POLITECNICO DI TORINO
Repository ISTITUZIONALE

Inventory reallocation in a fashion retail network: A matheuristic approach

Original
Inventory reallocation in a fashion retail network: A matheuristic approach / Brandimarte, P., Craparotta, G., Marocco, E.. - In: EUROPEAN JOURNAL OF OPERATIONAL RESEARCH. - ISSN 0377-2217. - STAMPA. - 317:(2024), pp. 603-615. [10.1016/j.ejor.2024.04.016]

Availability:
This version is available at: 11583/2988405 since: 2024-05-20T11:24:45Z

Publisher:
Elsevier

Published
DOI:10.1016/j.ejor.2024.04.016

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

01 November 2026

European Journal of Operational Research 317 (2024) 603–615

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Innovative applications of O.R.

Inventory reallocation in a fashion retail network: A matheuristic approach

Paolo Brandimarte [a,*], Giuseppe Craparotta [b], Elena Marocco [b]

[a] Dipartimento di Scienze Matematiche "Giuseppe Luigi Lagrange", DISMA, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy
[b] Evo Europe Limited, 3rd Floor 207 Regent Street, London, W1B 3HH, United Kingdom

ARTICLE INFO

Keywords:
Heuristics
Fashion retailing
Inventory rebalancing
Lateral shipments

ABSTRACT

We consider a fashion retail network consisting of a central warehouse, owned by a fashion firm, and a fairly large number of retail stores. Some stores are owned by the firm itself, whereas others are owned by franchisees. An initial inventory allocation decision is made at the beginning of the selling season and is periodically revised. Inventory reallocation comprises both direct shipments from the warehouse to stores and lateral shipments among the stores. Besides stock availability and shipping costs, a suitable reallocation policy must take into account the probability of selling each item, some operational constraints, as well as other preference factors that define the utility of shipping an item from a node of the network to another one. Since the problem does not lend itself to the application of typical tools from inventory theory, we propose an optimization model that complements such tools. The model, given the number of nodes and SKUs, may involve about one million binary variables, and just solving the LP relaxation may take hours using state-of-the-art software. Since typical metaheuristics for combinatorial optimization do not seem a viable alternative, we propose a matheuristic approach, in which a sequence of maximum-weight matching problems is solved in order to reduce the problem and restrict the set of potential shipping pairs, with a corresponding drop in the number of decision variables. Computational results obtained on a set of real-life problem instances are discussed, showing the viability of the proposed algorithm.

1. Introduction and problem statement

The use of lateral shipments to rebalance inventory across multiple locations is a key topic in inventory management (Paterson et al., 2011), and early references like (Allen, 1958, 1961) adequately point out its relevance. A wide variety of tools have been adopted to tackle problems in this class, including stochastic modeling and dynamic programming. In this paper, due to the peculiar nature of the problem that we address, we resort to a different framework, based on an integer linear programming model, whose solution requires the adoption of a suitable matheuristic. As we discuss in the following, key features of our case are the presence of several retail stores, whose managers may have different incentives, as well as a large number of items. Due to limitations on the number of outbound shipments that a store may manage, there are interactions among items, making the decomposition of the overall problem into a set of single-item subproblems inadequate.

More specifically, we deal with the problem of finding the best way to match offers and requests for fashion items across a network consisting of several retail stores and a central warehouse. It is often the case that a retail store missing an item asks other stores to supply that item. It is important to notice that this kind of lateral shipment may be of a different nature with respect to centrally managed and proactive lateral shipments planned according to formal and rational inventory control theory. On the one hand, requests for inter-store shipments are sometimes informally managed on a case-by case basis, possibly according to some business rules, even though they must be centrally authorized. On the other one, lateral shipments may be reactive, i.e., aimed at satisfying a specific customer request. In practice, how lateral shipments are managed is heavily influenced by the specific industry in which they are carried out. The case we deal with in this paper arose from the need to rationalize the practice of inventory management for an Italian fashion firm, Miroglio, which manages a retail network comprising about 200 stores, where a large number of SKUs are distributed.

The fashion industry setting is characterized by two essential features:

1. significant demand volatility due to the fashion content of items;
2. the need for fast execution, due to the limited sales time window.

* Corresponding author.
E-mail addresses: paolo.brandimarte@polito.it (P. Brandimarte), gcraparotta@toolsgroup.com (G. Craparotta), emarocco@toolsgroup.com (E. Marocco).
URLs: https://staff.polito.it/paolo.brandimarte/ (P. Brandimarte), https://www.toolsgroup.com/ (G. Craparotta), https://www.toolsgroup.com/ (E. Marocco).

https://doi.org/10.1016/j.ejor.2024.04.016
Received 23 December 2022; Accepted 18 April 2024
Available online 21 April 2024
0377-2217/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Moreover, the specific problem that we consider in this paper is complicated by a few additional peculiarities.

- Lateral shipments are both reactive and proactive. Proactive shipments may arise from updated sales forecasts, whereas reactive shipments may arise from specific customer requests (when they face a stockout, but are willing to wait for a specific item). An additional reason for lateral shipments is related to broken assortments, which occur when only a few sizes of a model are available at some stores, and the assortment must be consolidated in order to offer the full range of sizes.
- The retail network consists of both owned and franchised stores. Hence, store managers may react in different was to requests to release items to another store. Accounting for managers' incentives results in a problem where objective costs and revenues are mixed with intangible factors. Moreover, stores are not well equipped to deal with shipments, and the time spent to arrange them may be detrimental to store operations and customer care. Hence, there is a limit on the total number of outgoing shipments per week, especially from franchised ones, which is partly a consequence of intangible issues, rather than hard shipping costs.
- In addition to lateral store-to-store shipments, the central warehouse progressively releases items.

The problem requires to find the best way to match offers and requests for a large number of stores and a large number of SKUs. As we show later, in order to account from this variety of issues, we associate a utility coefficient to each possible match between an offer and a request. The resulting model is a sort of max-weight matching problem with side constraints, which may be formulated as a pure Binary Integer Programming (BIP) model. Due to the size of some real life problem instances, the model cannot be solved by commercial solvers, and we need heuristics. We will resort to a matheuristic exploiting a sequence of easy subproblems model in order to reduce the size of the overall model, which is then solved by a commercial solver. A significant advantage of this approach, with respect to alternative metaheuristics, is its flexibility in dealing with additional constraints and variations of the basic problem that we describe here.

It is important to stress that our approach is *not* meant to be a substitute to typical decision models based on stochastic inventory control, but rather a substitute. The model we propose is meant to rationalize and support a key business process in a real-life setting, in a flexible way, and it is agnostic with respect to demand forecasting approaches and business priorities, which may be easily incorporated into its input data.

1.1. Plan of the paper

In Section 2 we provide motivation for the study by placing it within a real industrial context and by describing how our solution approach fits within the overall decision architecture. In Section 4 we state the problem formally and discuss possible modeling choices, with particular emphasis on one possible way to quantify the marginal utility for each shipped item. The matheuristic model reduction approach is the topic of Section 5. Computational tests on real life problem instances are described in Section 6. Finally, we conclude with Section 7, where we also comment on the actual use of the model.

