Integrating workload smoothing and inventory reduction in three intermodal logistics platforms of a European car manufacturer

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

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

[b] *Geneva School of Economics and Management, GSEM, 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*

Abstract

We consider the optimization of container loading at three intermodal logistics platforms (ILP) of a large European car manufacturer (ECM). The decisions focus both on the loading day of each container and on its filling with the products in inventory, which are gradually received over the week from inland suppliers. The objective is either to reduce the largest inventory level needed in the ILP, or to smooth the weekly workload. We develop a solution methodology that allows the handling of complex loading constraints related to dimensions and weight of the products. We model the problem as a mixed integer linear program and we develop a decomposition heuristic to solve it. We perform extensive computation tests on real instances provided by ECM. Compared with current industrial practices, our solutions yield an average improvement of 46.8% for the inventory reduction and of 25.8% for the smoothing of the workload. Our results highlight the benefit of jointly optimizing container loading and operations scheduling.

Keywords: Logistics, intermodal logistics platforms, cross-docking, loading constraints, MILP, decomposition heuristics.

*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 *Computers & Operations Research*

1. Introduction

We consider the operational management of intermodal logistics platforms (ILP) of a large European car manufacturer denoted by ECM because of a non-disclosure agreement. We refer to this problem as the *ECM problem*. Over a given planning horizon (a week in this work, excluding the weekend), each ILP consolidates product flows from inland suppliers to offshore production plants, which are the ILP clients. Every day, products are unloaded from trucks and are then loaded into containers which are sent to the clients by ship at the end of the week. The products not shipped at the end of a day are stored and wait until the next day to be loaded on a container. Figure 1 illustrates the sequence of operations at an ILP. It shows a product flow from trucks to an ILP, to containers, ships to clients.

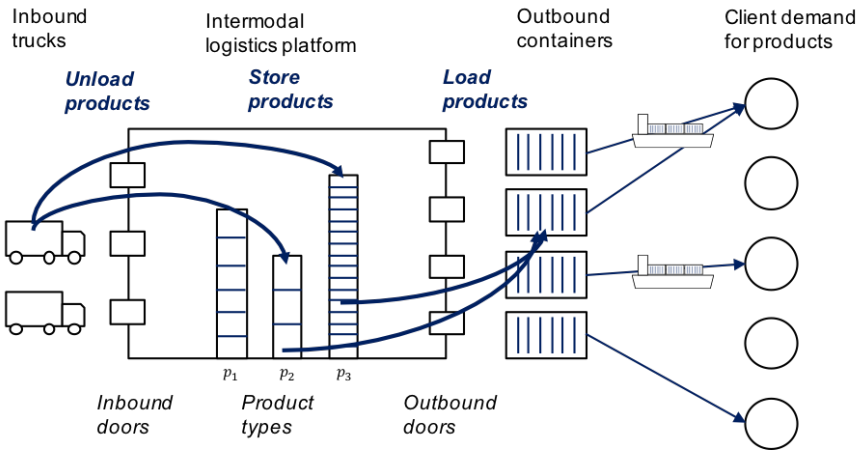

Figure 1: Product flows associated with an ILP.

The managers have to decide when to load each container in order to minimize the largest weekly inventory space (f^I) employed, or to smooth the workload activities (f^W). Focusing on f^I, it is preferable to load the containers as soon as possible. Minimizing f^W yield solution with approximately the same number of containers filled each day. The two objectives f^I and f^W are minimized lexicographically, and the priority of an objective on the other depends on the considered ILP: the ILPs with a loaded volume larger than 10,000 m^3 per week focus on f^I, whereas the smaller ILPs focus on f^W.

We assume that the number of containers loaded per day is not constrained and the sequence of container loadings has no impact on the objective function. Each client can receive multiple containers, but each container can only be sent to its assigned client. Products of the same type are interchangeable among clients. The daily workload is measured as the volume of products unloaded from the trucks and loaded into containers. The objective f^I is measured as the largest volume, over the week, of products remaining in the inventory at the end of a day. Regarding the inbound side, the trucks gradually deliver over the week all products needed for the ILP clients. The truck arrival days and their contents are considered as input data.

Concerning the outbound side, the client demand must be sent in containers by ship. A container can only be loaded when its full content is available in the ILP inventory. A first decision is to determine the loading day of each container, given that its content is fixed. Minimizing the number

of containers (by optimizing their contents) to ship all the demand is a difficult problem due to the presence of complex loading requirements, which involve 3D constraints (each box has a 3D shape and overlaying is forbidden in the containers), a total weight limitation (the total weight of all boxes loaded in the same container cannot exceed 22 tonnes), and the arrangement of boxes in stacks (the boxes are loaded in stacks and the range of allowed weights is limited by the height of the boxes in the stack). For an overview of common loading constraints, see Toffolo et al. (2017). The current practice at ECM consists of first optimizing the loading of the containers, and next of computing a weekly schedule for the associated operations. In the present work, the loading and the scheduling of the containers are optimized simultaneously through the minimization of f^I or f^W.

As highlighted by Toffolo et al. (2017), the loading problem itself is rather complicated and cumbersome. ECM solves this problem by using a dedicated algorithm that minimizes the number of containers (to ship the weekly demand) and satisfies the full set of 3D loading constraints. Therefore, ECM provided us with a feasible initial assignment of products to containers, where the products are packaged into boxes, and the boxes are loaded into containers. There are several product types and different box types. Usually, various product types are eligible to be loaded in a box. However, once loaded, a box can only contain a single product type to be determined. Starting from the ECM box-to-container assignments, we propose to revoke the decisions concerning the allocation of products to boxes. More precisely, we can modify the full content of each box with a tolerance of 10 kg (hence precluding any violation of the weight of a stack), but not its dimensions. In other words, we allow some permutations between the content of the boxes. Based on the employed real data, we have observed that 70% of the boxes can contain different product types, which means that the proposed permutation search space is likely to be large enough for the generation of very different solutions and for exhibiting a significant optimization potential. This type of box-content permutations allows us to keep tractable the high complexity related to *container loading*, this in order to integrate it with *container scheduling*. This integration would be very cumbersome if we were to consider the loading problem in its full complexity. The left part of Figure 2 depicts an assignment of boxes to a container. In this example, there are three types of boxes: b_1, b_2 and b_3. The container is loaded with two boxes of type b_1, five boxes of type b_2 and three boxes of type b_3. The right part of the figure shows that each box of type b_1 can hold either four products of type p_1 (with a total weight of 74 kg) or two products of type p_2 (with a total weight of 71 kg).

We make the following scientific contributions. We propose a mixed integer linear programming (MILP) model and a decomposition heuristic to solve the ECM problem. Using real data, we compare our results with the current practice for three different ILPs, and we assess the benefits of integrating container loading and container scheduling.

The remainder of the paper is organized as follows. Section 2 surveys the related literature. In Section 3, after determining the complexity of the problem, we present the MILP, as well as three decomposition strategies. Section 4 proposes two ways of eliminating variables from the MILP formulation in order to reduce its size. The solution method is described in Section 5, followed in Section 6 by computational experiments, where the efficiency of the proposed heuristic is assessed by making comparison with optimal solutions and with current industrial practices. Conclusions follow in Section 7.

