Inbound and Outbound Flow Integration for Cross-Docking Operations

Marc-Antoine Coindreau[a], Olivier Gallay[a,*], Nicolas Zufferey[b,d], Gilbert Laporte[c,d,e]

[a] *Department of Operations, HEC, University of Lausanne, CH-1015 Lausanne, Switzerland*

[b] *Geneva School of Economics and Management, University of Geneva, Uni-Mail, CH-1211 Geneva 4, Switzerland*

[c] *HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal H3C 3J7, Canada*

[d] *Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT), Montréal, Canada*

[e] *University of Bath, Claverton Down, Bath BA2 7AY, United Kingdom*

Abstract

We consider the optimization of the cross-docking operations at three INtermodal LOgistics Platforms (INOLPs) of a large European car manufacturer (ECM). The planning horizon is a week and the time bucket is a day. An inbound flow of products is gradually received over the week by truck from inland suppliers, and has to be loaded into containers which are then shipped to offshore production plants. The full content of a container must be available at the INOLP to enable its loading operations to start, hence temporary storage is needed. The objective is to minimize an inventory penalty, computed as the largest daily volume of temporary product storage observed over the planning horizon. The current practice at ECM is to first optimize the content of the inbound trucks and of the outbound containers independently, and then determine the loading day of each container to be shipped based on these fixed contents. We propose to integrate, within the same optimization framework, the decisions on both truck and container contents, which involve complex loading constraints related to the dimensions and weights of the products, with those on the scheduling of container loading. We model the resulting problem as a mixed integer linear program, and we develop a decomposition scheme for it, as well as a fix-and-optimize matheuristic. We perform extensive computational experiments on real instances provided by ECM. Results show that a combination of these two matheuristics is able to generate solutions that reduce the average

*Corresponding author

Email addresses: marc-antoine.coindreau@unil.ch (Marc-Antoine Coindreau), olivier.gallay@unil.ch (Olivier Gallay), n.zufferey@unige.ch (Nicolas Zufferey), gilbert.laporte@hec.ca (Gilbert Laporte)

Published in *European Journal of Operational Research*

inventory penalty by 40%.

Keywords: Logistics, cross-dock scheduling, matheuristic, fix-and-optimize.

1. Introduction

We model and solve an operations management problem encountered by a large European car manufacturer (denoted here as ECM as a result of a non-disclosure agreement) which consolidates product flows from inland suppliers to offshore production plants at INtermodal LOgistics Platforms (INOLPs). Over a given planning horizon (from Monday to Friday in this work), the products, which are collected by trucks at different supplier locations, are first unloaded and repacked at the INOLP. The products are then immediately loaded into containers, or temporarily stored until a full container content is available at the INOLP, hence allowing the loading operations to be launched. It is assumed that the necessary products for all container contents are received by truck over the week, hence allowing all planned container loading operations to take place. The containers are finally sent by ship at the end of the week to offshore production plants, which are the INOLP clients. We refer to this problem as the ECM Problem. Figure 1 illustrates the sequence of operations just described.

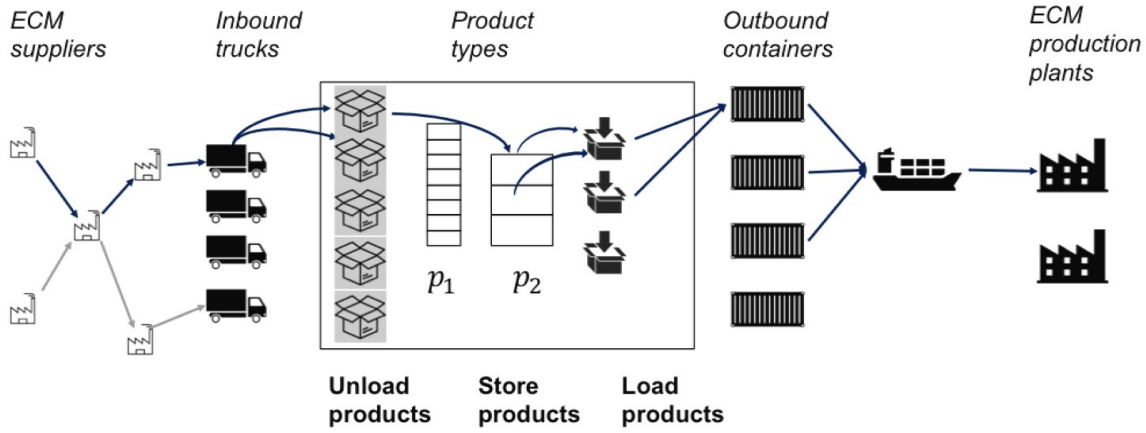

Figure 1: Product flow in an INOLP.

Inbound truck transportation is subcontracted. As a consequence, the truck routes cannot be modified, as they are contractually fixed for the long term. The complex routing subproblems associated with the inbound trucks have been previously and independently solved by ECM. They can typically be modeled as a Traveling Purchaser Problem (Boctor et al. 2003). However, the

INOLP managers can still decide which products should be collected on the truck routes.

Regarding the outbound side, a container can only be loaded after its full content has been delivered to the INOLP. This entails temporary storage, which generates inventory costs at the INOLP. Furthermore, high inventory levels may lead to an imbalanced workload since the stored products will ultimately have to be loaded into containers in the last days of the week. Therefore, ECM aims at minimizing, over a one-week planning horizon a penalty computed as the largest daily inventory volume required at the INOLP. To this end, three different types of decisions are inherent to the ECM problem: determining the contents of the trucks, that of the containers, and the loading day of each container.

At the end of the week, the demand of each client, i.e., the requested quantity of each product type, must be satisfied and loaded in its assigned containers. The products arriving on the inbound side can be sent to any outbound client requesting them. Since the containers are all sent by boat at the end of the week, the container loading sequence is unconstrained. Additionally, the number of containers loaded per day is unlimited.

Loading problems at inbound and outbound sides are rather complicated since they involve three-dimensional constraints (the three dimensions of the products are taken into account when loading, and overlaying is forbidden), a total weight limitation (the total weight of all products loaded in the same container cannot exceed 20 tonnes for the trucks and 22 tonnes for the containers), and specific arrangements of packaged products in stacks. In the latter case, the position of the products in a stack is constrained by its weight (for each product, there is weight limitation on the products to be stacked above it). For an overview of the loading constraints tackled by ECM, see (Toffolo et al. 2017), and, for an exhaustive overview and classification of common and practical loading constraints, see Bortfeldt and Wäscher (2013).

Because of such complex loading constraints, reassigning a product from a truck or a container to another one is not straightforward. Yet the problem is slightly simplified since the products are packaged into standardized boxes, and the loading constraints concern the box types only, irrespective of the products they contain. The complete consideration of the set of loading constraints is extremely complex, hence we restrict the solution space to product permutations between boxes of the same type. While this simplification allows to control the size of the problem, it still gives rise to a rich solution space to explore. Indeed, it has been observed that more than 70% of the boxes

are filled with different products but with a weight variation of less than 10 kg, which precludes any violation of the weight of a stack. Hence, starting from a feasible assignment of boxes to trucks or containers, numerous different assignments of products to boxes are possible. As a consequence, product permutations between boxes can be applied to optimize the truck or the container contents, while ensuring that the loading constraints will be satisfied.

The current practice at ECM is to determine the loading day of each container over the planning horizon, without revoking previously made decisions. Indeed, in a first phase, ECM uses standalone optimization tools to independently determine the truck and container contents. These contents are taken as inputs to a second phase dedicated to the scheduling of container loading operations. A particular case of the ECM problem was investigated in Coindreau et al. (2019), in which the truck contents are fixed and the decisions focus only on the loading day and the content of the containers. Even if substantial reductions were already observed for the inventory penalty, our work aims at extending the previous study by further integrating the decisions on the truck contents.

This paper makes the following scientific contributions. We introduce the ECM problem which integrates the optimization of both inbound and outbound product flows with container scheduling. To solve the ECM problem, we propose a mixed integer linear programming (MILP) model as well as two matheuristics, namely a decomposition matheuristic (DM) and a fix-and-optimize matheuristic (FOM). We perform extensive computational tests on the instances provided by ECM. We demonstrate the inventory penalty reduction resulting from our integrated approach.

The remainder of this paper is organized as follows. Section 2 provides a survey of the related literature. Section 3 introduces the MILP formulation of the ECM problem. Section 4 presents our two matheuristics and Section 5 summarizes their performance on real and on generated instances. Section 6 quantifies the gain achieved by our integrated approach with respect to the non-integrated current practice. This is followed by conclusions and perspectives in Section 7.

2. Literature Review

We first review the cross-docking literature that shares some similarities with the ECM problem. Next, we give an overview of matheuristics that are relevant for the present case, and we focus in particular on FOMs.