2. Motivation and industrial context

The work described in this paper was motivated by the requirements of Miroglio Fashion, Italy's third-largest retailer of women's apparel, with €520m+ annual revenues from a large number of stores across 11 brands. Like many fashion retail companies, Miroglio has to address a sequence of difficult problems, comprising design of the next collection, sourcing of raw materials, production planning under a significant degree of uncertainty, inventory management at both central warehouse and store level, and markdown/promotion decisions. In 2016, the management at Miroglio perceived the need of improved and optimized store inventory management, and they adopted a new replenishment tool to improve inventory management with a more data-driven approach, also involving store managers in the process. The tool was provided by Evo, a company based in Turin and London, focused on dynamic pricing, predictive supply chain management, and customer scoring. Evo, which recently became part of Toolsgroup, provides a range of tools for retail players, which aim at optimizing retail business decisions and extract value from company internal data and external data, combining artificial intelligence and human experience. The results of the cooperation between Miroglio and Evo to develop a more structured approach to decision-making about markdowns and promotions are described in a series of Harvard Business School (HBS) business cases (Gupta, 2019a, 2019b; Gupta & Lane, 2019). The goal of the overall system is to support decision making within a highly volatile and hardly predictable fashion market. The ultimate aim is to maximize expected profit, but in order to achieve this target, there are several subproblems that must be solved, including inventory rebalancing. Since store sales depend on several factors and are not uniform over the network, the need arises of periodically reallocating inventory over the network, in order to ship items where they are most requested, as well as to cope with broken assortment issues. This is the specific subproblem that we address in this paper, but we believe that it is useful to place it within a more general framework, in order to appreciate its relevance.

The retail network of Miroglio consists of more than 1000 stores. In particular, our case study focuses on two of the major brands of the company, which operate approximately 350 stores, collectively selling about 2 million pieces per year, with an average item selling price of €50. The number of distinct models (references) for each of the two brands exceeds 800 for the Winter collection and 1200 for the Spring collection. An individual Stock-Keeping Unit (SKU) is a pair featuring a model and a size. A model is usually available in 4–5 different sizes, so the number of distinct SKUs is about 3800 for the Winter collection and 5200 for the Spring collection. Given the wide SKU range, and the relatively short product lifecycle, it is hard to find the optimal initial allocation of items to stores. Hence, the company keeps some pieces in stock in the central warehouse, in order to ship them later to the most suitable stores. An important feature of the retail network is that stores are heterogeneous, since they can be directly owned by the company or franchised, and are associated with different brands, aimed at possibly different market segments. Furthermore, stores are distributed over all Italy, with an impact on sales due to taste and meteorological factors: a relatively heavy garment may sell less in Southern Italy than in Northern Italy. As a result, sales are not uniform across the network, and it can be useful to redistribute items among different stores, in order to better match inventory availability and customers demand. The logistic network of the company reflects these needs. As shown in Fig. 1, there are flows from the central warehouse to retail stores, as well as lateral shipments among retail stores. Further inventory release from the central warehouse and lateral shipments are carried out weekly, or every other week, depending on the brand. Such additional shipments are based on matching a list of SKU offers, which includes available inventory at the central warehouse of the firm, and a list of requests. When available inventory is not sufficient to meet requests, inventory rationing must be carried out. In the past, because of the organizational complexity involved, as well as the need to arrange and execute shipments quickly, given the fashion content of items, the firm did not want to consider lateral shipments among retail stores, as they lead to an informal and disordered negotiation process among store managers and the central warehouse management. The role of the model we describe here is to support this specific business process, rather than the full-fledged inventory management problem.

Besides speed of execution, there are a number of operational constraints that must be taken into account:

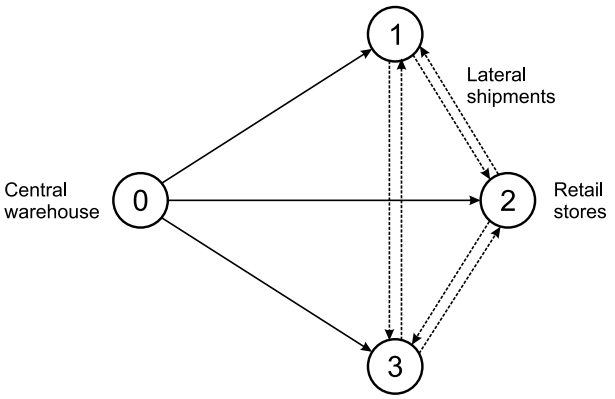


Fig. 1. An illustration of the network structure and shipping patterns. Dotted arcs refer to lateral shipments, whereas shipments from the central warehouse are depicted by a continuous line.

- Since the firm uses fast couriers, the shipments are always point-to-point, unlike some inventory routing problems with a VRP flavor (Bertazzi & Speranza, 2012; Cordeau et al., 2015).
- The warehouse cannot act as a transit point among stores, as unpacking incoming parcels and sorting out the content for outbound shipments would imply a significant operational workload, which would slow down the reallocation process and cannot be sustained.
- It is the responsibility of store managers to prepare the lateral shipments. However, as we pointed out in the introduction, there is no specific personnel for this task. Hence, an excessive time devoted to prepare shipments could have an adverse effect on store operations and lead to dissatisfaction on the part of both managers and employees. The issue is not related with a hard cost, but rather to intangibles, and it is particularly relevant at franchised stores. In order to take care of the issue, an upper bound is enforced on the number of outbound shipments from retail stores.

As a result, there is a minimum amount that is worth shipping between two retail stores, in terms of monetary value, as well as a maximum number of outgoing shipments from stores. It is worth noting that this has more to do with the difficulty of arranging parcel preparation and shipping at each store than with the shipping cost itself, which has a limited impact due to arrangements with the couriers. On the contrary, the only constraint on shipping from the central warehouse is related to stock availability. The frequency with which inventory reallocations are planned depends on contingencies, as well as the store brand, but lateral shipments may be carried out every week or every other week. Nevertheless, the demand forecasts that are the one of the inputs to quantify the utility of each match, as we shall see in Section 4.1, refer to the full selling season. Thus the time horizon is progressively shrunk.

Last but certainly not least, we should also assess the expected benefit of meeting a request for shipping additional stock to a retail store. Demand is uncertain and there is little point in shipping an item that has a low probability of being sold. In principle, one could formulate a stochastic optimization problem aiming at maximizing expected profit. However, a full-fledged stochastic optimization model is out of the question. Leaving computational issues aside, characterizing demand uncertainty for several items across several locations is a hopeless endeavor. Hence, the proposed model is deterministic and relies on marginally decreasing utilities associated with the shipment of each item from a source to a destination node. The estimation of such utilities will be clarified later, but this modeling choice results in a large-scale binary programming model, which requires an *ad hoc* solution approach. A matheuristic approach has been developed, based on a sequence of maximum-weight matching problems aimed at defining a restricted model, where only a subset of "interesting" decision variables is included. Then, the restricted model is solved using standard branch-and-cut.

3. Literature review and paper positioning

In this section we review some relevant literature in order to position our contribution. We want to highlight that our paper is clearly related to the well-known topic of lateral shipments, but the motivating problem has some specific features that set it apart from other studies.

The subject of lateral shipments on a network of inventory locations has been investigated for a long time. An early paper from the 60 s is (Allen, 1961). Indeed, given the sheer size of this literature, this section is not meant to be an extensive review, which is provided, e.g., by Paterson et al. (2011) and references therein. Actually, there is a variety of approaches involving lateral shipments, which may be reactive or proactive, and possibly integrated with replenishments. Here we just want to point out the building blocks of the overall approach and to position the proposed model and matheuristic solution strategy with respect to the available literature.