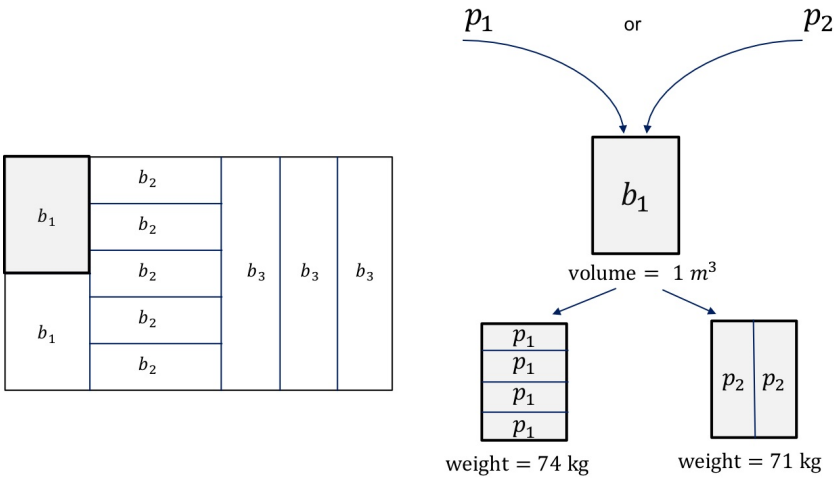

Figure 2: Assignment of boxes to a container and assignment of products to a box.

2. Literature review

The ECM problem shares some similarities with cross-dock scheduling. As in cross-docks, the ILPs act as consolidating points, the aim of which is to receive and unload an incoming flow of products arriving by truck, and then to load these products into outbound containers after a sorting process (Boysen and Fliedner 2010). Whereas in most of the cross-dock literature, the incoming products are immediately reloaded, hence precluding the use of storage, in the ILPs, products may have to be stored and wait at most until the end of the current week before being loaded into the containers. This is due to the fact that the totality of the content of a container must be available in order to launch the loading process. However, at a weekly level, all products received in the ILPs are sent (i.e., it is not possible to decide to hold any item in inventory for additional weeks) and the final storage at the end of the week is expected to be be null (like in classical cross-dock platforms at a daily level). The following paragraphs review the contributions on cross-docking that are the most related to the ECM problem.

According to the survey of Van Belle et al. (2012), most of the research on cross-docking has been undertaken after 2004. Only the literature concerning operational decisions is relevant to our study. Even though a number of papers that consider operational decisions share some specificities with the ECM problem (e.g., scheduling outbound flows when all the requested products are in inventory), their focus is usually on operational modeling, such as internal activities in the cross-dock (Bellanger et al. 2013), truck-to-door assignment (determining at which door a truck must be unloaded or loaded in order to minimize the movements in the cross-dock) (Enderer et al. 2017, Maknoon et al. 2017), or combining cross-dock and vehicle routing to minimize routing costs while satisfying internal constraints (Maknoon and Laporte 2017).

The ECM problem contains some features of the *Truck Scheduling Cross-Dock* (TSCD) problem, for which a review can be found in Boysen and Fliedner (2010). The TSCD focuses on the synchronization of inbound and outbound trucks to maximize the number of products that can directly be loaded in outbound trucks (Buijs et al. 2014), ideally while avoiding storage (Boysen 2010). Yu and Egbelu (2008) similarly consider a TSCD with storage considerations. As in the ECM problem,

the products arrive at the inbound doors from suppliers and are then sent by trucks to clients. The products are also interchangeable. The main difference between the work of Yu and Egbelu (2008) and ours lies in the fact that these authors consider only scalar constraints for the loading of outbound trucks (i.e., the constraints only concern the total weight of the transported products, but neither their size nor their position in the container is considered).

Serrano et al. (2017) studied a similar cross-docking platform, where temporary storage is allowed between truck arrival and container loading. However, in contrast to the ECM problem, scalar loading constraints are considered. The goal of the authors is to delay the minimum number of inbound trucks so that all internal constraints of the cross-docking platform are satisfied (i.e., storage, repacking and sorting activities).

Smoothing the workload has already been considered as an objective by Ladier et al. (2014). However, the context of their paper differs significantly from ours since the decisions involved concern the workforce dimensioning at a strategic level and the scheduling of specific activities during the day. As highlighted in (Merengo et al. 1999, Emde et al. 2010) in the context of assembly line balancing, there exist various criteria for workload smoothing (e.g., minimize either the sum of the divergences to the mean or the maximum divergence to the mean). In our situation, the choice of minimizing *the largest difference between the most loaded day and the least loaded one* results from discussions with ECM. Since the number of container-loading stations is fixed at the ECM ILPs, any high daily workload results in overtime, and any low daily workload creates idle times for some workers. Ultimately, the solutions obtained when minimizing the above-proposed function are likely to be similar to those that would be obtained when minimizing *the maximum divergence with respect to the mean daily workload*.

As highlighted by Toffolo et al. (2017), real-world multiple-container loading problems generalize the 3D packing problem. Accordingly, the associated combinatorial optimization problems are computationally complex due to the considered large set of specific constraints. As mentioned in Section 1, these constraints do not only concern the weight and size of the boxes to be loaded in the container, but also the specific way in which these boxes can be arranged into stacks and layers. In general, the objective focuses on minimizing the total number of containers required for packing the boxes. Container loading problems are most often solved using solution methods. In particular, Toffolo et al. (2017) propose a two-phase metaheuristic, the first phase of which consists of quickly generating a feasible assignment of the boxes to containers, which is then improved in a second phase through diversification and intensification mechanisms. Whereas loading and routing decisions have already been considered jointly (e.g., Gendreau et al. (2006)), no study seems to have addressed the complex set of real-world loading constraints arising in our problem in the context of optimizing the container-scheduling operations of a cross-dock. Here, we propose to integrate container loading and container scheduling decisions in the following way: starting from a feasible assignment of boxes to containers, we exploit the fact that some box types can transport different product types. As a result, we are able to take advantage of the huge variety of feasible assignments of products to containers while still satisfying the loading constraints.

3. Mathematical formulation

We discuss the complexity of the ECM problem in Section 3.1. In Section 3.2, we introduce the sets, parameters and variables needed to describe the model. Next, we present in Section 3.3 a quadratic formulation (Q) to accurately express the constraints and objectives. This formulation

is then linearized in Section 3.4. After discussing the size of the linear formulation (P), we propose three decompositions, fixing either the day in (P_t) (Section 3.5), the container contents in (P_x) (Section 3.6), or both in $(P_{t,x})$ (Section 3.7).

3.1. Complexity of the problem

To determine the complexity of the problem, assume the following:

- The planning horizon is limited to one day. Then, f^W is dropped and minimizing f^I is equivalent to maximizing the volume of the shipped products.
- The inbound trucks do not deliver all the demand during the limited planning horizon.
- The number of box types is equal to the number of product types (i.e., the content of the containers and their volumes is an input data that cannot be modified).

The resulting subproblem consists of choosing which containers to load during the day in order to maximize the shipped volume. The loading of a container is limited by the products available in the ILP inventory. Moreover, not all containers can be loaded because the demand has only been partially delivered. If there is only one product type, the subproblem is the *Knapsack Problem*, i.e., maximize a utility function while satisfying a volume constraint (Pisinger 1997). Since there is an inventory for each product type, this special case actually corresponds to the *Multidimensional Knapsack Problem* (MKP) (Puchinger et al. 2010), i.e., maximize a utility function (which is the shipped volume here) under multiple volume constraints (which correspond to the product availability here). This subproblem is described in Section 3.7. Since the MKP is NP-hard (Puchinger et al. 2010), the ECM problem is also NP-hard.