A cross-docking facility aims at consolidating inbound and outbound flows (here, from inland suppliers to offshore production plants) by making, as much as possible, direct product transfers from trucks to containers (Van Belle et al. 2012), hence avoiding excessive inventory at the logistic platforms (which differs from warehousing). As in Buijs et al. (2014), we consider a single cross-dock in a many-to-few network configuration, which means that the cross-dock under consideration is linked to many suppliers (here up to a few thousand) but only to a few customers (here less than 20 customers per INOLP).

More specifically, the ECM problem shares some features of the truck scheduling in a cross-dock (TSCD), for which a review can be found in Boysen and Fliedner (2010). In the TSCD, the unloading and loading operations of the trucks are viewed as a set of jobs, as defined in the job scheduling literature. The aim of the TSCD is to determine a sequence of inbound trucks arriving at the cross-docking platform and a sequence of outbound containers that are then loaded, in order to minimize a given objective, e.g., the makespan, as in Chen and Lee (2009) and Ye et al. (2018)). Whereas in some cases, the product transfers can be done without any need for temporary product storage (Boysen 2010), other situations require a temporary inventory (as for the ECM problem, momentary storage is observed in Yu and Egbelu (2008)). Despite its similarities with cross-docking, the ECM configuration precludes the use of the existing related methodologies. First, direct transfers of products from trucks to containers cannot always take place in the INOLP due to the constraints imposed on the scheduling of operations related to container loading. Indeed, it is required that the whole content of a container be available at the INOLP before proceeding to its loading. This creates an increased need for temporary storage that is rarely observed in standard cross-docking configurations. Second, and in contrast with the TSCD, the content of the trucks and that of the containers are modified during the optimization of the operations, and hence jobs can no longer be defined by a set of products to be unloaded and then loaded, as is done, for example, in Bellanger et al. (2013). Focusing solely on the inbound side, Serrano et al. (2017) consider the reassignment of the content of inbound trucks in a container scheduling context. In contrast with the ECM problem, a simplifying assumption is made by considering scalar loading constraints. To the best of our knowledge, no existing work provides loading solutions that ensure the non-violation of the complex loading constraints considered here and described in Toffolo et al. (2017). We refer to Coindreau et al. (2019) for a more extensive review of related cross-docking problems.

Several contributions acknowledge the importance of integrating internal cross-docking decisions.

Indeed, the cross-docking literature covers a large number of problems that are usually solved sequentially (e.g., from a strategic, tactical, and operational point of views). For instance, Rijal et al. (2019) show that simultaneously making decisions on truck scheduling and decisions on the assignment of dock doors to inbound or outbound flows leads to cost savings as high as 10%. These problems are usually solved sequentially, as dock doors are first assigned to inbound or outbound flows (tactical decisions) and then trucks are scheduled on the resulting door assignment (operational decisions). In the same vein, Tadumadze et al. (2019) show the importance of integrating the decisions on truck scheduling with those concerning workforce planning. Other contributions analyze the integration of cross-docking decisions with those concerning the supply chain. Among others, Enderer et al. (2017) highlight the gain that can be obtained when integrating dock door assignment with decisions concerning the routing of trucks. Despite the large number of papers that consider integrated decisions in a cross-docking context, no existing contribution considers the full potential that is offered by integrating decisions on both inbound and outbound flow scheduling with the decisions on container loading, as proposed in the present paper.

In a recent paper, Coindreau et al. (2019) have proposed a decomposition matheuristic for a sub-case of the ECM problem. We present below suitable types of matheuristics that can help tackle the further complexity brought by the integration of decisions for the truck content, in addition to considering those for the container content and container scheduling. Matheuristics typically combine mathematical programming and heuristics (Jourdan et al. 2009). Among the wide existing range of available matheuristics, FOMs (originally introduced by Gintner et al. (2005)) consist in iteratively fixing a subset of decision variables and then let generic solver find solutions on a smaller and simpler problem. Repeatedly fixing some variables and optimizing some others often allows to outperform the direct use of a solver applied to the full set of variables. In particular, FOM has been successfully applied to lot sizing (Sahling et al. 2009, Helber and Sahling 2010), timetabling (Dorneles et al. 2014), and location-routing problems (Rieck et al. 2014). Recently, FOM has been combined with generic metaheuristic frameworks such as variable neighborhood search (VNS) in Della Croce and Salassa (2014) and in Chen (2015), or variable neighborhood descent (VND) in Dorneles et al. (2014). These papers indicate that combining FOM with a metaheuristic outperforms the use of FOM only.

3. Mathematical formulation

Section 3.1 introduces the variables and sets related to the ECM problem. Section 3.2 presents a MILP model for minimizing the proposed inventory penalty (i.e., the largest temporary storage required over the week) by making decisions on the content of a given subset of trucks and containers. Section 3.3 describes specific configurations of the MILP that are relevant for the ECM problem.

3.1. Sets, parameters and variables

The superscripts "(in)" and "(out)" refer to inbound and outbound, respectively. Furthermore, "(nf)" and "(f)" refer to the set of trucks or containers for which the content is not fixed and fixed, respectively.

Sets:

- T: set of time periods (i.e., days),
- C: set of clients,
- P: set of product types,
- S: set of suppliers,
- B: set of box types,
- I: set of inbound trucks, which contains the following subsets:
  - $I^{(nf)}$: subset of trucks for which the content is not fixed and can therefore be optimized with the MILP,
  - $I^{(f)}$: subset of trucks for which the content is fixed,
  - I_t: subset of trucks that arrive on day $t \in T$,
  - $I_t^{(nf)}$: subset of trucks that arrive on day $t \in T$, for which the content is not fixed,
  - $I_t^{(f)}$: subset of trucks that arrive on day $t \in T$, for which the content is fixed,
- O: set of outbound containers, which contains the following subsets:
  - $O^{(nf)}$: subset of containers for which the content is not fixed and can therefore be optimized with the MILP,

– $O^{(f)}$: subset of containers for which the content is fixed,

– O_c: subset of containers assigned to client $c \in C$,

– $O_c^{(f)}$: subset of containers assigned to client $c \in C$, for which the content is fixed,

– $O_c^{(nf)}$: subset of containers assigned to client $c \in C$, for which the content is not fixed.

Parameters:

- d_{cp}: demand (in units) of client $c \in C$ for product type $p \in P$,
- $n_{ib}^{(in)}$: number of units of boxes of type $b \in B$ transported in truck $i \in I$,
- $n_{ob}^{(out)}$: number of units of boxes of type $b \in B$ transported in container $o \in O$,
- $\pi_{pi} = 1$ if truck $i \in I$ visits the supplier that can provide product type $p \in P$, $\pi_{pi} = 0$ otherwise,
- q_{pb}: number of units of product type $p \in P$ that can be transported in box type $b \in B$,
- $q_{op}^{(out)}$: number of products of type $p \in P$ sent by container $o \in O^{(f)}$,
- $q_{ip}^{(in)}$: number of products of type $p \in P$ delivered by truck $i \in I^{(f)}$,
- l_{pb}: weight (in kg) of a box of type $b \in B$ when filled with product type $p \in P$,
- $l^{(in)}$: maximum allowed weight (in kg) that can be transported by a truck,
- $l^{(out)}$: maximum allowed weight (in kg) that can be transported by a container,
- h_p: volume (in m^3) of a product of type $p \in P$,
- g_p: number of units of product type $p \in P$ available in the inventory at the beginning of the week; due to various reasons (e.g., lot sizing or wrong orders), there is an initial inventory in the INOLP that cannot be determined in the optimization process and is therefore taken as an input (the magnitude of this initial inventory is detailed later in column V^{init} of Table 2),
- $M = \max_{o \in O} \left\{ \sum_{b \in B} q_{pb} \cdot n_{ob} \right\}$: a sufficiently large number needed for the linearization.

Decision variables:

- z_{ibp}: number of boxes of type $b \in B$ assigned to product type $p \in P$ in truck $i \in I^{(nf)}$,
- x_{obp}: number of boxes of type $b \in B$ assigned to product type $p \in P$ in container $o \in O^{(nf)}$,
- $y_{ot} = 1$ if container $o \in O$ is loaded on day $t \in T$; $y_{ot} = 0$ otherwise,
- w_{opt}: number of units of product type $p \in P$ sent by container $o \in O^{(nf)}$ on day $t \in T$,

- u_{pt}: number of units of product type $p \in P$ in stock on day $t \in T$ before loading the containers,
- v_{pt}: number of units of product type $p \in P$ in stock on day $t \in T$ after loading the containers,
- r_{pt}: number of units of product type $p \in P$ received on day $t \in T$,
- s_{pt}: number of units of product type $p \in P$ sent on day $t \in T$,
- f: largest inventory-penalty value (in m^3) encountered during the planning horizon.