When demand is subject to considerable uncertainty, as is the case with fashion items, a common practice is to avoid committing all of the available inventory immediately to retail stores. Some inventory is kept centralized, which is a form of risk pooling strategy (Brandimarte & Zotteri, 2007). Furthermore, leaving room for multistage decisions, to be taken sequentially after sales information is collected, is a recommended strategy under significant uncertainty conditions (Fisher et al., 1994; Fisher & Raman, 1996). Besides sequential release of items from a centralized location, lateral shipments are another useful tool to cope with uncertainty A fair share of papers deals with a limited number of items or locations, and the most common approach relies on stochastic modeling for inventory control. See, for instance, Agrawal et al. (2004), Amrani and Khmelnitsky (2017), van Wijk et al. (2019), to appreciate the possible application of dynamic programming or stochastic programming. A relevant paper, in the retail context, is Agrawal and Smith (2009). See also Choi (2014), Fisher et al. (2001). The latter references deal with a stochastic model based on a dynamic programming recursion. We should also mention Grahovac and Chakravarty (2001), who deal with a case involving low-demand and expensive items, which is also relevant for fashion applications. Their approach relies on stochastic modeling, and it is limited to a single item.

On the contrary, the approach taken in this paper relies on an integer programming formulation and is quite similar in spirit to the model formulations proposed for Zara by Caro and Gallien (2010), Caro et al. (2010), as well as to the approach followed in Wang et al. (2022). The sheer size of the problem, as well as the complexity of the involved constraints and the need for possible model extensions, discourage a metaheuristic approach based on some form of local search. A matheuristic approach, which actually takes advantage of the mathematical model formulation may look more promising. A wide array of such methods have been proposed (Maniezzo et al., 2021). Common strategies are to solve a sequence of LP relaxations, progressively fixing integer variables to integer values, or to limit the number of integer variables in order to ease the computational effort of standard branch-and-cut. Indeed, this is the approach pursued by Wang et al. (2022), where a relax-and-fix strategy is successfully applied for planning lateral shipments in a fashion retail network. Unfortunately, their strategy cannot be applied to our model. The reason is the quite different nature of the two models:

- They consider a multiperiod model under the assumption that demand is perfectly forecast. As a result, their model involves the typical inventory balance equations, plus variables related to lateral shipments. The only operational constraint on shipments is related to a minimum shipment of a single SKU, and everything is centrally planned.

- On the contrary, as fully explained in the following section, in our model we consider demand uncertainty through expected utilities. Furthermore, even though shipments are centrally planned, we have to account for store manager preferences, since the overall approach relies on their collaboration. Finally, operational constraints on shipments involve all of the SKUs, not a single one, and the total number of outbound shipments per store is constrained.

As a result of these differences, our model does not rely on integer variables, but on a huge number of binary variables, and features a different structure, more akin to a matching problem with side constraints. So, even the LP relaxation of the model may take hours to solve with a state-of-the-art solver, precluding the adoption of matheuristics like fix-and relax. Since we do want to take advantage of the power of state-of-the-art MILP solvers, we have to reduce the model size in a clever way, by eliminating decision variables *before* applying the LP-based solver. Hence, we have adopted a model reduction approach based on a sequence of easy weighted matching subproblems, as we explain in Section 5. It is interesting to observe that a similar strategy is applied, in a different context, by Morabit et al. (2022), who apply machine learning techniques to speed up the solution of the column generation subproblem in a routing application. They analyze data collected over successive iterations in order to reduce the set of arcs to consider when solving an NP–hard variant of a shortest path subproblem, which is the pricing component of their decomposition approach.

Another potential source of interesting ideas is stochastic dynamic programming. It is well-known that a literal application of the dynamic programming principle is usually not feasible, due to different curse of dimensionality (Brandimarte, 2021). Nevertheless, a wide range of approximation strategies is available to overcome the difficulty (Powell, 2011). Indeed, variations of dynamic programming strategies have been proposed to deal with lateral shipments. Meissner and Senicheva (2018) apply approximate dynamic programming, but they consider a single SKU and a smaller network than we consider here. A decomposition strategy is applied by Feng et al. (2017) for proactive lateral shipments, whereas Seidscher and Minner (2013) deal with both reactive and proactive shipments. These two references, too, deal with a single item.

Hence, the problem that we consider, as well as the modeling and solution approach, stand apart from other research work. This is due to the size of the problem, the interactions between different SKUs, and the need to involve store managers into the process, which is not purely centralized. Needless to say, this does not imply in any way that our proposal is better than the alternatives. Indeed, it is quite difficult to compare it in terms of classical costs of inventory control models. We can only claim that our modeling framework has some value in terms of flexibility and that the solution approach based on variable selection is of interest in itself.

4. Model formulation

Let us define the relevant sets and associate them with indexes:

- The set of SKU codes, indexed by $i \in \mathcal{I}$.
- The set of destination nodes (retail stores), indexed by $l \in \mathcal{S}$.
- The set of origin nodes, indexed by $k \in \mathcal{S}_0 = \mathcal{S} \cup \{0\}$; this set includes retail stores, as well as the central warehouse associated with 0.

As we have pointed out, we may have lateral shipments between retail stores, but the central warehouse is neither a transhipment node for lateral shipments nor the destination of a reverse flow.

The basic problem data are denoted as follows:

- O_{ik}: offered amount of SKU i at source node $k \in \mathcal{S}_0$.
- R_{il}: requested amount of SKU i at store $l \in \mathcal{S}$.
- B_k: upper bound on the number of outgoing shipments from retail store $k \in \mathcal{S}$.
- W: fixed transportation charge per shipment.
- V_i: value of SKU $i \in \mathcal{I}$.
- $V_{\min}$: minimum value for an acceptable shipment.

The most critical part in modeling this problem is the choice of decision variables. A seemingly obvious choice would be to introduce integer (due to small volumes) decision variables X_{ikl}, representing the amount of SKU i shipped from source node $k \in \mathcal{S}_0$ to retail store $l \in \mathcal{S}$. However, by doing so we would disregard some relevant facts.

- The utility of shipping an item must take into account the probability of actually selling it. As a result, the (expected) utility derived from shipping two items to a given store need not be twice as much as the utility of a single item. In fact, the probability that random demand D exceeds a given threshold x, $\mathbb{P}\{D > x\}$, is a decreasing function of x, which results in marginally decreasing utility coefficients.
- Suppose that two units of an SKU are available at the central warehouse, and two stores are requesting two (or more) units each. Let us further assume that no unit is available at the two stores and that the two probability distributions of demand are identical. In terms of shipped SKU value, there is no difference between allocating two units to either store or one to each store. However, in terms of expected utility, the second solution would be preferred. Actually, this also depends on the probability of selling the shipped items, which in turns depends on the stock on hand at the stores. Marginal utilities are quantified in order to account for all of these issues.
- Priorities may also be associated with specific stores (e.g., recently opened ones, where supporting sales has strategic value), and they are reflected by utility coefficients.
- It may be preferable to ship items from the warehouse, rather than a store, as this should improve chances of selling the overall available inventory.
- Franchised store managers may be particularly reluctant to release items available in their inventory. Moreover, there are cases in which some available items are not willingly released by the manager of a store owned by the firm with an inventory surplus, but they are nevertheless considered for an enforced lateral shipment and included as offered items. Since it is preferable to satisfy a request by a willing offer, this should be accounted for by a lower utility coefficient.

