other customers. Figure 5 shows the specific commodity graph. In arc annotations, capacities are given between brackets (e.g. $[0, \kappa_{mdt}]$), and costs without (e.g. $c^{\texttt{exc}}_{md}$). Dotted arrows are related to the shared artificial vertices, continuous ones to depots, customers and route subgraphs. When not stated in the table, **Min** is 0, **Cost** is 0 and **Max** is ∞. In this commodity graph, we consider two types of routes. 1) new direct routes are modelled by direct arcs of the type $(t, d, \texttt{morning}) \to (t', c, \texttt{evening})$ and do not involve additional vertices. 2) former routes in which we can insert customer c. They are modelled with one vertex (t, r) for route r starting on day t, connected to the starting depot morning vertex and to each vertex of the form (t^r_s, s, r). The vertex (t^r_s, s, r) is related to the possible insertion position s in route r leading to an arrival day t^r_s at customer c. It is connected to the customer subgraph. The details of the route arcs are in the table of Figure 5. We cannot only add one vertex per former route, since the optimal insertion position depends on the commodity flows, and not only on routing costs. Therefore, the optimal insertion position cannot be pre-computed. Besides, we do not introduce an approximate cost for the transportation arcs. Instead, we define another graph $\mathcal{D}_c^{\texttt{veh}} = (\mathcal{V}_c^{\texttt{veh}}, \mathcal{A}_c^{\texttt{veh}})$ as follows.

Details of the vehicle flow graph. We consider the $\mathcal{V}_c^{\texttt{veh}}$ vertices:

- Artificial vertices labelled: `source` and `sink`.
- Depots vertices labelled: $(t, d, \texttt{morning})$ for $t \in [T]$, $d \in D$.
- Customer vertices labelled: $(t, c, \texttt{evening})$ for $t \in [T]$.
- Routes vertices labelled: (t, r) for each former route $r \in \mathbf{r}$ that can include an additional stop and that does not reach the vehicle capacity L already.
- Nodes per insertion position in former routes: (t^r_s, s, r) for each position s at which we can insert customer c in r without exceeding the time horizon T. The date t^r_s on which the route delivers customer c can be pre-computed.



(a) Graph details

Subgraph	Arc description	Origin	Destination	Min	Max	Cost
Routes	Delivery other customers	$(t, d, \texttt{morning})$	`delivery other customers`	z^-_{mdt}	z^-_{mdt}	
Routes	Transport $d \to c$ new route	$(t, d, \texttt{morning})$	$(t + \lfloor \frac{\tau_{dc}}{\tau_{\max}} \rfloor, c, \texttt{evening})$			
Routes	Transport $d \to r$ former route	$(t, d, \texttt{morning})$	(t, r)			
Routes	Transport $r \to (r, s)$ former route	(t, r)	(t^r_s, s, r)			
Routes	Transport $(r, s) \to c$ former route	(t^r_s, s, r)	$(t^r_s, c, \texttt{evening})$			
Artificial	Circulation	`delivery other customers`	`sink`			

(b) Additional commodity graph arcs compared with Table 1

Figure 5: Details of the commodity graph $\mathcal{D}^m_c$ for the customer reinsertion neighborhood.

The arcs of the customer reinsertion vehicle flow graph are defined in Table 3. We see that most of its structure is shared with the commodity flow graph described above. Besides, the costs c^r_{sc} are the ones induced by the insertion of customer c at position s in the stops of route r. They involve routing and inventory considerations at the other customers delivered by route r, because of the delays. They can be exactly computed considering the former list of r's stops, and enumerating the insertion possibilities. We emphasize the 1-maximum capacity on the arcs of the form $(t, d, \texttt{morning}) \to (t, r)$ combined with the circulation constraint enforce that at most one insertion position is chosen in former routes.

Proposition 6.1 *The problem (Cust-MILP) is a relaxation of the optimal customer reinsertion problem in the IRP solution, where the new direct routes are aggregated per routing arc $(d \to c)$.*

We sketch the proof. The constraints for new direct routes do not exactly model the routing structure: instead of a set of vehicles having each a content size L, it is as if we had one large vehicle with the total content size on each arc $d \to c$. More precisely, given $d \in D$, filling $p \in \mathbb{Z}^+$ vehicles of content size L and cost $c^{\texttt{veh}} + c^{\texttt{stop}} + c^{\texttt{km}}\Delta_{dc}$ each enables less loading freedom than filling one large vehicle of content size pL with same total cost $p(c^{\texttt{veh}} + c^{\texttt{stop}} + c^{\texttt{km}}\Delta_{dc})$. In the objective function, the costs $\mathbf{c}_m$ for $m \in M_c$ and c_x stem from the arc features stated in Table 1, the table of Figure 5 and in Table 3. Because of their values and because we relax the routing structure, this problem is a relaxation of the customer reinsertion in the current

Table 3: Arcs of the customer reinsertion vehicle flow graph $\mathcal{D}_c^{\texttt{veh}}$.
When not stated, **Min** is 0, **Cost** is 0 and **Max** is ∞.