3.2. Mixed integer linear programming formulation: $Q(O^{(nf)}, I^{(nf)})$

We denote by $Q(O^{(nf)}, I^{(nf)})$ the MILP formulation of the ECM problem for which the content of the $I^{(nf)}$ trucks and the $O^{(nf)}$ containers can be revoked and optimized. The problem is stated as follows:

$$\text{minimize} \quad f \tag{1}$$

subject to

$$f \geq \sum_{p \in P} h_p \cdot v_{pt} \qquad t \in T \tag{2}$$

$$v_{pt} = u_{pt} - s_{pt} \qquad p \in P,\, t \in T \tag{3}$$

$$v_{p0} = g_p \qquad p \in P \tag{4}$$

$$u_{pt} = v_{p,t-1} + r_{pt} \qquad p \in P,\, t \in T \tag{5}$$

$$r_{pt} = \sum_{b \in B} \sum_{(i \in I_t^{(nf)} | \pi_{pi} > 0)} q_{pb} \cdot z_{ibp} + \sum_{i \in I_t^{(f)}} q_{ip}^{(in)} \qquad p \in P,\, t \in T \tag{6}$$

$$s_{pt} = \sum_{o \in O^{(nf)}} w_{opt} + \sum_{o \in O^{(f)}} q_{op}^{(out)} \cdot y_{ot} \qquad p \in P,\, t \in T \tag{7}$$

$$\sum_{t \in T} y_{ot} = 1 \qquad o \in O \tag{8}$$

$$w_{opt} \leq M \cdot y_{ot} \qquad t \in T,\, o \in O,\, p \in P \tag{9}$$

$$w_{opt} \leq \sum_{b \in B} q_{pb} \cdot x_{obp} \qquad t \in T,\, o \in O,\, p \in P \tag{10}$$

$$\sum_{o \in O_c^{(nf)}} \sum_{t \in T} w_{opt} = d_{cp} - \sum_{o \in O_c^{(f)}} m_{op}^{f} \qquad c \in C,\, p \in P \tag{11}$$

$$\sum_{b \in B} \sum_{p \in P} l_{pb} \cdot x_{obp} \leq l^{(out)} \qquad o \in O^{(nf)} \tag{12}$$

$$\sum_{p \in P} x_{obp} \leq n_{ob}^{(out)} \qquad o \in O^{(nf)},\, b \in B \tag{13}$$

$$\sum_{p \in P | \pi_{pi} > 0} z_{ibp} \leq n_{ib}^{(in)} \qquad i \in I^{(nf)},\, b \in B \tag{14}$$

$$\sum_{b \in B} \sum_{p \in P} l_{pb} \cdot z_{ibp} \leq l^{(in)} \qquad i \in I. \tag{15}$$

$$\sum_{b \in B} z_{ibp} \leq \pi_{ip} \cdot n_{ib}^{(in)} \qquad i \in I^{(nf)}, p \in P \tag{16}$$

Constraints (2) compute the largest amount of storage space required in the INOLP. Constraints (3) (resp. (5)) compute the available inventory in the INOLP at the end (resp. at the beginning) of the day. Constraints (4) fix the initial inventory in the INOLP at the beginning of the planning horizon (i.e., the products that are not received during the week are assumed to be in inventory at the beginning of the week). Constraints (6) compute the amount of products received on each day at the INOLP. Constraints (7) compute the number of units of each product type sent on each day. Constraints (8) prevent a container from being loaded multiple times. Constraints (9) impose that products are sent on the loading day of a container. Constraints (10) limit the amount of products sent by containers. Constraints (11) impose that the demand of each client is satisfied. Constraints (12) and (13) (resp. (14) and (15)) define the loading constraints of the containers (resp. of the trucks). More precisely, constraints (12) (resp. (15)) ensure that the weight of the transported products does not exceed the container (resp. truck) capacity, and constraints (13) (resp. (14)) ensure that the number of boxes transported in a container (resp. truck) does not exceed the allowed limit. Last, constraints (16) forbid that a truck transports products delivered by non-visited suppliers.

3.3. Specific configurations of $Q(O^{(nf)}, I^{(nf)})$

The following specific configurations are introduced.

- Q: configuration where the full content of both containers and trucks is optimized (i.e., $O^{(nf)} = O$ and $I^{(nf)} = I$),
- Q_z: configuration where only the content of all the containers is optimized (i.e., the z_{ibp} variables are fixed: $O^{(nf)} = O$ and $I^{(nf)} = \emptyset$),
- Q_x: configuration where only the content of all the trucks is optimized (i.e., the x_{obp} variables are fixed: $O^{(nf)} = \emptyset$ and $I^{(nf)} = I$),
- $Q_z(O^{(nf)})$: configuration where all truck contents are fixed (i.e., $I^{(nf)} = \emptyset$) and the content of a subset of containers ($O^{(nf)}$) is optimized,
- $Q_x(I^{(nf)})$: configuration where all container contents are fixed (i.e., $O^{(nf)} = \emptyset$) and the content of a subset of trucks ($I^{(nf)}$) is optimized,
- $Q_{x,z}$: configuration where the decision making only focuses on the loading day of the containers (i.e., $O^{(nf)} = \emptyset$ and $I^{(nf)} = \emptyset$); it corresponds to current practice at ECM, according to which the content of the trucks and the containers is built in a pre-processing phase using two independent optimization tools, and $Q_{x,z}$ is then solved "by hand" (i.e., in a constructive fashion) by the decision maker.

Configurations Q_z and $Q_{x,z}$ have been considered in Coindreau et al. (2019). Furthermore, as considered in Coindreau et al. (2019), configuration $Q_{z,t}$ (resp. $Q_{x,z,t}$) stands for the decomposition of Q_z (resp. $Q_{x,z}$) that aims at maximizing the volume of products sent at the end of day t when the content of the trucks (resp. the content of both trucks and containers) is fixed. It has been shown in Coindreau et al. (2019) that $Q_{x,z,t}$ is equivalent to the multiple knapsack problem, which is known to be a NP-hard (Puchinger et al. 2010).

Table 1 summarizes the above configurations. For each configuration, the decision sets and the fixed variables are given. "×" indicates that the corresponding decision variables are taken into account. In the decompositions aimed at optimizing the volume shipped when fixing the loading day of the containers, the decisions concerning the loading operations of the containers are made on a subset

of containers. Accordingly, "(×)" means that the decision variables are partially taken into account. The first four configurations have been studied in Coindreau et al. (2019) and correspond to the situations where the content of the trucks is fixed.

Table 1: Considered configurations of the ECM problem.

Configuration	Fixed variables	Decision variables		
		Truck content	Container content	Loading day
Q_z	Truck content		×	×
$Q_{z,t}$	Truck content, loading day		×	(×)
$Q_{x,z}$	Truck and container content			×
$Q_{x,z,t}$	Truck and container content, loading day			(×)
Q	-	×	×	×
Q_x	Container content	×		×

4. Matheuristics

Since neither Q nor Q_z cannot be solved with CPLEX for the largest instances provided by ECM, we propose two matheuristics capable of handling large and complex cases. First, we introduce DM to solve configuration Q in Section 4.1. Section 4.2 details FOM, which aims at solving multiple times $Q(O^{(nf)}, I^{(nf)})$ with different selections of trucks and containers to be optimized. Section 4.3 proposes a matheuristic based on a combination of DM and FOM. Finally, Section 4.4 highlights additional advantages for ECM to favor FOM over alternative solution methods.

4.1. Decomposition matheuristic (DM)

As discussed by Archetti and Speranza (2014), the key idea behind a decomposition matheuristic is to divide the main problem into smaller subproblems that are easier to solve. Each subproblem is then solved by mathematical programming.

To solve Q, we propose to sequentially optimize the content of the containers and then the content of the trucks (i.e., solve Q_z and then Q_x). Coindreau et al. (2019) introduced a temporal decomposition matheuristic (called TDM) to solve Q_z. Each day, the container contents are reorganized so as to

maximize the sent volume of products that is shipped. In other words, the configuration $Q_{z,t}$ is solved from $t = 1$ to 5.

It turns out that Q_x is easier to solve than Q_z. Indeed, CPLEX is able to solve Q_x for all ECM instances within an hour. In contrast to Q_x, Q_z yields a much larger number of variables, since decisions can be made on both the container contents and their loading day. Whereas Q_z involves the x_{obp} variables for the container contents and the w_{opt} variables for the products sent on each day (the z_{ibp} variables being fixed), this configuration only considers the z_{ibp} variables for the truck contents (the x_{obp} variables being fixed and the w_{opt} variables being deduced from the values of x_{obp}).