3.2. Sets, parameters and variables

We now introduce our notations:

Sets

- T: set of time periods (i.e., days),
- C: set of clients,
- O: set of outbound containers,
- $O^c \subseteq O$: subset of containers assigned to client $c \in C$,
- P: set of product types,
- B: set of box types.

Parameters

- $d_{cp} \in \mathbb{N}$: demand (in units) of client $c \in C$ for product type $p \in P$,
- $r_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ received on day $t \in T$,
- $n_{ob} \in \mathbb{N}$: number of units of boxes of type $b \in B$ transported in container $o \in O$,
- $q_{pb} \in \mathbb{N}$: number of units of product type $p \in P$ that can be transported in box type $b \in B$,
- $l_{pb} \in \mathbb{R}^+$: weight (in kg) of a box of type $b \in B$ when filled with product type $p \in P$,
- $l_M \in \mathbb{R}^+$: maximum allowed weight (in kg) that can be transported by a container,
- $a_b \in \mathbb{R}^+$: volume (in m^3) of a box of type $b \in B$,
- $h_p = \min_{b \in B}\{a_b/q_{pb}\}$: volume (in m^3) of a product of type $p \in P$.

Decision variables

- $x_{obp} \in \mathbb{N}$: number of boxes of type $b \in B$ assigned to product type $p \in P$ in container $o \in O$,
- $y_{ot} = 1$ if outbound container $o \in O$ is loaded on day $t \in T$; $y_{ot} = 0$, otherwise,
- $u_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ in stock on day $t \in T$ before loading the containers,
- $v_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ in stock on day $t \in T$ after loading the containers,
- $s_{pt} \in \mathbb{N}$: number of units of product type $p \in P$ sent on day $t \in T$,
- $w_t \in \mathbb{R}^+$: workload on day t (in m^3),
- $f^I \in \mathbb{R}^+$: largest inventory value (in m^3) encountered during the planning horizon,
- $f^W \in \mathbb{R}^+$: largest workload imbalance (in m^3), i.e., largest difference between the most loaded day and the least loaded one.

3.3. Quadratic linear programming formulation (Q)

ECM considers two objectives f^I and f^W, as mentioned in Section 1. These objectives are minimized in a lexicographic fashion (i.e., the higher-level objective is first minimized, and the lower-level objective is then minimized while constraining the first one at its best value). The priority of an objective depends on the considered ILP.

Objectives

$$\text{minimize} \quad f^I \tag{1}$$
$$\text{minimize} \quad f^W \tag{2}$$

Over the planning horizon, objective (1) aims at minimizing the largest storage space used in the ILP (f^I), whereas objective (2) focuses on minimizing the workload imbalance (f^W). Both objectives are measured in m^3.

Constraints

$$\sum_{t\in T} y_{ot} = 1 \qquad \forall\ o \in O \tag{3}$$

$$\sum_{o\in O^c}\sum_{b\in B} q_{pb}\cdot x_{obp} = d_{cp} \qquad \forall\ c \in C,\ p \in P \tag{4}$$

$$\sum_{b\in B}\sum_{p\in P} l_{pb}\cdot x_{obp} \le l_M \qquad \forall\ o \in O \tag{5}$$

$$\sum_{p\in P} x_{obp} \le n_{ob} \qquad \forall\ o \in O,\ b \in B \tag{6}$$

$$s_{pt} = \sum_{o\in O}\sum_{b\in B} q_{pb}\cdot x_{obp}\cdot y_{ot} \qquad \forall\ p \in P,\ t \in T \tag{7}$$

$$u_{pt} = v_{p,t-1} + r_{pt} \qquad \forall\ p \in P,\ t \in T \tag{8}$$

$$v_{pt} = u_{pt} - s_{pt} \qquad \forall\ p \in P,\ t \in T \tag{9}$$

$$v_{p0} \ge 0 \qquad \forall\ p \in P \tag{10}$$

$$f^I \ge \sum_{p\in P} h_p\cdot v_{pt} \qquad \forall\ t \in T \tag{11}$$

$$w_t = \sum_{p\in P} h_p\cdot r_{pt} + \sum_{o\in O}\sum_{b\in B}\sum_{p\in P} a_b\cdot x_{obp}\cdot y_{ot} \qquad \forall\ t \in T \tag{12}$$

$$f^W \ge w_{t_1} - w_{t_2} \qquad \forall\ t_1, t_2 \in T. \tag{13}$$

Constraints (3) prevent a container from being loaded multiple times. Constraints (4) impose that the demand of each client is satisfied. Constraints (5) ensure that the weight of the transported products does not exceed the container capacity. Similarly, constraints (6) ensure that the number of boxes transported in a container does not exceed the allowed limit. Constraints (7) compute the amount of product type $p \in P$ sent on day $t \in T$. Constraints (8) (resp. (9)) update the available inventory in the ILP before (resp. after) loading containers on day $t \in T$. Constraints (10) determine the initial inventory in the ILP at the beginning of the planning horizon (the products that are not received during the week are assumed to be in inventory at the beginning of the week). Constraints (11) compute the largest amount of storage space used in the ILP. Constraints (12) compute the workload for day $t \in T$. Constraints (13) evaluate the gap between the heaviest and lightest workloads over all days.

3.4. (P): Mixed integer linear programming formulation

We denote by (P) the linearized formulation of (Q) and by (P^I) (resp. (P^W)) formulation (P) in which f^I (resp. f^W) is the objective. Additionally, $(P^{I|W})$ (resp. $(P^{W|I})$) corresponds to formulation (P) where f^I (resp. f^W) is minimized and f^W (resp. f^I) is constrained at its best-known value. The variables z_{obpt} are introduced to linearize the product $x_{obp}\cdot y_{ot}$. Formulation (P) keeps the constraints of (Q) that do not involve the product $x_{obp}\cdot y_{ot}$ (i.e., constraints (3–6, 8–11, 13)). In (P), constraints (7) and (12) become constraints (14) and (15), respectively. In addition,

constraints (16–18) are added to fix z_{obpt} at its appropriate value (i.e., $z_{obpt} = 0$ if $y_{ot} = 0$, and $z_{obpt} = x_{obp}$ if $y_{ot} = 1$). The constraints of the linearized model are then:

$$s_{pt} = \sum_{o \in O} \sum_{b \in B} q_{pb} \cdot z_{obpt} \qquad \forall\, p \in P,\, t \in T \tag{14}$$

$$w_t = \sum_{p \in P} h_p \cdot r_{pt} + \sum_{o \in O} \sum_{b \in B} \sum_{p \in P} a_b \cdot z_{obpt} \qquad \forall\, t \in T \tag{15}$$

$$z_{obpt} \leq n_{ob} \cdot y_{ot} \qquad \forall\, t \in T,\, o \in O,\, p \in P,\, b \in B \tag{16}$$

$$z_{obpt} \leq x_{obp} \qquad \forall\, t \in T,\, o \in O,\, p \in P,\, b \in B \tag{17}$$

$$z_{obpt} + n_{ob} \cdot (1 - y_{ot}) \geq x_{obp} \qquad \forall\, t \in T,\, o \in O,\, p \in P,\, b \in B. \tag{18}$$

There are $|O| \cdot |P| \cdot |B| \cdot |T|$ variables z_{obpt} in the linearization, i.e., more than 40 million for the smallest instance and above 15 billion for the largest instance considered in this study (the exact number of variables is given in Table 2 for all the instances provided by ECM). The size of the proposed MILP precludes commercial solvers from finding solutions for the large instances, and even from inputting the data. We therefore introduce some decompositions.