In order to avoid an overly complicated (nonlinear) objective function or a stochastic optimization problem, we might consider going to the opposite end of the spectrum and introduce *binary* decision variables X_{iklmn}. Subscripts i, k, and l refer to SKU code and source/destination nodes as before, respectively. Subscript $m \in \{1, 2, \ldots, O_{ik}\}$ refers to the item number (position) for SKU i at source node k, and $n \in \{1, 2, \ldots, R_{il}\}$ refers to the item number (position) for SKU i at destination node l. These decision variables should be multiplied by marginal utility coefficients U_{iklmn}. The precise meaning of subscripts m and n deserves some clarification. If, for instance, a retail store is offering two items of a given SKU, m would range from 1 to 2. Further suppose that the store manager is not quite willing to offer those items: he/she may be required to do that by the central planners for various reasons. In this case, being forced to release two items, rather than one, may lead to increased dissatisfaction, a disutility that can be reflected in the utility coefficients. The reduction of utility for item $m = 2$ is larger than the reduction for item $m = 1$. Subscript n plays a similar role when we consider a store requesting more than one item of a given SKU. The marginal utility is decreasing with respect to n, as the probability of selling additional items at a retail store is decreasing. Therefore, matching request $n = 1$, for a given SKU at a retail store, yields a larger marginal utility than matching request $n = 2$, and so on. A

Table 1
Table of offers/requests for a toy example. There are two retail stores (nodes 1 and 2), besides the central warehouse (node 0), and three SKUs (A, B, C).

Node	Offers	Requests
0	2 of A, 1 of B	–
1	1 of B	2 of A, 1 of C
2	1 of C	1 of A, 1 of B

possible exception is the case of a store manager who is requesting a reactive shipment to meet the realized demand by two customers who are willing to wait for the delivery of their preferred item. In this case, the selling probability would be 1 for both items, but the model can accommodate this case. In fact, we may deal with both reactive and proactive shipments.

Assuming that we may sensibly estimate the marginal utilities, a matter discussed later in Section 4.1, this choice of decision variables allows for a remarkable flexibility. Unfortunately, it also results in a computational nightmare, not only due to the sheer number of binary variables, but also due to potential issues with symmetry and alternative optimal solutions. Indeed, the most common case in which multiple items of the same SKU are available is the central warehouse, where such disutility considerations are not relevant. It is worth noting that well-known textbooks like (Williams, 2013) suggest using general integer variables, rather than binary ones, when possible. However, given the need to express marginally decreasing utilities, we would still need to introduce plenty of auxiliary variables to represent a piecewise linear utility.

A preliminary computational experience with this flexible modeling approach showed that real-life instances may be quite difficult to solve. Some problem instances required about 8 h only to solve the root LP relaxation, using state-of-the-art interior point methods. Therefore, we have adopted a compromise formulation, which may reduce the number of decision variables from possibly a few millions, in large-scale instances, to about one million, based on the following choice of core decision variables:

$$X_{ikln} = \begin{cases} 1 & \text{if node } k \in \mathcal{S}_0 \text{ sends an item of SKU } i \\ & \text{to satisfy request number } n \text{ for } i \text{ at store } l \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$$

These variables should be interpreted as matching variables (expressing the fact that a request is matched by an offer), rather than transportation variables, and they multiply corresponding marginal utility coefficients U_{ikln} (see Section 4.1). Actually, the matching variables X_{ikln} are only defined for quadruples (i, k, l, n) such that the source node k offers an SKU i that is requested by the destination node l. Let us denote the set of such quadruples by $\mathcal{M}$, the set of possible matchings. To clarify the meaning of these variables, a toy example of offers and requests is illustrated in Table 1, and the corresponding matching variables are visualized in Fig. 2. Note that in the case of SKU A, for which two items are available at the central warehouse, we do not specify which of the two units is used to match a request. Therefore, up to two arcs outgoing from node 0 may be selected (hence, we use the term *matching* in an arguably lose sense). On the contrary, two separate nodes are introduced to express the satisfaction of the two requests for SKU A by retail store 1, as different marginal utilities are associated with the two arcs.

We also need additional variables related to the use of a point-to-point arc for a shipment:

$$\delta_{kl} = \begin{cases} 1 & \text{if there is a shipment from node } k \in \mathcal{S}_0 \text{ to store } l \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$$

These variables are necessary to bound the number of outgoing shipments from a retail store, to account for fixed shipping charges, and to enforce a minimum shipped value (if really needed). They are not

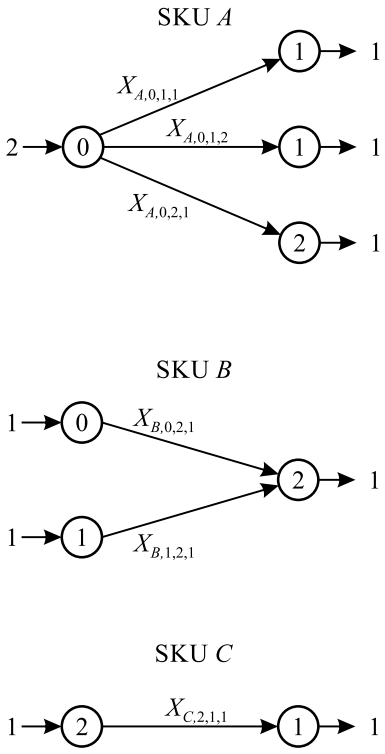


Fig. 2. The matching variables corresponding to the data of Table 1.

defined for every (source,destination) pair, but only for the subset of feasible shipping pairs, denoted by $\mathcal{P}$. This is just the projection of the set of matchings $\mathcal{M}$ on the set of (k, l) pairs for which there is at least a possible matching for an SKU.

Let us also define the set $\mathcal{O}_k \subseteq \mathcal{I}$ of SKUs offered by node $k \in \mathcal{S}_0$, as well as the set $\mathcal{R}_l \subseteq \mathcal{I}$ of SKUs requested by node $l \in \mathcal{S}$. Then, the optimization model may be stated as follows:

$$\max \quad \sum_{(i,k,l,n)\in\mathcal{M}} U_{ikln} X_{ikln} - W \sum_{(k,l)\in\mathcal{P}} \delta_{kl} \tag{1}$$

$$\text{s.t.} \quad \sum_{(l,n):(i,k,l,n)\in\mathcal{M}} X_{ikln} \le O_{ik}, \qquad k \in \mathcal{S}_0,\ i \in \mathcal{O}_k \tag{2}$$

$$\sum_{k:(i,k,l,n)\in\mathcal{M}} X_{ikln} \le 1, \qquad l \in \mathcal{S},\ i \in \mathcal{R}_l,\ n = 1, \ldots, R_{il} \tag{3}$$

$$X_{ikln} \le \delta_{kl}, \qquad (i,k,l,n) \in \mathcal{M} \tag{4}$$

$$\sum_{l:(k,l)\in\mathcal{P}} \delta_{kl} \le B_k, \qquad k \in \mathcal{S} \tag{5}$$