The proposed DM for solving Q is straightforward. It first solves Q_z with TDM. It then solves Q_x with CPLEX (i.e., by optimizing the truck contents and taking as input the previously optimized container-loading schedule and contents).

4.2. Fix-and-optimize matheuristic (FOM)

The FOM aims at optimizing the content of both the trucks and the containers by successively considering different subsets of trucks and containers to be optimized. The pseudocode of FOM is given in Algorithm 1. At each step, $|I^{(nf)}|$ trucks and $|O^{(nf)}|$ containers are randomly selected. $Q(O^{(nf)}, I^{(nf)})$ is then solved with CPLEX, and the provided solution is taken as input for the next iteration (we propose to adaptively update the size of the $I^{(nf)}$ and $O^{(nf)}$ sets with Algorithm 2 below). Algorithm 1 takes as input an initial feasible solution s_0, e.g., the one currently used by ECM. It stops after η_{max} iterations without improvement or after t_{max} minutes of execution time (see the 'While" loop). (σ, t_{MILP}) are the MILP parameters used to solve $Q(O^{(nf)}, I^{(nf)})$ in Step 2. More precisely, the MILP stops when the gap to optimality is below $\sigma\%$ or after t_{MILP} minutes of execution time. An initial pair of percentages $(\rho_1^I < \rho_2^I)$ (resp. $(\rho_1^O < \rho_2^O)$) is also given as input for the proportion of trucks (resp. containers) to be optimized in $Q(O^{(nf)}, I^{(nf)})$. Such proportions are updated each η iterations of Algorithm 1 (see Step 4). We consider two different values to be able to determine, during the execution of Algorithm 1, whether smaller percentages (i.e., ρ_1^I and ρ_1^O) or larger percentages (i.e., ρ_2^I and ρ_2^O) should be favored for the next iterations (see Algorithm 2 below). To evaluate the gain associated with the percentage (ρ_i^I, ρ_j^O) selected in Step 1, Step 3 computes the achieved inventory penalty reduction Δ_{ij} after η iterations and the associated required execution time τ_{ij}. Preliminary experiments (not reported here) have indicated that the

tuning ($\eta = 12$, $\sigma = 2\%$, $t_{MILP} = 10$ minutes, $\rho_1^I = 15\%$, $\rho_2^I = 17\%$, $\rho_1^O = 10\%$, $\rho_2^O = 11\%$) is efficient.

The local search framework, of which FOM is an extension, usually relies on the complete exploration of the given neighborhood at each iteration. Here, setting $\sigma = 0\%$ (i.e., each subproblem is optimally solved and the neighborhood at each iteration is completely explored) significantly worsens the performances of the FOM (compared with the case where σ is non-null). Indeed, despite the fact that a small gain could be achieved at each iteration when solving optimally each subproblem, the computational effort required to prove the optimality of each subproblem significantly reduces the number of iterations that are performed by FOM (up to three times less iterations performed when $\sigma = 0\%$ compared with $\sigma = 2\%$). Ultimately, $\sigma = 2\%$ appears to be a good trade-off between the intensification and the diversification abilities of FOM.

In the same vein, to increase the efficiency of the intensification at each iteration, we compared the two following cases: (1) selecting containers and trucks randomly and (2) selecting with higher probability containers and trucks having a larger number of allowed permutations (e.g., the containers for which the set of products that could be transported is high). In the latter case, each iteration leads to a greater improvement of the objective function but at the cost of an extensive computational effort. As a result, the diversification ability of FOM is reduced, which ultimately affects the efficiency of the algorithm. Consequently, the random approach for selecting containers and trucks has been selected for the final experiments.

Algorithm 1 Fix-and-optimize matheuristic (FOM)

Input: s_0, (σ, t_{MILP}), (η_{max}, t_{max}), (ρ_1^I, ρ_2^I), (ρ_1^O, ρ_2^O), η.

Initialization: set $l = 1$; set $\Delta_{ij} = 0$ and $\tau_{ij} = 0$ $(\forall\, i, j \in \{1, 2\})$.

While (execution time $< t_{max}$) **or** (a solution improvement has been made in the last η iterations), **do:**

1. *Select the set of trucks and the set of containers to optimize:* choose randomly (i, j) (where $i, j \in \{1, 2\}$), and select randomly $|I^{(nf)}| = \lceil \rho_i^I \cdot |I| \rceil$ trucks and $|O^{(nf)}| = \lceil \rho_j^O \cdot |O| \rceil$ containers.
2. *Solve* $Q(O^{(nf)}, I^{(nf)})$ with CPLEX and let s_l be the resulting solution.
3. *Evaluate the performance of the selected* (ρ_i^I, ρ_j^O)*:* set $\Delta_{ij} = f(s_l) - f(s_{l-1})$ (where $f(s_l)$ is the inventory penalty of s_l) and add to τ_{ij} the execution time required to solve $Q(O^{(nf)}, I^{(nf)})$.
4. *Periodically update the truck/container percentages:* if $(l \mod \eta) = 0$, update (ρ_1^I, ρ_2^I) and (ρ_1^O, ρ_2^O) with Algorithm 2; re-initialize $\Delta_{ij} = 0$ and $\tau_{ij} = 0$, $\forall\, i, j \in \{1, 2\}$.
5. *Move to the next iteration:* set $l = l + 1$.

Return: s_l (i.e., the last generated solution).

Algorithm 2 aims at choosing the pairs of percentages that will be used for the next sequence of η iterations of Algorithm 1 (see its Step 4). It takes as input the values of the percentages (ρ_1^I, ρ_2^I) and (ρ_1^O, ρ_2^O) used during the previous η iterations, as well as their associated inventory penalty reductions (Δ_{ij}) and execution times (τ_{ij}).

For each couple (ρ_i^I, ρ_j^O) (where $i, j \in \{1, 2\}$), Algorithm 2 first computes the improvement score $\theta_{ij} = \frac{\Delta_{ij}}{\tau_{ij}}$ (in m^3/minute) provided by $Q(O^{(nf)}, I^{(nf)})$ (with $I^{(nf)} = \lceil \rho_i^I \cdot |I| \rceil$ and $O^{(nf)} = \lceil \rho_j^O \cdot |O| \rceil$) during the last sequence of η iterations of Algorithm 1. If no percentage configuration has improved the solution (i.e., if $\theta_{ij} = 0\ \forall\, i, j \in \{1, 2\}$, case 1), it can either be due to the fact that the percentages (ρ_2^I, ρ_2^O) are too small (hence the solution space explored in $Q(O^{(nf)}, I^{(nf)})$ is too narrow), or (ρ_1^I, ρ_1^O) are too large (hence CPLEX cannot explore the solution space of $Q(O^{(nf)}, I^{(nf)})$ within t_{MILP} minutes to find a better solution than the current one). Therefore, in that case, we move the smaller percentage ρ_1^I (resp. ρ_1^O) of trucks (resp. containers) to an even smaller value, and the larger percentage ρ_2^I (resp. ρ_2^O) to an even larger value.

When at least one percentage pair has allowed the MILP to improve the input solution, let $(i^\star, j^\star) = \arg \max_{(i,j) \in \{1,2\}^2} \theta_{ij}$ (break ties randomly). In case 2 (resp. case 4), corresponding to $i^\star = 1$ (resp. $j^\star = 1$), as the smaller percentage ρ_1^I (resp. ρ_1^O) of trucks (resp. containers) to be optimized has yielded higher improvement score, we move the two percentages of trucks (resp. containers) to even

smaller values. Conversely, in case 3 (resp. case 5), corresponding to $i^\star = 2$ (resp. $j^\star = 2$), the larger percentage ρ_2^I (resp. ρ_2^O) of trucks (resp. containers) to be optimized has yielded higher improvement score, and we thus move the two percentages of trucks (resp. containers) to even larger values.

Algorithm 2 Update of the percentages of trucks and containers to be optimized in $Q(O^{(nf)}, I^{(nf)})$

Input: (ρ_i^I, ρ_j^O), Δ_{ij}, τ_{ij}, $\forall\, i, j \in \{1, 2\}$.

Initialization:

- Set $\delta^I = \rho_2^I - \rho_1^I$ and $\delta^O = \rho_2^O - \rho_1^O$.
- *Compute the improvement score for* (ρ_i^I, ρ_j^O)*:* set $\theta_{ij} = \frac{\Delta_{ij}}{\tau_{ij}}$, $\forall\, i, j \in \{1, 2\}$.

If $\theta_{ij} = 0\ \forall\, i, j \in \{1, 2\}$ (case 1), set: $\rho_1^I = \rho_1^I - \delta^I$; $\rho_2^I = \rho_2^I + \delta^I$; $\rho_1^O = \rho_1^O - \delta^O$; $\rho_2^O = \rho_2^O + \delta^O$.