3.5. (P_t): decomposition of (P) for a given day t

Formulation (P) can be decomposed into subproblems (P_t) for each day $t \in T$. In (P_t), the variables concerning the inventory (i.e., u_{pt} and v_{pt}) are dropped and the other variables remain the same (but the index related to day $t \in T$ is removed). Some notations are also modified: $u_p^{(t)}$ represents the amount of product type $p \in P$ available at the beginning of day t; $O^{(t)} \subseteq O$ represents the subset of containers to load at day $t \in T$; $d_{cp}^{(t)}$ is the demand (in units) of client $c \in C$ for product $p \in P$ not already loaded before day $t \in T$ (since other containers can be already shipped in previous days, and a part of the demand may be already satisfied); it replaces d_{cp} in constraints (4).

Regarding the objectives, f^W is dropped when optimizing over a single day, and f^I is equivalent to maximizing the volume of products sent at the end of the day. At the end of Section 5.2, we explain how (P_t) is used in the heuristic proposed to solve the ECM problem, and how f^W can be optimized a posteriori. The objective and constraints hence become:

$$\text{maximize} \sum_{o \in O^{(t)}} \sum_{p \in P} \sum_{b \in B} q_{pb} \cdot h_p \cdot z_{obp} \tag{19}$$

$$\text{subject to} \sum_{o \in O^{(t)}} \sum_{b \in B} q_{pb} \cdot z_{obp} \leq u_p^{(t)} \qquad \forall\, p \in P \tag{20}$$

$$z_{obp} \leq M \cdot y_o \qquad \forall\, o \in O^{(t)},\, p \in P,\, b \in B \tag{21}$$

$$z_{obp} \leq x_{obp} \qquad \forall\, o \in O^{(t)},\, p \in P,\, b \in B \tag{22}$$

$$z_{obp} + M \cdot (1 - y_o) \geq x_{obp} \qquad \forall\, o \in O^{(t)},\, p \in P,\, b \in B. \tag{23}$$

Objective (19) maximizes the volume sent at the end of the day. Constraints (4–6), on the assignment of products to containers are kept. The inventory constraints (8, 9, 14) are replaced with (20) which impose that the amount of products sent does not exceed the available inventory. Constraints (16–18) which are linked to the linearization are modified into constraints (21–23).

3.6. (P_x): decomposition of (P) for a given assignment x of products to boxes in containers

If the assignment of products to containers is known, like in the ECM current solution, the only decisions concern the loading day of each container. We call this problem the *Container Scheduling Problem*. It is related to a problem proposed by Larbi et al. (2011) in which a TSCD is considered and the content of both inbound and outbound trucks are known. A subproblem (P_x) is derived from (P), where the decision variables x_{obp} are fixed (e.g., those taken in a feasible solution, for instance the solution used by ECM), $m_{op} = \sum_{b \in B} x_{obp} \cdot q_{pb}$ is an input that represents the number of units of product p in container $o \in O$. The set B of boxes and the designation C of the clients are no longer needed. As for formulation (P), the following problems can be considered: $\left(P_x^I\right)$, $\left(P_x^W\right)$, $\left(P_x^{I|W}\right)$, $\left(P_x^{W|I}\right)$. With respect to formulation (P), the objectives remain the same. Constraints (4–6), related to the assignment of products to boxes, become redundant. Constraints (3, 8–11, 13) remain unchanged; constraints (7), which compute the amount of shipped products, become constraints (24); constraints (12), which set the workload, become constraints (25). The constraints then become:

$$s_{pt} = \sum_{o \in O} m_{op} \cdot y_{ot} \qquad \forall\ p \in P,\ t \in T \tag{24}$$

$$w_t = \sum_{p \in P} h_p \cdot r_{pt} + \sum_{o \in O} \sum_{p \in P} m_{op} \cdot h_p \cdot y_{ot} \qquad \forall\ t \in T. \tag{25}$$

3.7. $(P_{t,x})$: decomposition of (P_x) for a given day t

The model $(P_{t,x})$ uses the same formalism as (P_t) and (P_x). For a given day t, $(P_{t,x})$ maximizes the volume of containers loaded under the constraint of available inventory $(u_p^{(t)})$, as formulated below. This model is equivalent to the MKP. Indeed, the objective is to choose the appropriate containers to load in order to maximize the volume shipped at the end of the day, while imposing a capacity constraint for each product type:

$$\text{maximize} \sum_{o \in O^{(t)}} \left(\sum_{p \in P} m_{op} \cdot h_p \right) \cdot y_o \tag{26}$$

$$\text{subject to} \sum_{o \in O^{(t)}} m_{op} \cdot y_o \leq u_p^{(t)} \qquad \forall\ p \in P. \tag{27}$$

4. Elimination of variables

As already highlighted in Section 3.4, formulation (P) (as well as its decomposition (P_t)) involves the variables x_{obp}, the number of which increases with the cardinality of the sets O, B and P, which yields a number of x_{obp} variables so large that commercial solvers cannot even read the model (see column $|O| \cdot |B| \cdot |P|$ of Table 2 for an accurate evaluation of the number of variables x_{obp} in (P) and in its decompositions). In this section, we present a technique to fix the variables that can only take a single value at optimality, thus allowing their removal.

4.1. Fixing variables to 0

The variable x_{obp} counts the number of boxes of type b loaded with product of type p in container o. If container o does not transport a box of type b (i.e., if $n_{ob} = 0$), then the value of x_{obp} is forced to zero for each product type $p \in P$. Similarly, if box type b cannot transport a product of type p (i.e., $q_{pb} = 0$) because of non-matching sizes, weight limitations or product requirements, then x_{obp} is forced to zero for each container $o \in O$. These fixed variables can therefore be dropped from (P), thus reducing the size of the model. Let $X = \{x_{obp} \mid n_{ob} \cdot q_{pb} > 0\}$ be the set of variables that have not been fixed to 0. Column "$|X|$" of Table 2 provides the cardinality of this set for the ECM instances.

4.2. Single-value variables

A further reduction of X is achieved as follows. Formulation (P) looks for an optimal assignment of products to containers under the constraint that each box type b can only transport a subset of product types and that the assignment of boxes to containers is known (while satisfying complex loading constraints). We can count the maximum number of items of a given product type p that can be transported by the containers associated with each client c (i.e., $\sum_{o \in O^c} \sum_{b \in B} n_{ob} \cdot q_{pb}$). If this number is equal to the demand of client c, then the only way to satisfy this demand is to assign the products to the boxes that can transport them. More formally, let $\tilde{X} = \{x_{obp} \in X \mid \sum_{o' \in O^{c_o}} \sum_{b \in B} n_{ob} \cdot q_{pb} > d_{cp}\}$, where c_o is the client served by container o. The set of the non-fixed variables is $\tilde{X} \subseteq X$. The column "$|\tilde{X}|$" of Table 2 gives the size of set $\tilde{X}$ for the ECM instances.