Arc description	Origin	Destination	Min	Max	Cost
Start routes	$\texttt{source}$	$(t, d, \texttt{morning})$			
End routes	$(t, c, \texttt{evening})$	$\texttt{sink}$			
Transport $(d \to c)$ new route	$(t, d, \texttt{morning})$	$(t + \lfloor \frac{\tau_{dc}}{\tau_{\max}} \rfloor, c, \texttt{evening})$			$c^{\texttt{veh}} + c^{\texttt{stop}} + c^{\texttt{km}} \Delta_{dc}$
Transport $(d \to r)$ former route	$(t, d, \texttt{morning})$	(t, r)		1	
Transport $(r \to (r, s))$ former route	(t, r)	(t_s^r, s, r)			c_{sc}^r
Transport $((r, s) \to c)$ former route	(t_s^r, s, r)	$(t_s^r, c, \texttt{evening})$			
Circulation	$\texttt{source}$	$\texttt{sink}$			
Circulation	$\texttt{sink}$	$\texttt{source}$			

solution of the IRP.

Rebuilding routes. All the decisions we take are encapsulated in the commodity flow variable $\mathbf{y}$. From $\mathbf{y}$, we can easily update the solution $\mathbf{r}$, filling former routes with the indicated quantities. Then, a bin packing problem is solved approximately per depot and per day to create the new direct routes from the flows on direct routes arcs. Aggregating per $(d \to c)$ arc enables us to derive a MILP of reasonable size, but this relaxation can lead to a potential cost increase. Our LNS accepts the new IRP solution if its true cost is not more than 1% greater than the previous solution cost.

6.3 Commodity Reinsertion Subroutine

The last subroutine we detail here is the commodity reinsertion. As for the customer reinsertion, the idea is to perturb the solution at a broad scale, possibly increasing the cost.

Commodity reinsertion problem. We are interested in the following problem. Once a commodity $m \in M$ is removed from every delivery of the current solution, we want to choose the quantity of m to send through each former route, and through new direct routes. Former routes are the routes of the current solution with the commodity m removed that have a remaining load. The restriction to new direct routes is a choice to quickly compute a solution.

The commodity insertion MILP. As for the customer reinsertion, we couple a commodity flow variable $\mathbf{y}$ with a vehicle flow $\mathbf{x}$. The resulting MILP is given by:

$$\min_{\mathbf{y},\mathbf{x}} \quad \mathbf{y}_m^\top \mathbf{c}_m + c_x^\top \mathbf{x} \tag{Comm-MILP}$$

$$\text{subject to} \quad \mathbf{y}_m \in \mathcal{C}(\mathcal{D}_m^m, \mathbf{y}_m^{\min}, \mathbf{y}_m^{\max}) \tag{12a}$$

$$\mathbf{x} \in \mathcal{C}(\mathcal{D}_m^{\texttt{veh}}, \mathbf{x}^{\min}, \mathbf{x}^{\max}) \tag{12b}$$

$$y_{ma} \leq \left\lfloor \frac{L}{\ell_m} \right\rfloor x_a, \quad \forall a = (d \to c) \quad \forall d \in D_m \quad \forall c \in C_m \tag{12c}$$

$$\mathbf{x} \in \mathbb{Z}, \ \mathbf{y} \in \mathbb{Z} \tag{12d}$$

We highlight only one commodity flow and one vehicle flow are involved here. Based on the subgraphs introduced in Section 3.2, the inventory constraints and costs are exactly modelled. The routing structure is approximated, with a cost paid per unit of a single "large vehicle with total content size" per arc $(d \to c)$ for $c \in C_m$ and $d \in D_m$, modelled by constraint (12c).

Details of the commodity flow graph. Let $m \in M$ be the commodity we consider. As previously, we build a commodity graph $\mathcal{D}_m^m = (\mathcal{V}_m^m, \mathcal{A}_m^m)$. It shares the backbone structure with customer and depot subgraphs as well as artificial vertices defined in Section 3.2. We now detail the specific route subgraph for the commodity reinsertion problem. It aims at modelling both former routes and new direct routes. For a former route $r \in \mathbf{r}$, for c visited by r, we denote by $s \in [|P^r| - 1]$ its position in the list of visited stops, and by t_s^r the date on which it is delivered. We have a vertex (t_s^r, c, r) in the route subgraph. We then have the arcs of Table 4 to explicitly model the flow of the commodity m through former and new routes, in addition to those defined in Table 1.

Table 4: Additional arcs of the commodity reinsertion commodity flow graph compared to Table 1. When not stated, **Min** is 0, **Cost** is 0 and **Max** is ∞.

Subgraph	Arc description	Origin	Destination	Min	Max	Cost
Routes	Transport $d \to c$ new route	$(t, d, \texttt{morning})$	$(t + \lfloor \frac{\tau_{dc}}{\tau_{\max}} \rfloor, c, \texttt{evening})$			
Routes	Transport $(d \to c_1)$ former route r	$(t^r, d, \texttt{morning})$	(t_1^r, c_1, r)		$\left\lfloor \frac{L_{\text{free}}^r}{\ell_m} \right\rfloor$	
Routes	Transport $(c_s \to c_{s+1})$ former route r	(t_s^r, c_s, r)	(t_{s+1}^r, c_{s+1}, r)			
Routes	Deliver c_s former route r	(t_s^r, c_s, r)	$(t_s^r, c_s, \texttt{evening})$			