Else Determine $(i^\star, j^\star) = \arg \max_{(i,j) \in \{1,2\}^2} \theta_{ij}$ (break ties randomly).

If $i^\star = 1$ (case 2), set $\rho_2^I = \rho_1^I$ and $\rho_1^I = \rho_1^I - \delta^I$;

If $i^\star = 2$ (case 3), set $\rho_1^I = \rho_2^I$ and $\rho_2^I = \rho_1^I + \delta^I$;

If $j^\star = 1$ (case 4), set $\rho_2^O = \rho_1^O$ and $\rho_1^O = \rho_1^O - \delta^O$;

If $j^\star = 2$ (case 5), set $\rho_1^O = \rho_2^O$ and $\rho_2^O = \rho_1^O + \delta^O$.

Return: $(\rho_i^I, \rho_j^O), \forall\, i, j \in \{1, 2\}$.

4.3. Combined matheuristic (DM-FOM)

When the available execution time is larger than the run time of DM, we propose the following combined matheuristic, referred to as DM-FOM. In a first phase, we launch DM. In a second phase, we use the remaining available execution time to run FOM, taking the DM solution as an input and further improving it. Whereas DM-FOM aims at solving configuration Q (i.e., both the truck and container contents are optimized), TDM-FOM combines TDM and FOM in the same fashion to solve configuration Q_z (where $|I^{nf}| = 0$, i.e., the truck contents are fixed).

4.4. Facilitated implementation of FOM

Additional advantages of FOMs were highlighted by Papageorgiou et al. (2018). In the context of the ECM problem, FOM stands out from other matheuristics, and more generally from metaheuristics,

by the fact that sustainability and simplified maintenance of the code is ensured by ECM, implying that it is easier for the optimization team to manage one single MILP that relies on a general purpose solver rather than a low level code that requires high maintenance. Furthermore, FOM is able to handily adapt to new business settings since, it requires less effort to update one single MILP rather than customized algorithms.

5. Computational experiments

The models were coded in C++ and CPLEX 12.10 was called to solve the induced MILPs. Computations were launched on a 2.2 GHz Intel Core i7 with 16 Go 1600 MHz DDR3 of RAM memory. The ECM problem is solved once a week. In accordance with ECM, it is therefore reasonable to consider an overall execution time of 10 hours. However, for most of the experiments presented below, an execution time of one hour was sufficient to obtain the presented solutions. In the following, the percentage gap of the inventory penalty f_1 with respect to f_2 is computed as $100 \cdot \frac{f_1 - f_2}{f_2}$.

Section 5.1 describes the set of instances provided by ECM. We compare the solution methods in Section 5.2; results for configurations Q_z are given in Section 5.2.1 and results for Q in Sections 5.2.2 and 5.2.3. Configuration Q_x is not treated here since it can be solved directly with CPLEX.

5.1. Test instances

Table 2 gives the characteristics of the 24 instances under study. The 17 first instances were provided by ECM and are derived from three different INOLPs: V, G and M. The N instances were generated to assess the stability of the matheuristics on more complex situations. The first column indicates the name of the instances, columns 2 to 7 indicate the size of the sets introduced in Section 3.1. "V^{sent}" and "V^{init}" give the volume of the sent boxes and the volume of the boxes located in the inventory at the beginning of the week, respectively. The last two columns describe the size of configuration Q for each instance: "Nb. Var." (resp. "Nb. Const.") gives the number of variables (resp. constraints). We have removed from the model the variables that can only take a single value (e.g., for instance M7, there are $|O| \times |B| \times |P| \approx 10^{10}$ x_{obp} variables, but after variable elimination, configuration Q involves less than six millions variables (see Coindreau et al. (2019) for more details on this variable elimination procedure).

Each instance Ni ($i \in \{1, \ldots, 7\}$) has the same characteristics as its corresponding instance Mi, except for the number of suppliers (which has been decreased by 30%) and for the number of different boxes (which has been decreased by 50%). More precisely, on the one hand, 30% of the suppliers were randomly replaced by another one in the remaining 70%. On the other hand, the allowed weight variation for the boxes was increased to 50 kg instead of 10 kg (meaning that a box can transport different products until the weight variation does not increase 50 kg). These newly generated instances are used to evaluate the developed algorithms on more complex situations and to evaluate the effect of these two parameters on the management of the INOLP. Indeed, decreasing the number of suppliers and the number of boxes increase the complexity as in both cases, the number of possible permutations between trucks and between containers become larger. Compared with the M instances, the number of variables in the N instances is on average 18% higher (see column "Nb. Var.") and the number of constraints is on average 19% larger (see column "Nb. Const.").

5.2. Analysis of the performance of the proposed solution methods

We now proceed to the analysis of our matheuristics on the configurations Q_z and Q.

5.2.1. Results on configuration Q_z

In this section, we focus on configuration Q_z involving only the decisions on the contents and on the loading days of the containers. We benchmark FOM and TDM-FOM (both with $|I^{nf}| = 0$, as the truck contents are fixed) with respect to TDM. For Q_z, Coindreau et al. (2019) showed that TDM is able to find optimal solutions on the smaller instances V and G. For the larger instances M (n.b., the N instances were not considered in this seminal paper), TDM is able to find solutions exhibiting a significant gain compared with the ECM current practice. We do not report the results for the V and G instances as they are optimally solved with CPLEX and, starting from the ECM solutions, FOM is able to find optimal solutions within five minutes. Recall that FOM and TDM-FOM are limited to 10 hours of execution time.

Table 3 compares the results of TDM, FOM and TDM-FOM for the M and N instances. Columns "Obj." give the value of the objective function. "Time" indicates the time (in minutes) at which TDM returned its solution. "% best" provides the percentage gap with respect the best found

Table 2: Characteristics of the test instances.

Instance	$\|O\|$	$\|I\|$	$\|P\|$	$\|B\|$	$\|S\|$	$\|C\|$	V^{sent}	V^{init}	Nb. Var.	Nb. Const.
V1	28	48	326	206	151	17	1,681	634	12,031	14,107
V2	51	78	358	290	171	20	3,285	1,154	14,054	17,234
V3	49	67	424	315	190	21	2,915	898	16,554	20,492
V4	59	82	454	334	191	20	3,920	1,411	19,022	23,700
G1	67	98	1,181	616	544	8	4,462	1,098	119,238	166,937
G2	71	112	1,199	644	554	7	4,921	942	132,638	182,749
G3	68	89	1,353	575	572	8	5,131	1,341	161,969	218,955
G4	88	112	1,401	718	606	8	5,979	1,503	162,171	231,079
G5	80	122	1,548	605	646	8	6,110	18,92	243,688	315,456
G6	85	136	1,676	748	678	7	6,442	917	244,916	330,369
M1	383	677	6,564	999	542	17	78,399	15,392	3,283,199	4,382,861
M2	543	653	7,890	1,262	626	23	89,937	14,995	3,808,742	5,403,274
M3	644	903	7,865	1,226	568	22	105,082	14,933	4,372,938	6,234,108
M4	699	778	7,529	1,167	597	23	109,501	23,458	4,062,400	5,641,036
M5	623	741	8,349	1,159	608	23	106,073	14,211	4,923,679	6,754,536
M6	789	1,085	8,546	1,377	590	21	130,476	23,828	5,272,599	7,626,068
M7	829	1,104	8,649	1,387	597	22	142,679	26,115	5,883,213	8,577,182
N1	383	677	6,564	505	310	17	78,399	15,392	3,732,772	5,057,496
N2	543	653	7,890	682	344	23	89,937	14,995	4,592,790	6,542,114
N3	644	903	7,865	658	326	22	105,082	14,933	5,152,584	7,341,628
N4	699	778	7,529	622	338	23	109,501	23,458	4,796,172	6,764,057
N5	623	741	8,349	606	341	23	106,073	14,211	5,757,389	8,055,724
N6	789	1,085	8,546	764	333	21	130,476	23,828	6,288,690	9,073,930
N7	829	1104	8,649	773	337	22	142679	26,115	6,995,642	10,167,524

inventory penalty, which is always returned by DM-FOM. The columns "% $O^{(nf)}$" give the average percentage of containers optimized at each iteration of FOM.

On the one hand, one can observe that the TDM solutions can be further improved by FOM during the remaining available execution time. Indeed, FOM alone is able to outperform TDM on four out of the 14 instances. On the other hand, TDM-FOM is able to improve the results of TDM by 1.5% on average. Such improvements could not be achieved without the use of FOM, as different runs of TDM always return the same solution. Finally, the "% $O^{(nf)}$" values show that all instances do not require the same average percentage of containers to be optimized at each iteration of FOM. For example, large values are not appropriate for N7 because of the complexity resulting from its size (on average 11.8% of the containers are optimized for FOM). In contrast, for the smaller instances M3 and M5, much larger percentages of the containers are optimized (more than 20% for FOM). These results highlight the importance of dynamically updating, during the execution of FOM, the values of these percentages (see Algorithm 2).