4.3. Impact of variables elimination on the ECM instances

ECM operates three ILPs denoted by V, G and M. For each ILP, a representative set of data was provided by ECM, which captures the essential characteristics and situations observed over the past years. The ILPs V and G are very large, so that their storage space are not constraining, and hence the main objective at these locations is f^W. In contrast, f^I is the main objective for the ILP M since its large quantity of transiting products requires an important storage space, and the restricted available space must be used efficiently. Table 1 provides the characteristics of the 17 ECM instances under study. Column $|O|$ gives the number of containers to load during the week. Column $|B|$ (resp. $|P|$) indicates the number of different box types (resp. product types). Column $|C|$ gives the number of clients served by the ILP. Column "Nb. Boxes" (resp. "Nb. Products") gives the total number of boxes transported by the containers (resp. the total number of products transiting in the ILP). Column "% Boxes" displays the percentage of transported boxes that can receive a product different from the one carried in the ECM solution (within a tolerance of 10 kg per box). $I^{(sent)}$ represents the total inventory volume (in m^3) sent during the week (relative to all the products in column "Nb. Products"), and $I^{(init)}$ represents the inventory level of the ILP at the beginning of the week. In our instances, the products are either delivered during the week by inbound trucks, or are carried in the inventory from a previous week, and are therefore already available at the beginning of the week.

Table 2 provides the value of the product $|O| \cdot |B| \cdot |P|$ for each instance (i.e., the number of variables x_{obp}). Column "$|X|$" (resp. "$|\tilde{X}|$") indicates the size of set X (resp. $\tilde{X}$). Column "% non-fixed" gives the percentage of variables that are not fixed. After the variables elimination process, the resulting numbers of variables and constraints are presented in Table 3 for formulations (P), (P_t)

Table 1: Characteristics of the instances.

Instance	$\|O\|$	$\|B\|$	$\|P\|$	$\|C\|$	Nb. Products	Nb. Boxes	% Boxes	$I^{(sent)}$	$I^{(init)}$
V1	28	166	326	17	377,211	1,323	70.7%	1,680	659
V2	51	192	358	20	417,207	2,175	72.3%	3,285	1,277
V3	49	210	424	21	613,650	2,147	63.4%	2,915	962
V4	59	222	453	20	751,305	2,967	77.0%	3,920	1,821
G1	67	429	1,181	8	1,491,701	5,080	78.0%	4,461	1,283
G2	71	445	1,199	7	1,578,173	5,668	77.2%	4,921	1,250
G3	68	447	1,343	8	1,585,825	6,703	84.0%	5,131	1,301
G4	88	495	1,401	8	1,956,969	6,517	78.3%	5,978	2,027
G5	80	499	1,548	8	2,333,344	8,477	85.2%	6,109	1,830
G6	85	507	1,676	7	2,370,924	8,656	83.6%	6,442	1,409
M1	383	799	6,564	17	20,476,895	51,230	97.3%	78,398	16,417
M2	543	893	7,890	23	21,644,192	58,210	97.2%	89,937	16,244
M3	644	864	7,865	22	24,671,883	68,764	97.0%	105,082	15,946
M4	699	862	7,529	23	21,090,036	68,806	96.8%	109,501	27,048
M5	623	895	8,349	23	24,078,054	67,561	97.0%	106,073	16,661
M6	789	896	8,546	21	30,928,572	84,073	97.2%	130,476	27,328
M7	829	905	8,649	22	35,282,299	93,403	97.0%	142,679	30,716

and (P_x). Column "Nb. Var." (resp. "Nb. Constr.") indicates the number of variables (resp. constraints).

Table 2: Number of variables in (P).

Instance	$\|O\| \cdot \|B\| \cdot \|P\|$	$\|X\|$	$\|\tilde{X}\|$	% non-fixed
V1	1,515,248	668	442	0.029%
V2	3,505,536	883	519	0.015%
V3	4,362,960	1,065	675	0.015%
V4	5,933,394	1,305	883	0.015%
G1	33,945,483	13,758	12,797	0.038%
G2	37,882,405	15,284	14,188	0.037%
G3	40,821,828	18,622	17,507	0.043%
G4	61,027,560	19,618	17,944	0.029%
G5	61,796,160	27,683	26,282	0.043%
G6	72,227,220	28,874	27,603	0.038%
M1	2,008,695,588	420,090	407,608	0.020%
M2	3,825,853,110	518,412	503,904	0.013%
M3	4,376,211,840	601,015	584,093	0.013%
M4	4,536,508,602	542,537	527,444	0.012%
M5	4,655,277,165	651,286	634,152	0.014%
M6	6,041,543,424	736,934	718,116	0.012%
M7	6,488,869,005	833,631	812,775	0.013%

Table 3: Sizes of the formulations (P), (P_t) and (P_x) after the elimination of variables.

	(P)		(P_t)		(P_x)	
Instance	Nb. Var.	Nb. Constr.	Nb. Var.	Nb. Constr.	Nb. Var.	Nb. Constr.
V1	7,683	22,122	913	11,870	5,030	5,274
V2	8,740	30,597	1,090	18,918	5,625	5,809
V3	10,656	36,231	1,400	21,692	6,605	6,863
V4	12,389	42,799	1,826	25,319	7,090	7,337
G1	94,833	249,206	25,662	77,830	18,050	18,993
G2	103,469	272,164	28,448	83,822	18,340	19,285
G3	125,528	325,399	35,083	95,072	20,485	21,586
G4	129,120	346,550	35,977	110,089	21,455	22,534
G5	181,313	471,492	52,645	132,778	23,620	24,878
G6	191,184	495,888	55,292	139,397	25,565	26,931
M1	2,546,024	6,637,545	815,600	1,647,376	100,375	105,437
M2	3,144,490	8,352,285	1,008,352	2,186,514	121,065	126,813
M3	3,625,754	9,617,999	1,168,831	2,490,234	121,195	126,514
M4	3,281,095	8,809,257	1,055,588	2,366,265	116,430	121,193
M5	3,933,263	10,396,752	1,268,928	2,661,040	128,350	134,237
M6	4,440,832	11,796,494	1,437,022	3,050,093	132,135	137,555
M7	5,010,531	13,272,220	1,626,380	3,388,326	133,880	139,243

5. Methodology

We now describe two heuristics as well as the computation of lower bounds on f^I. The first heuristic is a greedy algorithm for the non-integrated version of the ECM problem (i.e., each container content is known and cannot be revoked). The second heuristic minimizes f^I or f^W while making decisions on both the loading of containers and on their loading day.

5.1. Heuristic for formulation (P_x)

The current practice at ECM complies with the following streamlined Algorithm 1. It starts from a given assignment of products to containers, which is the output of a first optimization step at ECM, and works day by day according to the level of products in inventory. Each day, containers are loaded as soon as the associated products are available, and container loading stops when a workload target is reached. For the subset of containers allowed to be loaded (i.e., for which the required products are in the inventory), various rules for the selection of the first container to load can be applied, which range from random selection to the consideration of specific characteristics of the ILP. Below, we only evaluate random selection, but we discuss the interest of considering specific rules in Section 6.4.1.

Algorithm 1 Greedy algorithm for (P_x)

Input: Assignment of products to containers (x_{obp}); initial inventory for each product p at the beginning of the week (u_{p0}); inflow product delivery schedule (r_{pt}).