$$\sum_{(i,n):(i,k,l,n)\in\mathcal{M}} V_i X_{ikln} \ge V_{\min} \delta_{kl}, \qquad (k,l) \in \mathcal{P} \tag{6}$$

$$X_{ikln},\ \delta_{kl} \in \{0, 1\}. \tag{7}$$

The objective function (1) is the total expected utility associated with matchings, minus the sum of shipping charges associated with shipments from sources to destinations. Constraint (2) bounds the number of matchings out of a source node for each SKU by the number of offered items O_{ik}. Constraint (3) states that each single request, at each node, for every item of each SKU, can be matched at most once. There is a lack of symmetry between constraints (2) and (3), because the matching variables depend on the position number n on the destination side, but there is no corresponding position m on the source side (also see Fig. 2, as far as SKU A is concerned). As we have pointed out, this is a modeling choice that reduces model complexity and computational effort, while still allowing to express the decreasing marginal utility of a

matching when a destination requests more than one item for a given SKU. Constraint (4) links matching and shipment variables: we may have a matching between nodes k and l, for whatever item position of whatever SKU, only if there is a shipment from k to l. Constraint (5) is operational in nature and limits the number of shipments out of a source node $k \in S$ to B_k. There is no limit for node 0, the central warehouse. Constraint (6) states that if a shipment arc (k,l) is active, then the total value of the shipment must be at least $V_{\min}$. Note that here we use the item value, not the utility of a matching, and that this constraint is more operational than economic in nature: it may contribute to limit the shipping burden out of a store. Finally, we require that all of the decision variables are binary. In real-life instances, this model may include about 1 million binary decision variables, and just solving the LP relaxation, by a state-of-the-art solver like Gurobi, may be quite expensive and take hours.

4.1. Setting the marginal utility coefficients

The most critical parameters of the proposed model are the coefficients U_{ikln} multiplying the matching variables X_{ikln} in the objective function (1). We refer to such coefficients as *marginal utilities*, or *utilities* for the sake of brevity. The utility of matching the request by retail store $l \in S$ for item number $n \in \{1,\dots,R_{il}\}$ of SKU $i \in \mathcal{I}$ with an item available at source node $k \in S_0$ depends on several considerations. Here we describe *one* possible approach to set marginal utilities. This is the approach that has been used in the computational experiments of Section 6, but we stress the fact that the modeling framework and the solution strategy are independent of this choice.

As far as the probability of selling that item is concerned, we rely on the following steps.

1. A forecasting model for weekly demand is used, based on the disaggregation of aggregate forecasts to store/reference/size level and integrated with information from store managers; we refer to Sirovich et al. (2018) for details.
2. Demand is assumed to be a Poisson random variable, with an expected value corresponding to the demand forecast obtained at the previous step.
3. The probability $\hat{\pi}_{iln}$ of selling each requested item n of SKU i at store l is evaluated according to the Poisson distribution, accounting for current on-hand inventory.
4. The expected contribution to revenue is estimated as $V_i \cdot \hat{\pi}_{iln}$.

In order to determine the utility coefficients, suitable corrections are applied to account for policy rules and preferences, in the form of multiplicative factors, which can be larger or smaller than 1, resulting in an increase or decrease of utility:

- Utility is increased for items released from central inventory, and it is decreased for items released from a franchised store.
- Utility is also decreased if the release is enforced.
- On the destination side, utility is increased for strategic stores.

Additional considerations could play a role in the quantification of marginal utilities, and this is related with the specific features of the business process we aim to support. For instance, we may deal with both proactive and reactive lateral shipments, related to demand forecasts or realized demand of patient customers, respectively. The model is flexible and agnostic with respect to such features.

5. The solution approach

Due to the computational burden of solving the proposed model exactly, a heuristic approach is needed. As we have pointed out, the size of the problem discourages a local search metaheuristic strategy, and the difficulty in solving the LP relaxation discourages a matheuristic based on the full MILP model. Hence, the approach we have pursued relies on a model reduction strategy, before applying commercial branch-and-cut (possibly within a suboptimality tolerance). The key observation stems from constraint (4), which links the matching variables X_{ikln} with the shipping variables δ_{kl}. When a shipping variable is set to 0, a large number of matching variables is forced to 0. If we could reduce the set of shipping pairs from the set of feasible shipping pairs $\mathcal{P}$ to a subset $\mathcal{P}_*$ of really useful pairs, the overall problem size would be considerably reduced. To get a quantitative feeling, we observe that the overall number of binary variables in the model may be huge, but the number of decision variables related to pairs of arcs is quite reasonable. If we consider 200 stores, there might be about 40,000 possible lateral arcs (actually less, as some offers and requests do not match for a pair of stores, or the shipped value would be too small). In our experience, the number of outgoing shipments per store is bounded between 5 and 10; hence, if we allow 8 outgoing shipments per store on the average, at most 3,200 lateral shipments can be carried out. This quick calculation suggests that indeed, by setting a δ_{kl} variable to zero, preventing a shipment from node k to store l, we may eliminate a significant number of matching variables, drastically reducing the problem size without degrading the solution too much. We do so only for store-to-store shipments. Then we introduce all of the potential arcs from the central warehouse into the reduced model, as their number is rather small (200 in the case of 200 retail stores).

In order to assess the value of a shipping pair (k,l) (from source node $k \in S$ to destination node $l \in S$) we may consider the total item value

$$V_{kl} = \sum_{i \in \mathcal{I}} V_i \cdot \min\{O_{ik}, R_{il}\}, \qquad k \neq l. \tag{8}$$

Let us denote by $N_{\max}$ the maximum number of shipping pairs that we want to include in the reduced model. A very simple algorithm would sort feasible shipments in decreasing order of their value, pick the best one, update offers and requests of the involved stores, update the ranking, and repeat until either no shipment is feasible or we exceed the maximum threshold $N_{\max}$. In the procedure, we should also keep track of the allowable outgoing shipments for each store, and eliminate a source node when its limit has been reached. The idea is quite simple, but overly greedy.

In order to reduce the amount of myopia in the previous greedy strategy, we may look for the best subset of *independent* shipping pairs, where "independent" means that the pairs do not interfere with each other, as each node may play the role of the source in at most one shipment, and the role of the destination in at most one shipment. This may be found by solving a maximum-weight matching problem, where the weight of a pair is given by Eq. (8):

$$\begin{aligned}
(\mathcal{M}) \quad \max \quad & \sum_{(k,l)\in\mathcal{P}} V_{kl}\delta_{kl} \\
\text{s.t.} \quad & \sum_{l:(k,l)\in\mathcal{P}} \delta_{kl} \le 1, \qquad k \in S \\
& \sum_{k:(k,l)\in\mathcal{P}} \delta_{kl} \le 1, \qquad l \in S \\
& \delta_{kl} \in \{0,1\},
\end{aligned}$$

where $\mathcal{P}$ is the set of feasible shipping pairs. Note that at this level we are matching pairs of stores, rather than single offers and requests. By solving this (easy) problem, we will obtain a collection of N pairs. However, not all of them are necessarily worth including in the reduced model. The most valuable matchings are probably good candidates for inclusion in the overall solution. However, the same does not necessarily apply to the lowest value matchings; by committing items to them, we might preclude better shipments. Therefore, we introduce a hyperparameter α, representing the fraction of shipping pairs in the optimal matching that are introduced in the reduced model. Hence, after solving a matching problem, we rank the pairs according to their shipment value and select the top $\lceil \alpha N \rceil$ ones for inclusion as eligible shipping pairs in the reduced model. A small value of α increases the number of matching subproblems to be solved, but may avoid

Table 2
Descriptive statistics of the problem instance features.