Table 3: Results of Q_z for TDM, FOM and TDM-FOM (M and N instances).

	TDM			FOM			TDM-FOM	
Instance	Obj.	Time[min]	% best	Obj.	% $O^{(nf)}$	% best	Obj.	% $O^{(nf)}$
M1	44,078	13	1.7%	43,326	18.1%	0.0%	43,474	15.5%
M2	46,435	34	0.9%	46,641	15.4%	1.3%	46,023	15.2%
M3	53,417	32	0.1%	53,371	23.0%	0.0%	53,417	11.7%
M4	48,581	28	0.0%	52,920	15.9%	8.9%	48,578	11.5%
M5	51,503	29	0.0%	51,517	22.3%	0.0%	51,503	11.7%
M6	56,889	71	2.0%	56,935	18.6%	2.1%	55,774	18.5%
M7	60,274	47	3.8%	59,069	15.4%	1.8%	58,052	15.5%
N1	38,495	17	4.3%	37,961	14.6%	2.9%	36,894	14.1%
N2	35,360	33	1.2%	36,521	15.9%	4.5%	34,939	14.2%
N3	37,329	34	0.2%	39,751	16.9%	6.7%	37,249	12.0%
N4	34,073	37	2.0%	40,103	11.9%	20.0%	33,409	13.4%
N5	39,448	38	0.5%	42,299	15.9%	7.8%	39,244	14.6%
N6	40,733	60	0.5%	46,735	13.8%	15.3%	40,526	11.2%
N7	34,520	100	4.9%	51,012	11.8%	55.1%	32,899	13.3%

5.2.2. Results of CPLEX, DM and FOM on small instances

For the V and G instances, Table 4 compares the results of DM and FOM with the optimal solutions proven by CPLEX in the eponymous columns. The columns "Obj." and "Time" are defined as above (but the time is given in seconds). For DM and FOM, the column "% opt." provides the gap with

respect to the optimal solution. The columns "% $O^{(nf)}$" and "% $I^{(nf)}$" give the average percentage of containers and trucks optimized at each iteration of FOM, respectively (this will be commented later). We observe that DM is faster than FOM, but the latter heuristic yields better solutions. Indeed, for FOM, the average gap to optimality never exceeds 2% for each instance. Interestingly, FOM requires on average 32% less execution time than CPLEX. The results of DM-FOM are not reported for the V and G instances. Indeed, FOM already shows a good performance for these smaller instances, and neither the objective nor the execution time are significantly improved by DM-FOM.

Table 4: Results of Q for CPLEX, DM and FOM (V and G instances).

	CPLEX		DM			FOM				
Instance	Obj.	Time[s]	Obj.	Time[s]	% opt.	Obj.	Time[s]	% $O^{(nf)}$	% $I^{(nf)}$	% opt.
V1	691	1	694	<1	0%	692	4	24.1%	34.4%	0%
V2	1,229	24	1,229	1	0%	1,229	1	8.3%	11.7%	0%
V3	1,481	1	1,481	1	0%	1,481	8	24.5%	37.1%	0%
V4	1,727	22	1,727	1	0%	1,727	9	22.4%	31.0%	0%
G1	2,489	15	2,643	11	6%	2,536	86	26.3%	39.9%	2%
G2	2,421	156	2,590	12	7%	2,450	129	26.4%	40.2%	1%
G3	2,621	56	2,632	28	0%	2,624	79	24.7%	31.1%	0%
G4	3,339	249	3,406	17	2%	3,346	85	23.0%	32.0%	0%
G5	2,651	2,174	3,105	21	17%	2,705	338	26.1%	40.9%	2%
G6	3,246	500	3,404	25	5%	3,326	198	22.3%	35.0%	2%

5.2.3. Results of DM, FOM, and DM-FOM on large instances

Table 5 compares the results of DM, FOM and DM-FOM for the M and N instances. For DM, we report the execution time (in minutes) in the column "Time". The time is not reported for FOM and DM-FOM since for these instances, the full 10-hour time budget is used. The column "% best" provides the gap with respect to the solution value found by DM-FOM (which is always the best solution). Finally, for both FOM and DM-FOM, we report the average percentage of trucks and containers optimized at each iteration in columns "% $O^{(nf)}$" and "% $I^{(nf)}$", respectively.

Table 5 highlights that DM-FOM allows to efficiently use the available 10 hours of execution time and outperforms both DM and FOM. The average gap between DM and DM-FOM (resp. between FOM and DM-FOM) is 17.7% (resp. 16.0%). DM turns out to be a powerful first phase for the ECM problem: it demonstrates the importance of considering the problem characteristics to find appropriate decomposition techniques. Here, maximizing the volume sent daily is particularly

efficient and is one of the strengths of DM. DM is able to identify good solutions (within 21 minutes of execution time for the smallest instance M1, and 345 minutes for the most complex instance N7) and is therefore recommended as a warm start for FOM, as opposed to initially feeding FOM with the ECM solution. The larger the instance, the less efficient is FOM and the more important it becomes to use DM as a first phase for FOM: for instance M1, FOM alone improves by 26.9% the results of DM whereas for N6, FOM alone improves DM results by 5.3%. Note that FOM does not improve the DM results for N7, this is due to the fact for this instance DM provides a solution for which the largest inventory the INOLP is close to the inventory observed at the beginning of the week, hence few improvements could be made on that solution. It is interesting to note that considering simultaneously the optimization of the truck and of the container contents is necessary in order to be able to further improve the results returned by DM.

Additional experiments (not reported here) indicate that letting 10 hours of execution for DM only (i.e., iteratively solving Q_z and Q_x for 10 hours) does not improve the solution found after one single iteration of DM (i.e., solve Q_z then Q_x once). Indeed, decomposing the resolution with Q_z followed by Q_x is efficient to quickly find a rather good solution, but cannot, in contrast to FOM, further improve it. In this case, as the truck (or container) contents are always optimized to be suitable to the container (or truck) contents given as input, reoptimizing always yields similar truck and container contents.

The average gap between DM and DM-FOM is larger when solving Q than when solving Q_z (on average, it moves from 1.5% for Q_z to 17.7% for Q). This indicates that DM is more efficient on configuration Q_z than on Q. Hence, when integrating the decisions on both the truck and container contents with those on the container scheduling (i.e., configuration Q), FOM becomes an essential tool. The average gap between FOM and DM-FOM is equal to 15.9%, highlighting again the importance of considering the solution of DM as a warm start for FOM.

Tables 4 and 5 show how the values of the percentages of trucks and containers to be optimized at each iteration of FOM adapt to the characteristics of the instances. Typically, the percentage of optimized trucks is larger than the percentage of optimized containers. This stems from the increased complexity of Q_z compared with Q_x. Furthermore, we observe that the larger is the instance, the smaller is the percentage of trucks or containers to be optimized at each iteration of FOM. The strength of FOM lies more in the number of performed iterations within the allowed time rather than on the magnitude of the improvement achieved at each iteration.

Introducing instances N allows us to test our algorithms on more complex cases (as highlighted in Section 5.1, those instances exhibit larger solution spaces). We first notice that it takes twice as long for DM to solve the N instances compared to the M instances. As for the M instances, FOM is able to further improve the results provided by DM on the N instances. On average, FOM improves by 22.1% the results of DM for the M instances and by 15.4% for the N instances. This difference in the magnitude of improvement can be explained by the fact that less time budget is left to FOM to work on the solutions provided by DM. Moreover, as the problem to be solved is more complex, a larger amount of time is required to solve each subproblem and hence less iterations are performed by FOM. Nevertheless, FOM and DM-FOM are able to efficiently solve the N instances, as shown in Sections 5.2.3 and 6.3.

Table 5: Results of Q for DM, FOM and DM-FOM (M and N instances).