Details of the vehicle flow graph. This flow graph is only used to model new direct routes. Indeed, all former routes are reused in this problem: we only choose the commodity flow through them. The only routing decision is related to the creation of new direct routes. Therefore, the *commodity reinsertion vehicles graph vertices* $\mathcal{V}_m^{\texttt{veh}}$ are the following: 1) Artificial vertices labelled: `source` and `sink`. 2)Depot vertices labelled: $(t, d, \texttt{morning})$ for $t \in [T]$, $d \in D_m$. 3) Customer vertices labelled: $(t, c, \texttt{evening})$ for $t \in [T]$, $c \in C_m$. Let $\mathcal{A}_m^{\texttt{veh}}$ be the arcs of the vehicle flow graph. They are detailed in Table 5. Instead of introducing this graph and the vehicle flow variable $\mathbf{x}$, we could have used the "vehicle fraction costs" defined in Section 4.1. Nonetheless, in this setting, since only one commodity flow variable is considered, we would have heavily underestimated the routing costs, creating vehicles that are almost empty in the end. Indeed, with vehicle fraction costs, filling only a small portion of a vehicle is not a problem.

Table 5: Arcs of the commodity reinsertion vehicle flow graph $\mathcal{D}_m^{\texttt{veh}}$. When not stated, **Min** is 0, **Cost** is 0 and **Max** is ∞.

Arc description	Origin	Destination	Min	Max	Cost
Start routes	`source`	$(t, d, \texttt{morning})$			
Transport $(d \to c)$	$(t, d, \texttt{morning})$	$(t + \lfloor \frac{\tau_{dc}}{\tau_{\max}} \rfloor, c, \texttt{evening})$			$c^{\texttt{veh}} + c^{\texttt{stop}} + c^{\texttt{km}} \Delta_{dc}$
End routes	$(t, c, \texttt{evening})$	`sink`			
Circulation	`source`	`sink`			
Circulation	`sink`	`source`			

Rebuilding routes. The commodity reinsertion subroutine proceeds as follows. From the information written in the $\mathbf{y}$ flow variable, it fills the former routes by decoding the corresponding flow. Then, it iteratively creates new direct routes to zero the corresponding commodity flow in $\mathbf{y}$. In contrast with the previous section, restricting ourselves to new direct routes is a limitation. It applies a `routing local search` to address the potential cost rise induced. The new solution is accepted by the LNS, even if its cost is higher.

7 Computational Experiments

First, we want to analyze the quality of the solutions of our LNS. We compare them with the route-based matheuristic in terms of cost. We hope to reduce both routing and inventory costs with our LNS, based on

the variety of neighborhoods and perturbations along different axes of the problem. Since we deal with a time limit criterion for practical reasons at Renault, we want to see how time is spread among the subroutines of our LNS. We are also interested in the ratio of cost gain per CPU time for each subroutine. Since the aim of our LNS is to modify the solution at a large scale, we intuitively want to spend time in the MILP-based large neighborhoods and perturbations. The routing local search may be quickly stuck in local minima. Second, we would like to know if each subroutine of our LNS is indeed useful to find good solutions. Last, we would like to know if our algorithms scale to a more general setting of long routes.

7.1 Experimental Setting

Available instances. Thanks to various interactions with Renault's operations research and machine learning team, we build 71 IRP instances, at the European scale and over roughly 20 days each. They correspond to the same industrial use case but to different periods. We show the instances' number of depots, number of customers and number of commodities on Figure 6. The maximum number of stops is $S_{max} = 3$ to comply with the car manufacturer requirements.

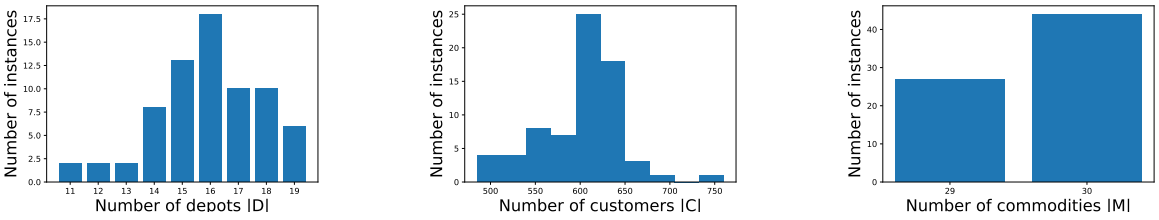


Figure 6: Histograms of the dimensions of the extracted instances.

We present results for a greater maximum number of stops $S_{\max} = 10$ to know if the algorithms scale to a more general framework.

Our code is in the Julia language (Bezanson et al. 2017). We use Gurobi optimizer (Gurobi Optimization, LLC 2021) to solve linear programs and MILPs, JuMP (Dunning, Huchette, and Lubin 2017) to model mathematical programs, and `Graphs.jl` (Fairbanks et al. 2021) to define graph structures. Both code and instances are publicly available.