	Stores	SKU	Offers	Requests	Store pairs	Matchings
Min	30.0	118	290	475	132	631
Q1	197.8	1067	5328	7383	2110	20,697
Median	215.5	1738	11,773	13,958	8214	48,520
Mean	204.3	1787	13,384	16,370	17,266	179,293
Q3	247.0	2260	19,109	23,222	31,933	291,454
Max	259.0	4239	38,729	50,518	60,137	887,079

Algorithm 1 Model reduction based on maximum-weight matching

Set the maximum number of shipping pairs $N_{\max}$ and the fraction of accepted pairs α.
Set the termination flag stop $\leftarrow$ false and initialize the set of eligible pairs $\mathcal{E} \leftarrow \emptyset$.
For each shipping pair in the feasible set $\mathcal{F}$ set the shipment value according to Eq. (8).
while not stop **do**
  Solve the maximum-weight problem ($\mathcal{M}$). Let N be the number of pairs in the selected pairs.
  If $N = 0$ exit from the while loop.
  Sort the selected pairs in decreasing order of their value.
  for $k = 1 : \max\{\lceil \alpha N \rceil, N_{\max}\}$ **do**
    Include the pair in position k to the set $\mathcal{E}$, if it is feasible (i.e., it does not violate the maximum number of outbound shipments from the source node and its value is larger than the minimum threshold).
    Update offers and requests at source and destination nodes, respectively.
    Reduce $N_{\max}$ and the number of allowed outbound shipments from the destination node.
    If $N_{\max} = 0$, then stop $\leftarrow$ true and exit from the for loop.
  end for
end while

myopic selections. In order to solve a new matching subproblem, we have to update offers and request at nodes introduced in the subset of eligible shipping pairs, as well as reducing the number of possible outgoing shipments for each node. We keep adding eligible pairs until we exceed the upper threshold $N_{\max}$. The model reduction approach is more formally specified as a high-level pseudo-code in Algorithm 1.

There are possible variations on this basic procedure:

1. Rather than using shipment values, we may evaluate the suitability of a shipping pair by using the utility coefficients. On the one hand, this looks more consistent with the actual optimization model. On the other hand, however, we have to update utility-based matching weights whenever we update offers and requests, following the sequence of selected pairs. This is not necessarily consistent with the overall reduced model, where we also consider shipments from the central warehouse.
2. We may also introduce a second control parameter, denoted by β, which is an allowance on the number of outgoing shipments from each store. Hence, the upper bound for store k may be relaxed to $B_k + \beta$. The aim of this parameter is to leave more freedom to arrange shipments in the reduced model; however, it may result in a sensible increase in computational effort.

6. Computational testing

6.1. The data set

In order to test the performance of the proposed matheuristic strategy, we have collected 76 problem instances. It is a limited number of instances, but they are significant, as they consist of real data. Due to confidentiality reasons, we cannot report economic data, but Table 2 reports summary statistics for the following features:

- Stores: the number of stores involved in the inventory reallocation (not including the central warehouse).
- SKU: the number of SKUs.
- Offers: the number of offers (in terms of SKUs offered by any node, not number of items).
- Requests: the number of requests.
- Store pairs: the number of store pairs that could exchange items, as there is a match in terms of SKUs offered and requested. This does not include the central warehouse, which may contribute at most a number of shipments corresponding to the number of stores. Note that pairs are ordered: in principle, store A could both send items to and receive items from store B, but they are separate shipments.
- Matchings: the number of matchings at the item level (i.e., the number of binary matching variables).

We report the usual summary statistics (minimum and maximum value, the three quartiles, and the mean). Clearly, there is a mix of small and large scale problems, as sometimes the reallocation procedure is run for a limited subset of nodes in the network. Hence, we may have to solve both easy and hard problems.

The upper bound on the outgoing shipments from stores has been set to 5 for franchised stores, and 10 for owned stores, in order to reflect the respective peculiarities. As we have pointed out, this choice is not necessarily related to hard costs, but rather to intangible issues related to the inconvenience perceived by store managers when having to prepare too many parcels. We remark again that involving store managers is an essential requirement in our application. Moreover, based on preliminary experience on the available problem instances, we have decided to relax the constraint on the minimum shipment value. In practice, this constraint increased the CPU time without having a significant impact on the solution quality. To understand the reason, we may observe that if the number of feasible outgoing shipments from a node is large with respect to the maximum number allowed, uninteresting shipments will not be considered by the model anyway. If it is small, we may just post-process the solution eliminating small-value shipments without really degrading solution quality. However, this is specific of our computational experiments. In the actual use of the model, the constraint on minimum shipped value has been sometimes enforced. The fixed shipping charge is set to a relatively small value, based on data provided by Miroglio (but again, this only refers to this specific testbed).

6.2. Solution strategies

We compare the exact solver with the two versions of matheuristic: the one in which the matching subproblems are solved using the item values to define weights, and the one in which we use expected utilities. To be more precise, let us define the algorithmics settings:

- When using straightforward branch and cut, we leave Gurobi (version 9.5.2) to its default settings, with the only exception of the suboptimality tolerance, which is set to 1% to cut CPU time without degrading too much solution quality. The root node LP relaxation is solved in parallel by the interior method, primal simplex, and dual simplex. The first thread to complete the LP relaxation stops the process. The experiments were run on a HP

Table 3
Comparing expected utility across solvers.

	Exact	Val1000	Val2000	Val3000	Util1000	Util2000	Util3000
Min	15,980	15,808	15,808	15,808	15,573	15,573	15,575
Q1	267,913	262,308	263,955	264,422	261,834	263,457	263,615
Median	436,359	430,568	448,914	442,164	428,722	447,171	443,546
Mean	429,251	474,221	497,881	494,643	473,237	496,355	495,165
Q3	575,322	652,920	669,501	668,425	648,671	669,520	664,902
Max	1,274,560	1,600,711	1,718,857	1,726,396	1,592,015	1,710,087	1,724,039

Table 4
Comparing CPU time (seconds) for branch and cut across solvers.

	Exact	Val1000	Val2000	Val3000	Util1000	Util2000	Util3000
Min	0.006	0.004	0.004	0.004	0.004	0.005	0.004
Q1	1.246	0.198	0.336	0.342	0.176	0.303	0.329
Median	8189.284	0.628	2.629	2.592	0.552	2.211	2.337
Mean	5531.865	1.163	828.629	1434.037	0.789	528.597	1262.690
Q3	10,800.421	1.437	568.01	3600.210	1.205	163.480	3600.224
Max	10,805.204	11.012	3601.086	3601.576	2.809	3605.686	3605.479
Limit	49%	0%	20%	38%	0%	9%	32%

ZBook 15 G6 Mobile Workstation, equipped with an Intel Core i9-9880H 2.30 GHz processor. Of the 8 thread available on the logical cores, 7 were allocated to Gurobi. We set a time limit of 10,800 s (i.e., three hours). This approach is referred to as `Exact`, even though it is not really exact due to the relative suboptimality tolerance.