	DM			FOM				DM-FOM		
Instance	Obj.	Time[min]	% best	Obj.	% best	% $O^{(nf)}$	% $I^{(nf)}$	Obj.	% $O^{(nf)}$	% $I^{(nf)}$
M1	25,046	21	26.9%	19,741	0.0%	17.2%	26.9%	19,805	15.6%	22.1%
M2	36,653	44	26.3%	29,015	0.0%	16.5%	23.8%	30,241	15.6%	19.1%
M3	35,804	73	26.7%	28,265	0.0%	15.4%	20.4%	28,296	15.6%	19.1%
M4	37,295	54	26.5%	32,660	10.7%	16.2%	23.7%	29,490	15.9%	16.7%
M5	34,040	55	25.6%	27,109	0.0%	16.2%	26.5%	27,204	16.1%	21.3%
M6	41,517	125	10.7%	39,918	6.5%	13.4%	17.5%	37,499	11.6%	19.5%
M7	42,830	151	12.0%	44,595	16.6%	11.5%	18.2%	38,242	12.7%	18.0%
N1	18,380	22	17.3%	15,665	0.0%	16.0%	24.4%	15,704	15.7%	19.6%
N2	26,699	40	25.7%	23,462	10.5%	14.6%	23.4%	21,236	13.1%	21.6%
N3	22,443	118	16.9%	21,251	10.7%	14.2%	20.5%	19,194	14.4%	22.1%
N4	26,876	50	14.4%	26,997	14.9%	13.1%	19.0%	23,501	14.8%	24.1%
N5	26,154	55	28.2%	24,759	21.4%	14.1%	22.1%	20,398	12.9%	17.7%
N6	27,692	393	5.3%	35,018	33.1%	11.0%	17.2%	26,309	12.8%	18.9%
N7	26,376	345	0.0%	53,091	101.3%	10.6%	16.3%	26,376	11.5%	18.9%

6. Managerial insights

We now evaluate the potential gain in terms of inventory penalty offered to ECM when simultaneously considering both the truck and container contents in the optimization, together with the loading day of the containers. In particular, we compare the results with those obtained in situations where only the decisions on the content of the trucks or the containers, or neither (which sets for the current practice at ECM), are integrated with those on the scheduling of container loading operations. In Section 6.1, we compare the returned optimal solutions for the V and G instances.

Next, in Section 6.2, we compare the best found solutions for the M instances. Section 6.3 analyzes the impact of a reduced number of suppliers and of a smaller box sets by comparing the M and N instances. Section 6.4 summarizes the improvement potential achieved by our integrated approach.

6.1. V and G instances

Table 6 compares the obtained solutions for configuration Q (i.e., both the contents of the trucks and the containers are optimized) with those achieved when (1) the content of the containers is fixed (column "Q_x"); (2) the content of the trucks is fixed (column "Q_z"); (3) both the content of the trucks and that of the containers are fixed (column "ECM" which corresponds to the configuration $Q_{x,z}$). The columns "Obj." report the value of the optimal solution and the columns "Time" give the time (in minutes) at which CPLEX returned the optimal solution. Columns "% (ECM)", "% Q_z" and "%(Q_x)" give the improvement percentage with respect to configuration ECM, Q_z and Q_x, respectively. For example, the improvement achieved by configuration Q over configuration Q_z is displayed in column "% Q_z" and is computed as $\frac{f(Q)-f(Q_z)}{f(Q_z)}$, where $f(Q_z)$ (resp. $f(Q)$) designates the obtained inventory penalty when considering configuration Q_z (resp. Q).

As already discussed in Coindreau et al. (2019), reconsidering the content of the containers during the optimization of the container loading operations allows to significantly improve the solution currently used at ECM (with an average improvement of 5% for the V instances and of 15% for the G instances). The gain achieved is of the same magnitude when integrating only the decisions on the content of the trucks in the optimization (with an average improvement of 17% for the V instances and of 14% for the G instances). The main improvement is achieved when we consider simultaneously the content of the trucks and containers together with the loading day of the containers. Compared with the results obtained in Coindreau et al. (2019), the additional average improvement brought by solving Q instead of Q_z amounts to 16% for the V instances, and to 19% for the G instances. Compared with the current practice at ECM, the average improvement achieved by optimizing both on the truck and container contents is 20% for the V instances, and 31% for the G instances. For these V and G instances, we recall that the largest execution time to find the optimal solutions with CPLEX is 31 minutes.

Table 6: Results of Q for the V and G instances.

	ECM		Q_z			Q_x			Q				
Instance	Obj.	Time[min]	Obj.	Time[min]	% (ECM)	Obj.	Time[min]	% (ECM)	Obj.	Time[min]	% (ECM)	% Q_z	% Q_x
V1	1,158	< 1	1,158	< 1	0%	695	< 1	-40%	691	1	-40%	-40%	-1%
V2	1,454	< 1	1,377	< 1	-5%	1,307	< 1	-10%	1,229	< 1	-15%	-11%	-6%
V3	1,710	< 1	1,631	1	-5%	1,484	< 1	-13%	1,481	1	-13%	-9%	0%
V4	2,117	< 1	1,956	< 1	-8%	1,888	< 1	-11%	1,727	< 1	-18%	-12%	-9%
G1	3,292	< 1	2,801	1	-15%	3,144	1	-4%	2,489	2	-24%	-11%	-21%
G2	3,690	< 1	3,074	3	-17%	3,130	1	-15%	2,421	5	-34%	-21%	-23%
G3	3,517	< 1	2,943	1	-16%	3,141	1	-11%	2,621	1	-25%	-11%	-17%
G4	4,318	< 1	3,672	3	-15%	4,005	2	-7%	3,339	4	-23%	-9%	-17%
G5	4,843	< 1	4,172	5	-14%	3,450	4	-29%	2,651	31	-45%	-36%	-23%
G6	4,706	< 1	4,059	4	-14%	4,037	3	-14%	3,246	12	-31%	-20%	-20%

6.2. M instances

Table 7 presents, for the M instances, the best solutions obtained (computation details are provided in Section 5) for the configurations depicted in Section 6.1. The columns of Table 7 correspond to those of Table 6. When the time is not reported, this means that the algorithm used the entire allowed 10 hours of execution time to obtain the achieved inventory penalty. The results for the N instances are not reported here as this section aims at comparing our solutions with those currently obtained by ECM. For these instances, Section 6.3 shows how the solutions are impacted by a reduction of the number of suppliers and of different box types.

For these larger instances, and similarly to the smaller instances, Table 7 shows that optimizing only the truck or the container contents leads to similar average improvement when compared to the ECM current practice. The average improvement of Q_z (resp. Q_x) over ECM is 34% (resp. 29%). The main improvement comes when considering all the decisions simultaneously (both the truck, the container contents, and the container loading day). The average inventory-penalty reduction when solving Q instead of Q_z (as done in Coindreau et al. (2019)) is 37%. Compared with the ECM current practice, the gain is up to 72%, with an average of 58%. Such observations confirm the importance of integrating decisions on both the inbound and outbound sides at ECM's INOLPs.

6.3. Impact of a reduced number of suppliers and box types

Table 8 compares the results for the M and N instances (we recall that instance Ni, $i \in \{1, \ldots, 7\}$, has the same characteristics as instance Mi except for the number of suppliers and of box types,

Table 7: Results of Q for the M instances.

	ECM		Q_z		Q_x			Q			
Instance	Obj.	Time[min]	Obj.	% (ECM)	Obj.	Time[min]	% (ECM)	Obj.	% (ECM)	% Q_z	% Q_x
M1	73,301	< 1	43,474	-40.7%	46,438	9	-36.6%	19,805	-73.0%	-54.4%	-57.4%
M2	56,530	< 1	46,023	-18.6%	44,221	7	-21.8%	30,241	-46.5%	-34.3%	-31.6%
M3	77,077	< 1	53,417	-30.7%	51,864	15	-32.7%	28,296	-63.3%	-47.0%	-45.4%
M4	71,518	< 1	48,578	-32.1%	52,972	8	-25.9%	29,490	-58.8%	-39.3%	-44.3%
M5	73,436	< 1	51,503	-29.9%	51,362	10	-30.1%	27,204	-63.0%	-47.2%	-47.0%
M6	93,518	< 1	55,774	-40.4%	72,754	28	-22.2%	37,499	-59.9%	-32.8%	-48.5%
M7	100,504	< 1	58,052	-42.2%	67,084	57	-33.3%	38,242	-61.9%	-34.1%	-43.0%

that have been decreased respectively by 30% and by 50%). Columns "M" (resp. "N") provide the value of the objective function for the M (resp. N) instance, each line corresponding to one single instance ID. Columns "Impr. N" give the improvement achieved when considering the N instance compared with the respective M instance.

The N instances allow to draw the attention on the potential advantage offered by reducing the number of suppliers and the number of box types. Over all formulations (Q_z, Q_x, and Q), decreasing these two numbers yields an average improvement of 30%. While these results have to be mitigated by the fact that these reductions are hypothetical and could potentially not be implemented in practice, they highlight the interest of working on the standardization and reduction of box types.

Table 8: Impact of the reduction on the number of suppliers and on the number of box types.