7.2 Cost Results

The cost of the IRP solution after our LNS is the first natural metric we consider in order to evaluate our approach. We can compare it with the cost after the route-based matheuristic defined in Section 3. To do so, we run both the LNS and the route-based matheuristic over the European-scale instances illustrated by Figure 6 with a time limit of 90 minutes. On Figure 7, we show the box plots of the cost due to the depots' inventory, the customers' inventory, the customers' shortage, the vehicles, the stops and the kilometers. The orange lines indicate the median over the instances, the ends of the boxes the extreme quartiles ($Q1$ and $Q3$), and the whiskers the range $[Q1 - 1.5(Q3 - Q1), Q3 + 1.5(Q3 - Q1)]$. Outlier points correspond to data outside of the whiskers. The average total cost over the instances solved by the initialization + local search algorithm is 2.82M€, compared with 2.48M€ after the route-based matheuristic, and 2.14M€ after the LNS. We thus manage to get better IRP solutions with the LNS, inducing 11.8M€ saving per year compared with the initialization + local search algorithm, and 5.89M€ compared with the route-based matheuristic. On Figure 7 we notice not only one aspect of the cost is lower: the shortage costs, as well as the routing costs (vehicles costs, stop costs and kilometer costs) have quartiles corresponding to smaller values after the LNS. We thus emphasize the LNS manages to alter several key axes of an IRP solution, which is the aim of the different neighborhoods and perturbations we design.

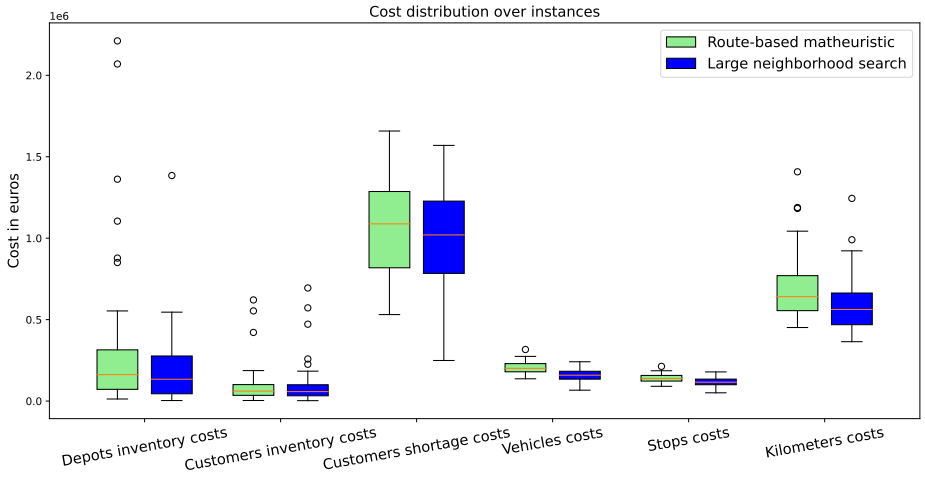


Figure 7: Box plots of the distribution of the solution cost per origin over instances. Green rectangles are related to the route-based matheuristic, blue ones to the LNS.

7.3 Analysis of the LNS Components

7.3.1 Time per Operator

We proceed to a more detailed analysis of the resolution steps illustrated on Figure 1. We first compare the total time spent in the initialization + local search algorithm, the routing local search, the fixed-path routes large neighborhood, and the customer and commodity reinsertion perturbations. For the context, after tuning the LNS parameters, we perform per outer step: one iteration of the routing local search, one pass over each depot for the fixed-path routes large neighborhood, 200 customer reinsertion steps, and 15 commodity reinsertion steps. To set these values, we strike a balance between our different operators to optimize the routing and inventory aspects of an IRP solution in the limited time we have. We emphasize that if we just proceed to one pass over every neighborhood and perturbation without restricting the depots, customers, or commodities considered for the MILP-based operators, we can barely do two outer iterations of the LNS in 90 minutes. With the parameters we set we do 4.5 outer iterations of the LNS on average. This is a major difficulty since most of the metaheuristic literature is based on the assumption that several passes over the operators can be done, paving the way for adaptive LNS for instance (Gendreau and Potvin 2010). On the left of Figure 8 we show the initialization + local search algorithm is indeed fast, taking 3.76 minutes on average. The total time spent in the routing local search amounts to 9.96 minutes on average. The fixed-path routes neighborhood, customer reinsertion and commodity reinsertion perturbations account for roughly similar fractions of the total duration, respectively 29.6, 26.5 and 20.4 minutes. This is one of our balancing criteria. Therefore, about 85% of the time is spent solving MILPs to modify the solution at a large scale. This is interesting in terms of code performance since we rely on the efficiency of the Gurobi solver (Gurobi Optimization, LLC 2021). We proceed to a warm start using the current solution for each MILP, and we profile our Julia code. The main difficulty is inherent in the routing local search, Algorithm 4. Indeed, as stated in Section 5, even a very local neighborhood involves substantial computations. Besides, because of the bindings between routing and inventory costs, selecting promising neighborhoods a priori is a challenge.

7.3.2 Cost Gain Over Time

On the right of Figure 8, we display the box plots of the cost gain per CPU time (in €/minute) of our four main LNS components: the routing local search, fixed-path routes neighborhood, customer reinsertion,

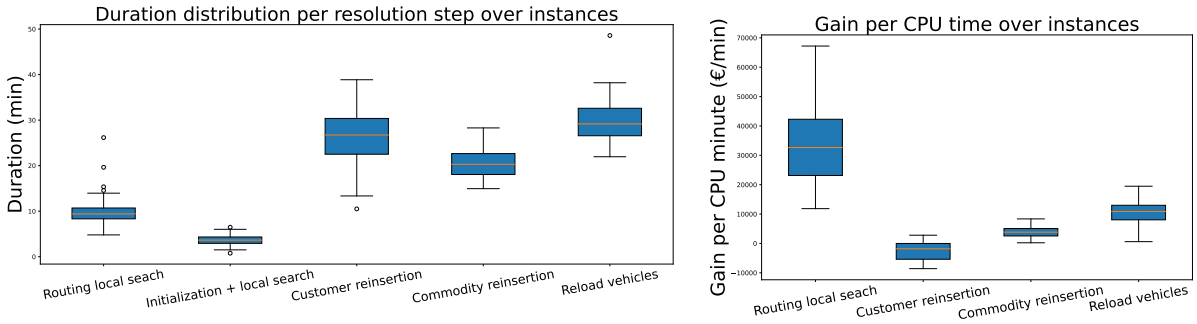


Figure 8: Box plots of the time spent, and of the cost gain per CPU time per operator.