For each day in the week, **do**

While there are containers allowed to be loaded **and** current workload is below average workload, **do**

(1) Choose a container: select a container from the set of containers allowed to be loaded.
(2) Load the selected container: remove the selected container from the list of containers to load and update the resulting inventory in the ILP.
(3) Update the set of allowed containers.

5.2. Heuristic for large instances

For those instances that are too large to be solved by commercial solvers, we describe in Algorithm 2 a heuristic based on the decomposition (P_t). Each day, based on the available inventory, the shipped volume is maximized. At the end of the day, the demand, the available products, and the list of non-loaded containers are updated. The strength of this heuristic lies in the fact that it allows decision makers to load containers based on past and present information only, without using forecasts on future product arrivals. As a result, the output of this algorithm is robust even though some suppliers do not deliver some products on their expected day. The overall computation time is proportional to the length of the planning horizon, since Algorithm 2 works day by day. Furthermore, at each step (i.e., every day) of the algorithm, the number of variables and constraints in (P_t) is smaller than the numbers reported in Table 3. Indeed, for the early steps, the cardinality of P becomes smaller since only a subset of products have been received. For the later steps, the cardinality of O also becomes smaller since many containers have been already shipped.

Algorithm 2 focuses on f^I, but it can easily be adapted to tackle f^W. Indeed, in addition to the loading day of each container, Algorithm 2 returns an assignment of products to containers that aims at maximizing the volume of product sent at each step t (day). To smooth the workload, the loading of some containers can simply be delayed. This can be achieved by solving (P_x^W), with the product-to-box assignment returned by Algorithm 2.

Algorithm 2 Heuristic to minimize the largest inventory

Input: Assignment of boxes to containers (n_{ob}); initial inventory for each product p at the beginning of the week (u_{p0}); inflow product delivery schedule (r_{pt}); client demand (d_{cp}).

For day $t = 1$ to 5, **do**

(1) Solve (P_t).
(2) Fix variables: move products from the inventory to the containers, with respect to (P_t).
(2) Update data: remove the loaded containers and update the available inventory and the client demand.

5.3. Lower bound on f^I

Considering a cumulative inventory for each day (i.e., the sum of all products received), we can compute the largest volume that can be shipped at the end of day $t \in T$. This yields an upper bound on the largest volume in inventory at the end of each day, and therefore a lower bound LB^I on f^I. This bound is formally computed based on the difference between the cumulative volume ($I_t^{(c)} = I^{(init)} + \sum_{t' \leq t} \sum_{p \in P} h_p \cdot r_{pt'}$) of products in stock at the beginning of day t, and the largest volume $f(P_t)$ that can be sent at the end of day t. The value of LB^I is then

$$LB^I = \max_{t \in T} \{I_t^{(c)} - f(P_t)\}. \tag{28}$$

The bound LB^I is used to evaluate – a posteriori – the quality of the solutions returned by Algorithm 2. Unfortunately, the computing time required to compute LB^I is of the same order of magnitude as the computing time required to perform Algorithm 2. Indeed, to compute LB^I, we need to solve five times formulation (P_t), which is similar to what is performed in Algorithm 2. If we first compute LB^I, and we next perform Algorithm 2 tightened by LB^I, this will not reduce the overall computing time.

6. Computational results

Section 6.1 introduces some notation needed to understand our numerical experiments. Section 6.2 compares, qualitatively, the different optimization approaches that can be applied to the ECM problem, which vary with respect to their degree of integration. Section 6.3 details the values of the optimal solutions for the V and G instances, which have a tractable size for CPLEX (see Table 3 for the size of the formulation after eliminating variables). Moreover, the output of Algorithm 2 is compared to these optimal values. Section 6.4 compares our results with the ECM current practice on all instances.

The formulations and all the heuristics were coded in C++. The solver is CPLEX 12.4 and is called with the concert technology. Computations were launched on a 2.2 GHz Intel Core i7 with 16 Go 1600 MHz DDR3 of RAM memory.

6.1. Notation

We use the formalism $f_{(ind)}(Form)$. The index *ind* indicates the solution method. More precisely, $ind = h$ if our *heuristic* (i.e., Algorithm 2) is applied, whereas $ind = g$ if the current-practice *greedy* heuristic (i.e., Algorithm 1) is employed. $Form$ indicates the formulation (see Section 3.4), among (P), (P^I), $(P^{I|W})$, $(P^{W|I})$, (P^I_x) and (P^W_x). In both (P^I_x) and (P^W_x), the assignment of products to containers is fixed as in the ECM current solution. $f^\star(Form)$ refers to the optimal solution of formulation $Form$. Tables (4–6) provide the execution times in minutes. Gaps are expressed in percent. The percentage gap between $f_{(h)}(P^I)$ and $f^\star(P^I)$ is denoted as $\%f^\star(P^I)$, and is computed as $\frac{f_{(h)}(P^I)-f^\star(P^I)}{f^\star(P^I)} \cdot 100$.

6.2. Comparison of the various optimization approaches

For objective f^I and for any instance, Figure 3 shows the expected ranking of the values LB^I, $f^\star(P^I)$, $f_{(h)}(P^I)$, $f^\star(P^I_x)$ and $f_{(g)}(P^I_x)$. We use a uniform step size between each pair of values, since initially we have no quantitative insight. The grey and black rectangles highlight the benefits of the main approaches. The rectangle "Non-integrated method" indicates the range of values that can be obtained when the container content is fixed: $f^\star(P^I_x)$ is the value of the optimal non-integrated solution. The rectangle "Integrating loading constraints" covers the solution values that can be reached in the case of a full, accurate but cumbersome, integration of the loading constraints to the container scheduling problem. The rectangle "Revoking products to boxes" shows where our results are expected to lie: $f_{(h)}(P^I)$ sets for the best-known solution value. More precisely, the difference between $f^\star(P^I_x)$ and $f^\star(P^I)$ corresponds to the largest achievable gain ensuring the non-violation of the loading constraints. Depending on the rule used to select the containers to be loaded each day, Algorithm 1 can be more or less efficient. Because the ECM practice of selecting the containers involves an experience-based understanding of the ILP that is hard to replicate, we use a random container selection for an estimation of the ECM results. Therefore, any current-practice solution value lies between $f^\star(P^I_x)$ and $f_{(g)}(P^I_x)$ (see the rectangle "Current practice"). The actual size of each rectangle of Figure 3 will be discussed in Section 6.4.2, relying on Figure 4.

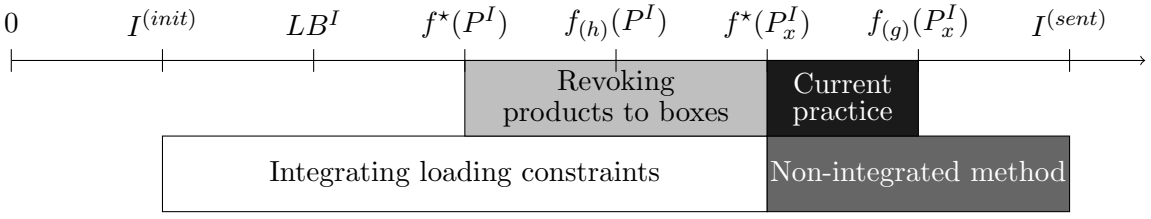

Figure 3: Comparison of the expected f^I-gains of the various approaches.