- When applying the matheuristic, we must specify some hyperparameters:
  - The maximum number of shipping pairs (between stores) $N_{\max}$ has been set to 1000, 2000, and 3000. Since we may define weights by value or by utility, we obtain six heuristic solutions strategies: `Val1000`, `Val2000`, `Val3000`, `Util1000`, `Util2000`, and `Util3000`. In the following, we will refer to both exact and heuristic solution strategies as *solvers*, for the sake of brevity.
  - When solving the restricted problem by branch and cut, we use similar settings as in the `Exact` case: suboptimality tolerance of 1%, but CPU time limit of 3600 s (1 h). Note that this only refers to the branch and cut phase, and it does not include the time to solve the sequence of matching subproblems, as well as the time to build the problem instance for Gurobi using the Python interface (which may be substantial, but is related to implementation details, rather than the solution algorithm). Hence, the allocation of time to the different solvers may be considered substantially fair.
  - The fraction of accepted pairs α (when solving each maximum weight matching subproblem) has been set to 0.5.
  - In the model reduction phase, we do not allow more outgoing shipments than the limit per store (i.e., we set β to 0).

These settings are the result of preliminary computational experience with this specific testbed, but they are not necessarily the best ones in general.

6.3. Aggregate analysis

To get a first visual and numerical feeling for the relative performances of the seven solvers, we report descriptive statistics for the objective value in Table 3, and for the branch and cut times (in seconds) in Table 4. We note that no economic information can be associated with the objective values, as they are related to an estimate of expected utility. The same information is depicted in boxplot form in Figs. 3 and 4, respectively.

The differences between means and medians suggest that data are skewed, especially CPU times. Clearly, considering means and medians of expected utilities is a crude analysis, since we are mixing quite different problem instances. Nevertheless, some interesting patterns emerge. If we look at medians in Table 3, we observe that the best performance seems to be obtained by including about 2000 shipments among stores in the reduced model, as is done with the `Val2000` and `Util2000` solvers. Adding more of them looks counterproductive, which is actually an effect of the CPU time limit. In the last line of Table 4 we report the fraction of problem instances for which the time limit was reached. In almost 50% of the cases, branch and cut applied to the full model is stopped by the time limit. It is also worth noting that quite often a very limited number of branch and cut nodes is explored, since most of the work is carried out at the root node, solving the LP relaxation, running heuristics, and generating cuts. This never happens for the case of 1000 shipping pairs, at the cost of a lower solution quality. Increasing the shipping pairs to 2000 or 3000 has a significant impact on computational effort, as it can be seen from the mean of CPU times and in the fraction of unsolved problem instances. If we consider the mean of expected utilities, we still see that solvers with 1000 pairs are outperformed, whereas solvers with 2000 and 3000 pairs seem comparable in terms of objective function (with a considerable increase in computational effort in the latter case). This is not completely in line with what we observe in terms of medians, but it is due to the skew in the data.

This preliminary analysis suggests that we should disaggregate the analysis with respect to problem size, and that we might focus on comparing `Exact` with `Val2000` and `Util2000`, which is what we do in the next section.

6.4. Disaggregation of easy and hard problems

In order to split the problem instances in two classes, we use the median number of possible shipments between stores, which is 8214, as reported in Table 2. Thus, the 76 problem instances are split into two classes of 38 instances each.

Figs. 5 and 6 report boxplots for expected utility and CPU time, respectively, for instances below the median. The message is quite clear. Apart from some outliers, the computational effort for the full model is pretty small, and there is no reason to deteriorate solution by resorting to the matheuristics based on model reduction (even though they perform fairly well). This is not quite surprising, given the limited size of this subset of problem instances, which are easy, yet realistic.

A completely different picture emerges from Figs. 7 and 8, which refer to problem instances above the median. In this case, even `Val1000` and `Util1000` outperform `Exact`, due to the time limit. Increasing the pairs from 2000 to 3000 has a significant impact on computational effort, which is not justified in terms of solution quality. Using weights

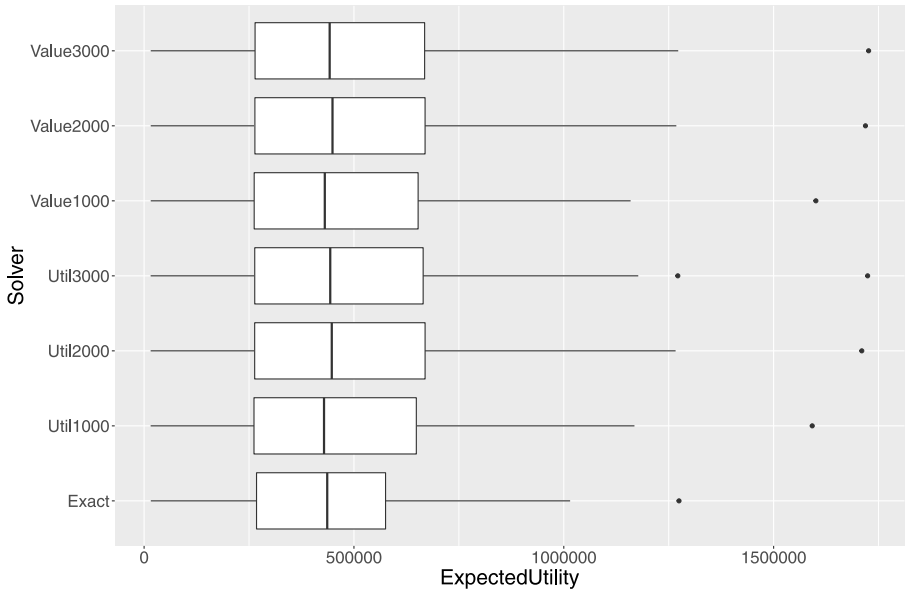


Fig. 3. A boxplot representation of expected utility across solvers.

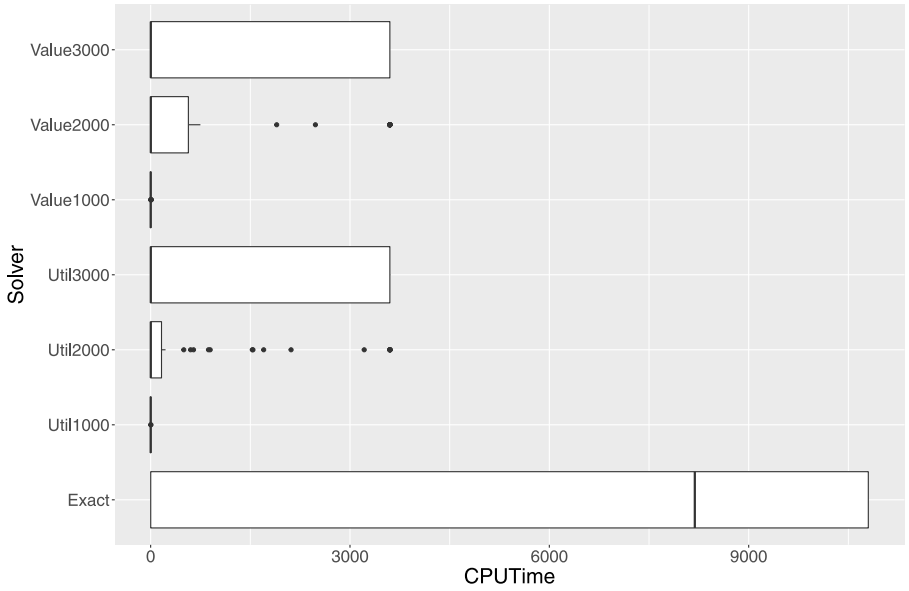


Fig. 4. A boxplot representation of CPU time (seconds) for branch and cut across solvers.

based on utilities, rather than plain values, seems to be beneficial in terms of computational effort, arguably because it results in smaller models.