and commodity reinsertion. We highlight that the routing local search has the highest gain per CPU time, 37.3k€/minute on average. Its large variance over the instances can be related to the balance between routing and inventory costs that may vary from one instance to another, depending on the demand and release profiles. The unit costs themselves do not change between instances. In spite of the difficulty mentioned in the previous paragraph, those neighborhoods are efficient and crucial. It is partly due to the structure of our perturbations: a customer or commodity reinsertion step creates new routes that are only direct. The reason for this choice is to avoid spending too much time solving greater perturbation MILPs that only involve a portion of the solution. It is therefore useful to combine them with routing neighborhoods that merge or mix routes in different manners.

The fixed-path routes neighborhood, customer and commodity reinsertion perturbations alter both routing and inventory variables, with 10.7k€/minute, −6.33k€/minute and 4.40k€/minute ratios on average. Therefore, most of the customer reinsertion perturbation steps, as well as some commodity reinsertion perturbation steps, increase the solution cost. They are designed to escape from local minima. While tuning LNS parameters, we notice that when restricting the amount of commodities or customers considered per outer LNS iteration, the descent neighborhoods get quicker stuck in local minima and the final solution cost is greater. We show further details with ablation tests in Section 7.3.3. We emphasize that separating the gains per operator is not totally obvious. It is indeed the mix between the axes of the IRP (from routes to commodities and customers) that enables us to substantially reduce the cost in our LNS. The performance of each operator alone is less crucial than the interactions between them.

7.3.3 Ablation Tests

In order to further highlight the role of the fixed-path routes large neighborhood and reinsertion perturbations, we proceed to ablation tests. The reference is our LNS run during 90 minutes with the number of inner steps per outer iteration described in Section 7.3.1. During ablation tests, we run it with the same parameters and time limit, but without the customer reinsertion, commodity reinsertion or fixed-path routes neighborhood respectively. We compare the results of the gaps on Figure 9 with plots of cumulative distributions over instances with respect to the gap of the solutions. We must emphasize our lower bound derived in Section 4.1 is not tight, leading to high gaps by definition. It seems therefore relevant to use them as a relative metric to compare the solution processes, and not as an absolute indicator on a solution quality. The first remark we can make is that the route-based matheuristic (blue curve) performs better than our initialization + local search algorithm (orange curve), but worse than the LNS, even when one of its components is removed. This confirms the conclusion drawn from the cost analysis in Section 7.2, emphasizing the performance of our LNS. Besides, we show that whatever is the component removed from the LNS, the resulting algorithm with same time budget leads to worse solutions. Indeed, overall, the curve of the cumulative distribution of our whole LNS (brown) is above any other curve. This is particularly true when removing the reload fixed-path vehicles neighborhood (purple), but also for the customer (red) and commodity reinsertion (green) ablations that lead to similar performance. Both perturbation ablations result in 5% additional average gap compared

to the whole LNS. We expect this trend to be accentuated when the time limit increases, since looping over neighborhoods and perturbations is key to escape from local minima.

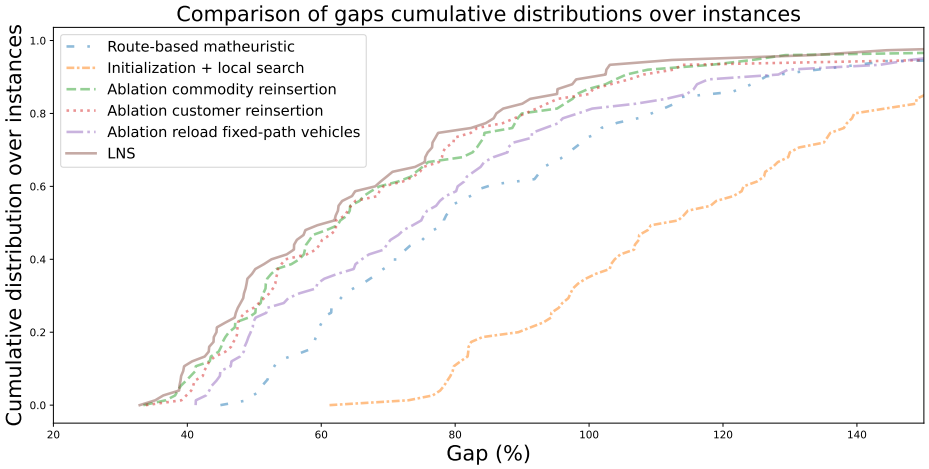


Figure 9: Cumulative distributions of the gap among instances solutions.

7.4 Results with Longer Routes

As stated above, limiting the routes to $S_{\max} = 3$ stops maximum is restrictive, though this is an important constraint in the current operations of Renault. Indeed, longer routes imply a huge additional workload for the drivers.