6.3. Optimal results for the V and G instances

Table 1 indicates that instances V and G are at least 10 times smaller than instances M. These much smaller sizes mean that these instances can be solved optimally by CPLEX with formulation

(P). While in these two cases f^W is the main objective, Table 4 gives the optimal values for both f^I and f^W, and presents a comparison with Algorithm 2 for both objectives f^I and f^W. The value of the second objective is also provided in columns "$f^\star\left(P^{I|W}\right)$" and "$f^\star\left(P^{W|I}\right)$".

Algorithm 2 returns an optimal solution in 17 of the 20 cases. Considering f^I, we obtain a low gain when the loading of a container allowed to be loaded on a given day is postponed by one day. In other words, it is preferable to load the containers as soon as possible. In addition, the objectives f^I and f^W may conflict. It may indeed be preferable not to load the containers early and to spread the work over the less busy following days. For example, in instance G4, the optimal value $f^\star(P^W)$ for the largest workload imbalance is 2,341 m^3 and the associated required largest storage space is 3,827 m^3. When decreasing the storage space to its optimal value $f^\star(P^I)$ (i.e., 3,745 m^3), the difference between the most and least busy days ($f^\star(P^{W|I})$) increases to 2,699 m^3. Since the workload also takes into account the unloading of inbound trucks, if many of them are unloaded during a specific day, it is preferable to delay some container loadings in order to smooth the workload.

Table 4: Optimal results for the V and G instances, and performance of Algorithm 2.

	Formulation (P^W)			Formulation (P^I)			Algorithm 2				
Instance	$f^\star(P^W)$	$f^\star(P^{I\|W})$	Time	$f^\star(P^I)$	$f^\star(P^{W\|I})$	Time	$f_{(h)}(P^W)$	Time	$\%f^\star(P^W)$	$f_{(h)}(P^I)$	$\%f^\star(P^I)$
V1	893	1,260	< 1	981	1,237	< 1	893	< 1	0.0%	982	0.1%
V2	1,579	1,922	< 1	1,246	1,960	< 1	1,579	< 1	0.0%	1,246	0.0%
V3	1,669	2,060	< 1	1,546	1,976	< 1	1,669	< 1	0.0%	1,546	0.0%
V4	2,233	2,715	< 1	1,894	2,690	< 1	2,233	< 1	0.0%	1,894	0.0%
G1	2,756	2,679	1	2,679	2,756	10	2,756	< 1	0.0%	2,679	0.0%
G2	1,508	3,182	13	3,076	1,718	6	1,508	< 1	0.0%	3,076	0.0%
G3	3,357	3,642	< 1	2,977	4,022	5	3,357	< 1	0.0%	2,977	0.0%
G4	2,341	3,827	14	3,745	2,699	12	2,479	< 1	5.8%	3,745	0.0%
G5	3,060	4,061	13	3,966	4,118	9	3,060	< 1	0.0%	3,968	0.1%
G6	3,204	4,603	13	4,065	3,345	1	3,204	< 1	0.0%	4,065	0.0%

6.4. Comparison with current practice

In Sections 6.4.1 and 6.4.2, we compare the best-known solution values for (P^W) and (P^I) with the output of Algorithm 1 and with the optimal solution values of (P^W_x) and (P^I_x) (i.e., an estimate of the values observed in practice). Only those results concerning the main objective are considered for the comparison (i.e., f^W for instances V and G, f^I for instances M), since no additional insight can be gained from the conflict between the two objectives.

6.4.1. f^W-values for instances V and G

Table 5 compares the optimal solution values $f^\star(P^W)$ for formulation (P^W), the optimal solution values $f^\star\left(P^W_x\right)$ for formulation (P^W_x) (when the container contents are fixed by ECM), and the output $f_{(g)}(P^W)$ of Algorithm 1 (which estimates the ECM results). The gaps between the different formulations with respect to $f^\star(P^W)$ and $f^\star\left(P^W_x\right)$ are expressed in the columns "%...". We observe that even if Algorithm 1 is fast (it requires less than a second of execution time), it delivers results on formulation (P^W_x) for which there is substantial room for further improvement. When

the containers content is fixed, the percentage gap between the values returned by Algorithm 1 and the optimal values is on average of 16.8% for the V instances and of 12.6% for the G instances. Algorithm 1 could be further improved (with respect to $f^\star(P_x^W)$) with the use of more refined rules for selecting the containers to load. This has not been investigated since the optimal solution values for formulation P_x^W are obtained within less than a minute with CPLEX. Table 5 also shows that a substantial gain can be achieved by integrating the loading decisions in the container scheduling problem (see column "$\%f^\star(P^W)$" under "Formulation (P_x^W)"). Indeed, the average percentage gap between (1) the optimal solution values of (P_x^W) using the ECM current assignment of products to containers, and (2) the optimal solution values of (P^W), is on average 6.1% for the V instances and 33.6% for the G instances. More generally, we observe that the average gain becomes larger as the instance size increases. Indeed, the G instances involve a volume of handled products that is on average five times larger than for the V instances.

Table 5: Results for the V and G instances (i.e., focusing on f^W).

	Formulation (P^W)		Formulation (P_x^W)			Current practice (Algorithm 1)			
Instance	$f^\star(P^W)$	Time	$f^\star(P_x^W)$	Time	$\%f^\star(P^W)$	$f_{(g)}(P_x^W)$	Time	$\%f^\star(P_x^W)$	$\%f^\star(P^W)$
V1	893	< 1	893	< 1	0.0%	995	< 1	11.4%	11.4%
V2	1,579	< 1	1,671	< 1	5.8%	2,028	< 1	21.4%	28.4%
V3	1,669	< 1	1,758	< 1	5.3%	2,055	< 1	16.9%	23.1%
V4	2,233	< 1	2,445	< 1	9.5%	2,829	< 1	15.7%	26.7%
G1	2,756	1	3,407	< 1	23.6%	3,650	< 1	7.1%	32.4%
G2	1,508	13	2,122	< 1	40.7%	3,313	< 1	56.1%	119.7%
G3	3,357	< 1	4,397	< 1	31.0%	4,940	< 1	12.3%	47.2%
G4	2,341	14	3,125	< 1	33.5%	3,717	< 1	18.9%	58.8%
G5	3,060	13	4,152	< 1	35.7%	4,241	< 1	2.1%	38.6%
G6	3,204	13	4,469	< 1	39.5%	4,542	< 1	1.6%	41.8%

6.4.2. f^I-values for instances M

Table 6 compares three approaches for minimizing f^I, which is the main objective for the M instances: the output $f_{(h)}(P^I)$ of Algorithm 2, the optimal results using the ECM current assignment $f^\star(P_x^I)$ of products to containers, and the output $f_{(g)}(P^I)$ of Algorithm 1. The lower bound LB^I is also provided for each instance.

We observe from Table 6 that Algorithm 1 is very efficient for formulation (P_x^I). Indeed, its optimality gap never exceeds 2%. In other words, given a feasible assignment of products to containers, only marginal gains can be achieved by using CPLEX to minimize the storage in the ILP. As mentioned in Section 5.1, Algorithm 1 randomly selects the next container to load (see step (1) of Algorithm 1). Therefore, using more refined selecting rules at step (1) could only yield marginal gains (between 0.2% and 1.9%). We observe that for (P_x^I), which is a more difficult problem than MKP (see Section 3.6), both CPLEX with an execution time of less than a second, and Algorithm 1, with an optimality gap below 2%, exhibit good performances. This can be explained by the fact that the instances have a significant proportion (between 50% and 70%) of the product types that can only be assigned to a single container. Therefore, the solution space of these instances remains of tractable size.