	Q_z			Q_x			Q		
Inst. ID	M	N	Impr. N	M	N	Impr. N	M	N	Impr. N
1	43,474	36,894	17.8%	46,438	37,136	25.0%	19,805	15,704	26.1%
2	46,023	34,939	31.7%	44,221	41,852	5.7%	30,241	21,236	42.4%
3	53,371	37,249	43.3%	51,864	42,705	21.4%	28,296	19,194	47.4%
4	48,578	33,409	45.4%	52,972	48,508	9.2%	29,490	23,501	25.5%
5	51,503	39,244	31.2%	51,362	44,508	15.4%	27,204	20,398	33.4%
6	55,774	40,526	37.6%	72,754	64,723	12.4%	37,499	26,309	42.5%
7	58,052	32,899	76.5%	67,084	61668	8.7%	38,242	26,376	45.0%

6.4. Improvement potential

For each instance, any achieved inventory penalty at the INOLP lies between the inventory penalty of the ECM solution and the inventory penalty observed at the beginning of the week (see Table 2 for the values V^{init} of the volume of products stored at the beginning of the week). For instance

M1, the initial inventory volume V^{init} stored at the beginning of the week is 15,392 m^3 and the largest inventory volume $f(Q_{x,z})$ observed in the current ECM solution is 73,301 m^3. Any improving solution lies within these two bounds and the maximum theoretical improvement potential for this instance is 57,909 m^3. Considering Q_z allows a reduction of the largest storage volume to 43,783 m^3, and the savings for ECM is 29,518 m^3, which represents 51% of the maximum theoretical improvement potential (when only direct product transfers would take place).

Figure 2 displays, for all instances and for both configurations Q_z and Q, the achieved percentage of the maximum theoretical improvement potential. Each bar represents an instance, and the bold bar indicates the average for the V, G, and M instances. For each INOLP, Figure 2 highlights the significant additional gain achieved when considering Q over Q_z. It furthermore indicates that, on average and for both configurations Q_z and Q, the larger is the instance, the larger is the achieved improvement in terms of inventory penalty. This shows that our solution methods can take advantage of the increased potential for product exchange between trucks and containers in larger instances.

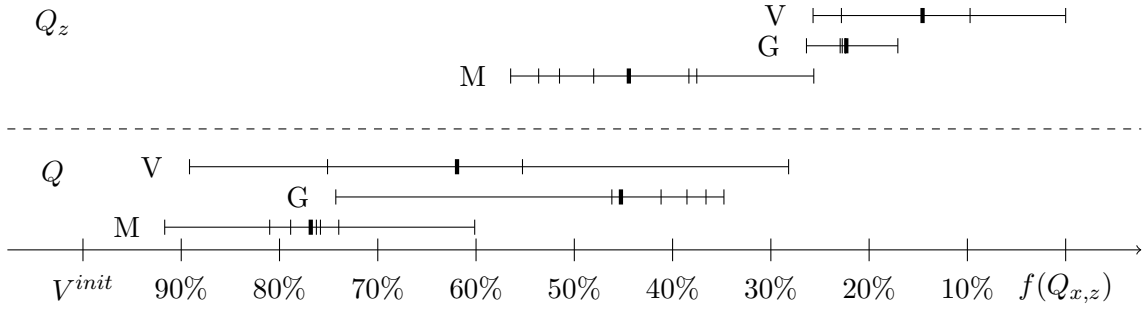

Figure 2: Percentage of the maximum theoretical improvement potential achieved by configurations Q_z and Q.

7. Conclusions

We have modeled and solved an industrial problem that considers the scheduling and the product assignment for both the inbound and outbound flows in a cross-docking platform. Whereas Coindreau et al. (2019) integrated the decisions on the outbound container content with the scheduling of their operations over the week, here we additionally included the decisions on the inbound truck contents in the same optimization framework. Concerning the complex loading constraints that affect both trucks and containers, we proposed an efficient formulation capable of quickly capturing the feasibility of different contents, as well as to evaluate their quality. We were able to realize the

significant improvement potential offered by the proposed integrated optimization framework for all instances provided by the involved company ECM.

We have developed and compared two heuristics, namely a decomposition matheuristic (DM) and a fix-and-optimize matheuristic (FOM). DM is faster but less efficient than FOM. The results are better when both methods are combined. Computational experiments showed that, compared with current industrial practice, allowing product reassignment from one container to another and from a truck to another, can reduce the average required largest inventory volume by 58% for the large instances and by 27% for the small ones. Moreover, compared with the situation where only the content of the containers is included in the decision making, the integrated formulation allows an additional average reduction of 37% for the large instances, and of 18% for the small ones. From a managerial point of view, revoking the content of the trucks may be a more challenging task than acting on those of the containers, as it involves third parties. However, this study clearly shows that implementing such aspects has the potential of yielding a significant improvement with respect to current practice and should therefore be considered.

Acknowledgement

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 2015-06189. This support is gratefully acknowledged. Thanks are due to the reviewers for their valuable comments.

References

References

Archetti, C. and Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2(4):223–246.

Bellanger, A., Hanafi, S., and Wilbaut, C. (2013). Three-stage hybrid-flowshop model for cross-docking. *Computers & Operations Research*, 40:1109–1121.

Boctor, F. F., Laporte, G., and Renaud, J. (2003). Heuristics for the traveling purchaser problem. *Computers & Operations Research*, 30(4):491–504.

Bortfeldt, A. and Wäscher, G. (2013). Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20.

Boysen, N. (2010). Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research*, 37(1):32–41.

Boysen, N. and Fliedner, M. (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega*, 38(6):413–422.

Buijs, P., Vis, I. F., and Carlo, H. J. (2014). Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, 239(3):593–608.

Chen, F. and Lee, C.-Y. (2009). Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research*, 193(1):59–72.

Chen, H. (2015). Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega*, 56:25–36.

Coindreau, M.-A., Gallay, O., Zufferey, N., and Laporte, G. (2019). Integrating workload smoothing and inventory reduction in three intermodal cross-docking platforms of a European car manufacturer. *Computers & Operations Research*, 112:104762.

Della Croce, F. and Salassa, F. (2014). A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, 218(1):185–199.

Dorneles, Á. P., de Araújo, O. C., and Buriol, L. S. (2014). A fix-and-optimize heuristic for the high school timetabling problem. *Computers & Operations Research*, 52:29–38.

Enderer, F., Contardo, C., and Contreras, I. (2017). Integrating dock-door assignment and vehicle routing with cross-docking. *Computers & Operations Research*, 88:30–43.

Gintner, V., Kliewer, N., and Suhl, L. (2005). Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, 27(4):507–523.

Helber, S. and Sahling, F. (2010). A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics*, 123(2):247–256.

Jourdan, L., Basseur, M., and Talbi, E.-G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620–629.

Papageorgiou, D. J., Cheon, M.-S., Harwood, S., Trespalacios, F., and Nemhauser, G. L. (2018). Recent progress using matheuristics for strategic maritime inventory routing. In *Konstantopoulos, C., Pantziou, G. (eds) Modeling, Computing and Data Handling Methodologies for Maritime Transportation*, volume 131, pages 59–94. Intelligent Systems Reference Library, Springer, Cham.

Puchinger, J., Raidl, G. R., and Pferschy, U. (2010). The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing*, 22(2):250–265.

Rieck, J., Ehrenberg, C., and Zimmermann, J. (2014). Many-to-many location-routing with inter-hub transport and multi-commodity pickup-and-delivery. *European Journal of Operational Research*, 236(3):863–878.

Rijal, A., Bijvank, M., and de Koster, R. (2019). Integrated scheduling and assignment of trucks at unit-load cross-dock terminals with mixed service mode dock doors. *European Journal of Operational Research*, 278(3):752–771.

Sahling, F., Buschkühl, L., Tempelmeier, H., and Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research*, 36(9):2546–2553.

Serrano, C., Delorme, X., and Dolgui, A. (2017). Scheduling of truck arrivals, truck departures and shop-floor operation in a cross-dock platform, based on trucks loading plans. *International Journal of Production Economics*, 194:102–112.

Tadumadze, G., Boysen, N., Emde, S., and Weidinger, F. (2019). Integrated truck and workforce scheduling to accelerate the unloading of trucks. *European Journal of Operational Research*, 278(1):343–362.

Toffolo, T. A., Esprit, E., Wauters, T., and Vanden Berghe, G. (2017). A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. *European Journal of Operational Research*, 257(2):526–538.

Van Belle, J., Valckenaers, P., and Cattrysse, D. (2012). Cross-docking: State of the art. *Omega*, 40(6):827–846.

Ye, Y., Li, J., Li, K., and Fu, H. (2018). Cross-docking truck scheduling with product unloading/loading constraints based on an improved particle swarm optimisation algorithm. *International Journal of Production Research*, 56(16):5365–5385.

Yu, W. and Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, 184(1):377–396.