To emphasize the benefits in terms of costs and gaps induced by longer routes, we solve the same instances as previously, but with $S_{\max} = 10$, a time limit of 180 minutes, and 100 customer reinsertion steps per LNS outer iteration. Every other parameter remains the same. We summarize the average cost impact with Table 6. We show that we can reach lower costs with each algorithm, with a large gain after the initialization + local search algorithm. We also demonstrate our LNS scales to this more general setting. To complete the study, we display the gap cumulative distributions of our three algorithms on Figure 10. We observe smaller gaps in this setting, which was expected since we relax the constraint of maximum number of stops in a route.

We emphasize our LNS manages to significantly improve gaps compared with the route-based matheuristic and initialization + local search algorithm, at the price of heavier computations. Indeed, although the time limit is set equal for each solution process, the iterative LNS reaches it, and the two other methods stop earlier.

The choice and size of the neighborhoods is therefore a key to adapt the route-based matheuristic and the LNS to new instances. Since this study is motivated by an industrial context where the IRP is solved every day, we can hint at some statistical ways to do so. A perspective for further research may be the use of machine learning techniques, either for tuning our algorithms parameters, or to enhance their building subroutines with structured prediction. Besides, additional computations could be useful to analyze in details the influence of $S_{\max}$, with intermediate values between 3 and 10.

Table 6: Comparison of the average costs over instances between $S_{\max} = 3, 10$

	Initialization + local search	**Matheuristic**	**LNS**	**Lower bound**
Average cost $S_{\max} = 3$	2.82M€	2.48M€	2.14M€	1.35M€
Average cost $S_{\max} = 10$	2.38M€	2.20M€	1.88M€	1.35M€

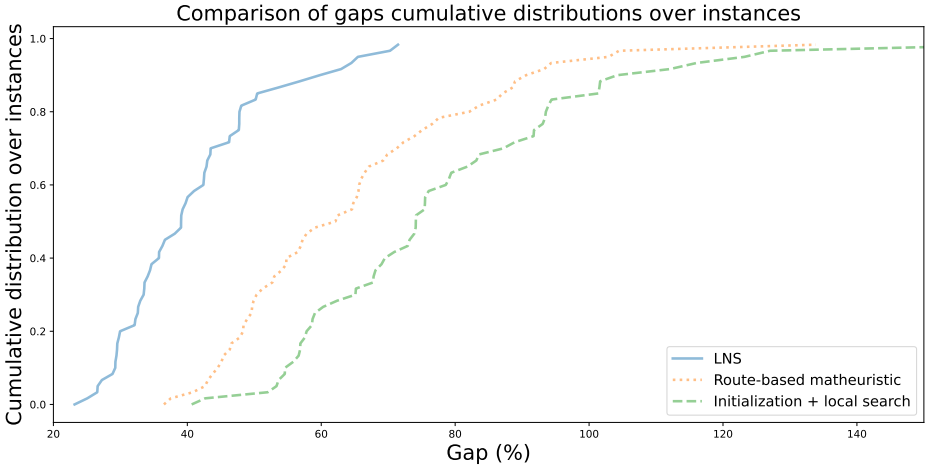


Figure 10: Cumulative distributions of the gap for $S_{\max} = 10$ and 180 minutes time limit.

8 Conclusion

In this study motivated by an industrial partnership with Renault, we consider European-scale multi-attribute IRP instances with route durations. This inherently hard problem has to be solved in a limited time of 90 minutes every day. To do so, we design a large neighborhood search based on the generalization of TSP and SDVRP neighborhoods Section 5, a large neighborhood Section 6.1 inspired by recent matheuristics designed for the IRP (Bertazzi et al. 2019; Archetti, Boland, and Speranza 2017), and two new perturbations Section 6.2 and Section 6.3 based on MILPs. We also derive an initialization + local search algorithm that relies on flows Section 4.1. It allows us to quickly initialize our IRP instances with non-trivial solutions, and to derive a lower bound. To the best of our knowledge, this lower bound is unknown in the IRP literature. We extract and process a dataset of 71 European-scale multi-attribute IRP instances that we make available publicly. Numerical experiments in Section 7 show the results of our LNS. We highlight that it outperforms the route-based matheuristic, and that each component of the LNS brings useful contributions.

We also emphasize some limits and perspectives. The choice of the neighborhoods is hard to define a priori, and we do not have time to browse the space of neighborhoods repeatedly. Parallel computing and multi-threading could be some perspectives to consider, although even small changes imply inventory dynamics over the whole time horizon, leading to overlaps between neighborhoods. Cache use would thus be a challenge. Some techniques of machine learning for operations research could also be considered in this direction, in a reinforcement learning (Wu et al. 2021) or structured learning (Parmentier 2022) paradigm for example. Besides, we suffer from the poor quality of our lower bound to compute gaps. Deriving a better relaxation for this multi-attribute IRP is a challenge that has not been addressed in the literature to the best of our knowledge. We could add valid inequalities leveraging the research on exact solutions of smaller problems (Manousakis et al. 2021; Desaulniers, Rakke, and Coelho 2015) to improve the quality of our neighborhoods. When dealing with real data from Renault, solving the IRP in a rolling horizon framework to prepare code industrialization, we face release and demand randomness. A stochastic IRP (Nolz, Absi,

and Feillet 2014; Coelho, Laporte, and Cordeau 2012) may be necessary over the large-scale instances to enable robustness. We plan to explore those research paths in our future work.