As shown in Table 6, revoking the assignment of products to containers can lead to substantial gains. Column "$\%f_{(h)}(P^I)$" under "Formulation (P_x^I)" displays the gain found by revoking the loading of containers (compared to the optimal solutions with the ECM current assignment of products to containers). It lies between 26.6% and 72.5%. The average gap between the best-known solution value and the best non-integrated solution value is 46.8%. Additionally, the lower bounds presented in column "LB^I" indicate that the output of Algorithm 2 is close to optimality, except for one instance for which the optimality gap is 18.8%. We conclude that an algorithm that would consider future information to schedule the containers (i.e., the inbound flows during the next days) could only yield a marginal gain.

Table 6: Results for the M instances (i.e., focusing on f^I).

		Algorithm 2 (P^I)			Formulation (P_x^I)			Current practice (Algorithm 1)			
Instance	LB^I	$f_{(h)}(P^I)$	Time	$\%LB^I$	$f^\star(P_x^I)$	Time	$\%f_{(h)}(P^I)$	$f_{(g)}(P_x^I)$	Time	$\%f^\star(P_x^I)$	$\%f_{(h)}(P^I)$
M1	33,653	36,162	16	6.9%	62,399	< 1	72.5%	62,829	< 1	0.7%	73.7%
M2	42,980	44,465	29	3.3%	59,049	< 1	32.8%	59,314	< 1	0.4%	33.4%
M3	54,897	55,386	30	0.9%	75,727	< 1	36.7%	75,875	< 1	0.2%	37.0%
M4	49,313	50,751	32	2.8%	72,109	< 1	42.1%	72,296	< 1	0.3%	42.5%
M5	56,605	57,889	35	2.2%	73,294	< 1	26.6%	73,461	< 1	0.2%	26.9%
M6	53,614	56,191	41	4.6%	93,335	< 1	66.1%	93,843	< 1	0.5%	67.0%
M7	51,415	63,293	45	18.8%	98,630	< 1	55.8%	100,540	< 1	1.9%	58.8%

Figure 4 quantifies the qualitative aspects displayed in Figure 3 with the average values computed on all M instances. It illustrates that the current practice offers very few improvement opportunities when considering the used non-integrated approach (the rectangle "Current practice" is small compared with the rectangle "Revoking product to boxes"). It also shows that the potential improvement on our heuristic is small since the gap to the lower bound LB^I is on average of 6.3% over all instances. Finally, among the 97% of boxes for which a change of product assignment is possible (average over all M instances, see Table 1), our solutions yield a box content that is different from the one proposed by ECM for 64.8% of the boxes. In other words, our optimization results yield very different container loading plans while satisfying all loading constraints.

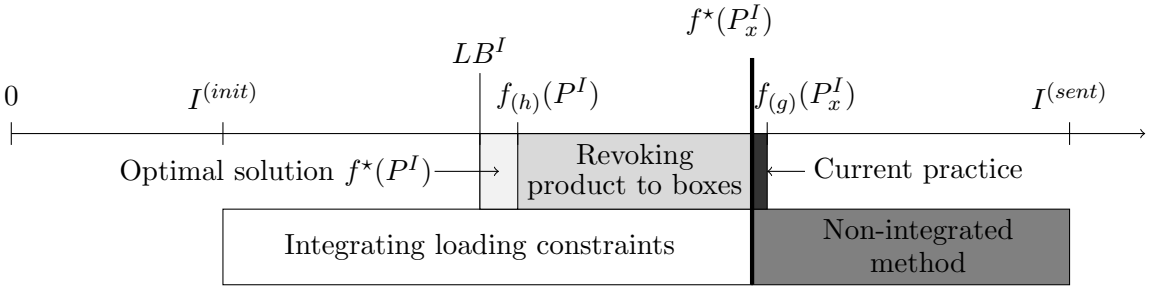

Figure 4: Quantification of the expected f^I-gains for various approaches (average values for all M instances).

7. Conclusions

We have modeled and solved a scheduling of outbound flows in an ILP under complex loading constraints. This problem was proposed by ECM and is encountered in three of its ILPs. The aim was to compute an improved schedule for the loading day of each container to either reduce the largest required inventory space or the weekly workload imbalance.

To solve the problem, we have developed both an exact algorithm and a heuristic. Whereas the exact approach is able to solve to optimality two thirds of the provided industrial instances, the heuristic can efficiently tackle the remaining larger instances. Matching the industrial requirements concerning the execution time (i.e., less than one hour), the heuristic returns results with an optimality gap lower than 7% for 85% of the large instances. Furthermore, the heuristic allows handling the uncertainty in the inbound flows. Indeed, when the suppliers do not exactly deliver the products on their scheduled day, the heuristic can be relaunched based on the actual deliveries without impacting its performance. Results show that compared with current practice, our heuristic yields a 26% to 73% improvement in the inventory level, with an average of 46.8%. Similarly, an average improvement of 25.8% was found for the smoothing of the workload. In other words, substantial gains can be achieved for both objectives.

From a managerial point of view, the efficiency gains for the ILPs are achievable without involving third-parties, since neither the suppliers nor the clients of the ILP are impacted by the considered decisions (i.e., each product arrival day at the ILP remains the same and all the client demands are covered at the end of the week). We have shown that optimizing only the container loading days yields marginal gains. Indeed, if the container contents remain unchanged, a gain not exceeding 2% can be achieved on the largest required inventory space. In contrast, adapting the container contents based on the inbound deliveries, while optimizing the loading day of the containers, helps to significantly improve the objective value.

As a future avenue of research, we mention the integration of additional types of decisions. For instance, one may also determine the arrival schedule of the inflow trucks and the allocation of resources to the container loading platforms.

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

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

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

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

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

Emde S, Boysen N, Armin S (2010) Balancing mixed-model assembly lines: a computational evaluation of objectives to smoothen workload. *International Journal of Production Research* 48(11):3173–3191.

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

Gendreau M, Iori M, Laporte G, Martello S (2006) A tabu search algorithm for a routing and container loading problem. *Transportation Science* 40(3):342–350.

Ladier AL, Alpan G, Penz B (2014) Joint employee weekly timetabling and daily rostering: A decision-support tool for a logistics platform. *European Journal of Operational Research* 234(1):278–291.

Larbi R, Alpan G, Baptiste P, Penz B (2011) Scheduling cross docking operations under full, partial and no information on inbound arrivals. *Computers & Operations Research* 38(6):889–900.

Maknoon Y, Laporte G (2017) Vehicle routing with cross-dock selection. *Computers & Operations Research* 77:254–266.

Maknoon Y, Soumis F, Baptiste P (2017) An integer programming approach to scheduling the transshipment of products at cross-docks in less-than-truckload industries. *Computers & Operations Research* 82:167–179.

Merengo C, Nava F, Pozetti A (1999) Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research* 37:2835–2860.

Pisinger D (1997) A minimal algorithm for the 0-1 knapsack problem. *Operations Research* 45(5):758–767.

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

Serrano C, Delorme X, 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.

Toffolo TA, Esprit E, Wauters T, 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, Cattrysse D (2012) Cross-docking: State of the art. *Omega* 40(6):827–846.

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