Acknowledgments

We are grateful to the Renault supply-chain and IT teams for the partnership we have set and the industrial motivation they provide us with, especially to Alain Nguyen, Thaddeus Leonard, Nicusor-Eugen Plescan, Christian Serrano, Ludovic Doudard and Aimé-Frédéric Rosenzweig. We also would like to thank Vincent Leclère for his advice on the form of the article.

References

Archetti, C., L. Bertazzi, G. Laporte, and M. G. Speranza (Aug. 2007). "A Branch-and-Cut Algorithm for a Vendor-Managed Inventory-Routing Problem". In: *Transportation Science* 41.3, pp. 382–391. ISSN: 0041-1655. DOI: 10/b9kkm7 (cit. on p. 2).

Archetti, C., N. Boland, and M. G. Speranza (Apr. 2017). "A Matheuristic for the Multivehicle Inventory Routing Problem". In: *INFORMS Journal on Computing* 29.3, pp. 377–387. ISSN: 1091-9856. DOI: 10/gbsn75 (cit. on pp. 2, 7, 24).

Archetti, C., G. Guastaroba, D. L. Huerta-Muñoz, and M. G. Speranza (Nov. 2021). "A Kernel Search Heuristic for the Multivehicle Inventory Routing Problem". In: *International Transactions in Operational Research* 28.6, pp. 2984–3013. ISSN: 0969-6016, 1475-3995. DOI: 10/gkzwqh (cit. on pp. 2, 7).

Archetti, C. and M. G. Speranza (2016). "The Inventory Routing Problem: The Value of Integration". In: *International Transactions in Operational Research* 23.3, pp. 393–407. ISSN: 1475-3995. DOI: 10/gctq3d (cit. on p. 1).

Archetti, C. and M. G. Speranza (2008). "The Split Delivery Vehicle Routing Problem: A Survey". In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Ed. by B. Golden, S. Raghavan, and E. Wasil. Operations Research/Computer Science Interfaces. Boston, MA: Springer US, pp. 103–122. ISBN: 978-0-387-77778-8. DOI: 10.1007/978-0-387-77778-8_5 (cit. on p. 2).

Benoist, T., F. Gardi, A. Jeanjean, and B. Estellon (Mar. 2011). "Randomized Local Search for Real-Life Inventory Routing". In: *Transportation Science* 45.3, pp. 381–398. ISSN: 0041-1655. DOI: 10.1287/trsc.1100.0360 (cit. on p. 2).

Bertazzi, L., L. C. Coelho, A. De Maio, and D. Laganà (Feb. 2019). "A Matheuristic Algorithm for the Multi-Depot Inventory Routing Problem". In: *Transportation Research Part E: Logistics and Transportation Review* 122, pp. 524–544. ISSN: 1366-5545. DOI: 10/ggdvkf (cit. on pp. 2, 7, 24).

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah (2017). "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1, pp. 65–98. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671 (cit. on pp. 3, 20).

Campbell, A. M. and M. W. P. Savelsbergh (Nov. 2004). "A Decomposition Approach for the Inventory-Routing Problem". In: *Transportation Science* 38.4, pp. 488–502. ISSN: 0041-1655. DOI: 10/fkn54g (cit. on p. 2).

Coelho, L. C. and G. Laporte (Nov. 2013). "A Branch-and-Cut Algorithm for the Multi-Product Multi-Vehicle Inventory-Routing Problem". In: *International Journal of Production Research* 51.23-24, pp. 7156–7169. ISSN: 0020-7543. DOI: 10/gjvnnf (cit. on p. 2).

Coelho, L. C., A. De Maio, and D. Laganà (Dec. 2020). "A Variable MIP Neighborhood Descent for the Multi-Attribute Inventory Routing Problem". In: *Transportation Research Part E: Logistics and Transportation Review* 144, p. 102137. ISSN: 13665545. DOI: 10/gm954p (cit. on pp. 2, 7).

Coelho, L. C., G. Laporte, and J.-F. Cordeau (2012). *Dynamic and Stochastic Inventory-Routing*. Tech. rep. CIRRELT-2012-37. CIRRELT Montreal (cit. on p. 25).

Cordeau, J.-F., D. Laganà, R. Musmanno, and F. Vocaturo (Mar. 2015). "A Decomposition-Based Heuristic for the Multiple-Product Inventory-Routing Problem". In: *Computers & Operations Research* 55, pp. 153–166. ISSN: 03050548. DOI: 10/f6w8pp (cit. on p. 2).

Desaulniers, G., J. G. Rakke, and L. C. Coelho (Oct. 2015). "A Branch-Price-and-Cut Algorithm for the Inventory-Routing Problem". In: *Transportation Science* 50.3, pp. 1060–1076. ISSN: 0041-1655. DOI: 10/f8zpgc (cit. on pp. 2, 24).

Dror, M. and P. Trudeau (1990). "Split Delivery Routing". In: *Naval Research Logistics (NRL)* 37.3, pp. 383–402. ISSN: 1520-6750. DOI: 10.1002/nav.3800370304 (cit. on p. 2).

Dunning, I., J. Huchette, and M. Lubin (2017). "JuMP: A Modeling Language for Mathematical Optimization". In: *SIAM Review* 59.2, pp. 295–320. DOI: 10.1137/15M1020575 (cit. on p. 20).

Fairbanks, J., M. Besançon, S. Simon, J. Hoffiman, N. Eubank, and S. Karpinski (2021). *JuliaGraphs/Graphs.jl: an optimized graphs package for the Julia programming language*. URL: https://github.com/JuliaGraphs/Graphs.jl/ (cit. on p. 20).

Fischetti, M. and M. Fischetti (2016). "Matheuristics". In: *Handbook of Heuristics*. Ed. by R. Martí, P. Panos, and M. G. Resende. Cham: Springer International Publishing, pp. 1–33. ISBN: 978-3-319-07153-4. DOI: 10.1007/978-3-319-07153-4_14-1 (cit. on p. 2).

Gendreau, M. and J.-Y. Potvin, eds. (2010). *Handbook of Metaheuristics*. Vol. 146. International Series in Operations Research & Management Science. Boston, MA: Springer US. ISBN: 978-1-4419-1663-1 978-1-4419-1665-5. DOI: 10.1007/978-1-4419-1665-5 (cit. on p. 21).

Gurobi Optimization, LLC (2021). *Gurobi Optimizer Reference Manual*. URL: https://www.gurobi.com (cit. on pp. 20, 21).

Lagos, F., N. Boland, and M. Savelsbergh (Jan. 2020). "The Continuous-Time Inventory-Routing Problem". In: *Transportation Science*, trsc.2019.0902. ISSN: 0041-1655, 1526-5447. DOI: 10.1287/trsc.2019.0902 (cit. on p. 1).

Manousakis, E., P. Repoussis, E. Zachariadis, and C. Tarantilis (May 2021). "Improved Branch-and-Cut for the Inventory Routing Problem Based on a Two-Commodity Flow Formulation". In: *European Journal of Operational Research* 290.3, pp. 870–885. ISSN: 03772217. DOI: 10.1016/j.ejor.2020.08.047 (cit. on pp. 2, 24).

Nolz, P. C., N. Absi, and D. Feillet (Jan. 2014). "A Stochastic Inventory Routing Problem for Infectious Medical Waste Collection". In: *Networks* 63.1, pp. 82–95. ISSN: 00283045. DOI: 10.1002/net.21523 (cit. on pp. 2, 24).

Parmentier, A. (Jan. 2022). "Learning to Approximate Industrial Problems by Operations Research Classic Problems". In: *Operations Research* 70.1, pp. 606–623. ISSN: 0030-364X, 1526-5463. DOI: 10.1287/opre.2020.2094 (cit. on p. 24).

Savelsbergh, M. and J.-H. Song (July 2008). "An Optimization Algorithm for the Inventory Routing Problem with Continuous Moves". In: *Computers & Operations Research* 35.7, pp. 2266–2282. ISSN: 03050548. DOI: 10/ft9dtw (cit. on p. 1).

Su, Z., Z. Lü, Z. Wang, Y. Qi, and U. Benlic (Mar. 2020). "A Matheuristic Algorithm for the Inventory Routing Problem". In: *Transportation Science*, trsc.2019.0930. ISSN: 0041-1655, 1526-5447. DOI: 10/gm954w (cit. on p. 2).

Wu, Y., W. Song, Z. Cao, and J. Zhang (2021). "Learning Large Neighborhood Search Policy for Integer Programming". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., pp. 30075–30087. URL: https://proceedings.neurips.cc/paper/2021/file/fc9e62695def29ccdb9eb3fed5b4c8c8-Paper.pdf (cit. on p. 24).

Table 7: Notations

Name	Description
t	Day
T	Horizon
d	Depot (facility)
D	Set of depots
c	Customer
C	Set of customers
m	Commodity
M	Set of commodities
M_c	Set of commodities used by customer c
M_d	Set of commodities used by depot d
$\mathbf{I}^0_{md}$	Initial inventory of m at d
$\mathbf{I}^0_{mc}$	Initial inventory of m at c
b^+_{mdt}	Number of commodity m released on day t by d
b^-_{mct}	Number of commodity m demanded on day t by c
κ_{mdt}	Free inventory capacity of m at d on the evening of day t
κ_{mct}	Free inventory capacity of m at c on the evening of day t
$\mathcal{V}$	Set $D \cup C$ of the vertices of the locations graph
v	Node of $\mathcal{V}$
$\mathcal{A}$	Arcs of the locations graph
a	Arc
Δ_a	Distance in kilometers corresponding to arc a
τ_a	Duration in hours corresponding to arc a
P	Path in the locations graph
S_{max}	Maximum number of stops in a route
$\tau_{\max}$	Number of transport hours per day
ℓ_m	Length of a commodity m
L	Length of a vehicle (homogeneous)
$c^{\texttt{exc}}_{md}$	Unit excess inventory storage cost of m at d per night
$c^{\texttt{exc}}_{mc}$	Unit excess inventory storage cost of m at c per night
$c^{\texttt{short}}_{mc}$	Unit shortage cost of m at c per day
$c^{\texttt{km}}$	Per kilometer cost of the routes
$c^{\texttt{veh}}$	Unit cost for using a vehicle
$c^{\texttt{stop}}$	Unit cost for making a stop

A Notations

In Table 7, we recap the main concepts and notations the dimensions, the initial inventory, the demand and release, the free inventory capacities, the locations graph, the commodities and vehicles lengths, and the unit costs.