This suggests drawing a more detailed picture for the 38 more difficult problem instances, which is reported in Table 5. In order to make the pattern clearer, we give relative ratios, i.e., the ratio of the expected utility obtained by each solver with the best solution among the three of them on each instance. Thus, 100% corresponds to the best performance, and the fractions measure the relative quality of the solution provided by the three solvers on each instance. The rows have been sorted in increasing order with respect to the ratio for `Val2000`. This solver emerges as the best one in terms of solution quality. Apart from the first seven instances, where `Exact` is the best performer and the matheuristics fall short of obtaining the optimal value (within the relative suboptimality tolerance of 1%), `Val2000` is always the best performer, with a small advantage with respect to `Util2000`. As we have observed, the latter solver results in smaller CPU times, though. The tradeoff between solution quality and computational effort, in practical terms, can be a matter of debate. The inventory reallocation problem need not be solved in real time, and increasing CPU times by a few minutes might be justified, if this results in better quality of solutions. We should however observe that quality is not really measured in terms of objective monetary value, but in terms of expected utilities that are just estimated by a sensible but somewhat arbitrary procedure. Hence, the improvement in terms of expected utility obtained by `Val2000` with respect to `Util2000` is questionable. What is evident, though, is that if we do not reduce the model size by a suitable procedure, poor solutions are typically obtained within the allocated CPU time limit.

6.5. A further experiment with extended CPU time allowance

One might wonder whether the time limit enforced on the `Exact` solver is restrictive, and whether this invalidates any conclusion about

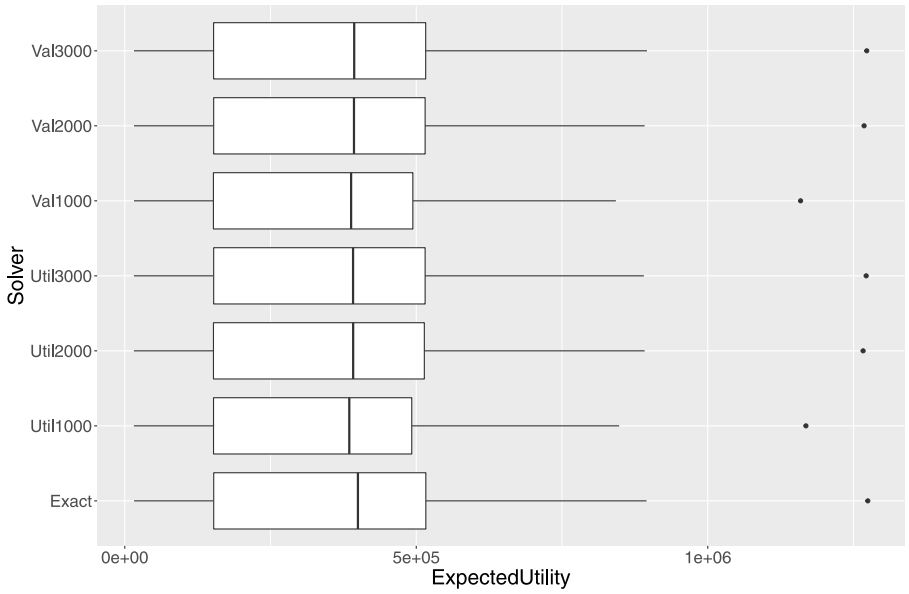


Fig. 5. Boxplot representation of expected utilities for instances below the median.

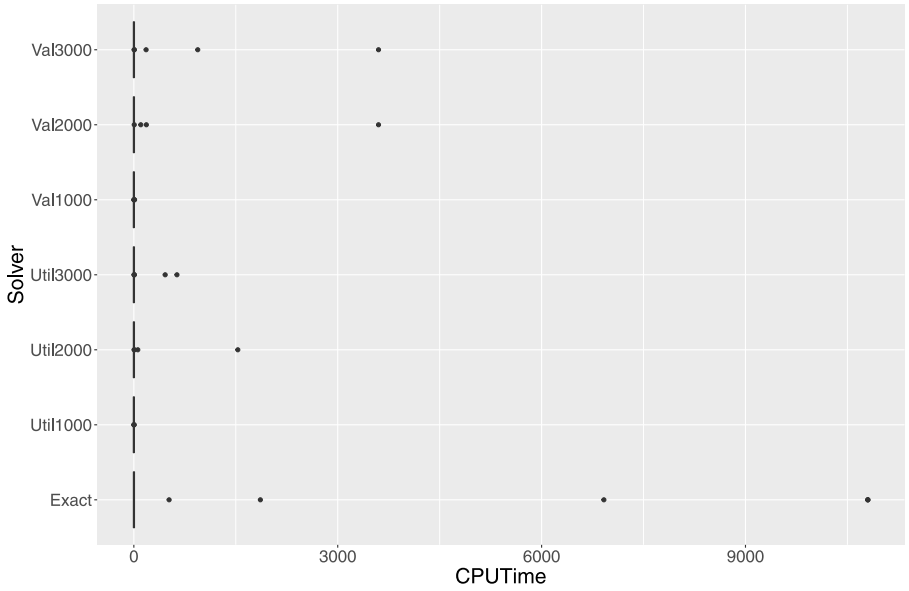


Fig. 6. Boxplot representation of computational effort for instances below the median.

the quality of the heuristic solution. To run a further check we reconsidered the 17 problem instances for which the gap between the objective of the incumbent solution and the best bound was above 30%. For the sake of illustration, here are the gaps when the time limit stopped branch-and-cut:

97.1%, 55.3%, 55.1%, 42.5%, 40.6%, 38.2%, 36.7%, 35.3%, 35.2%,

35.2%, 34.7%, 33.1%, 32.7%, 31.2%, 30.8%, 30.6%, 30.1%.

We started branch-and-cut by setting the cutoff value to the value provided by the `Val2000` solver, using again a relative gap of 1%. In other words, we asked the exact algorithm to improve the heuristic solution by at least 1%, and we allowed 24 h to do so. In no case a better solution was reported.

We cannot claim that this really certifies the quality of the heuristic solution, but at least it shows that the three hours time limit was not a restrictive choice, and that the model may indeed result in very difficult BIP problems. We cannot rule out the possibility that an alternative approach, possibly a metaheuristic, could do better. However, we want to emphasize that a matheuristic approach, relying on a reduced formal optimization model, may be more flexible when additional restrictions are enforced (which did happen, as discussed in the conclusions). Moreover, it allows us to leverage any improvement in commercial solvers without writing a single line of code.

7. Conclusions and directions for further research

Both the optimization model and the matheuristic solution approach that we have described in the paper have been implemented in a module within an overall inventory management solution devised by Evo for Miroglio Fashion. The system is actually in use by Miroglio to manage the stores inventory of the four main brands of the company and the corresponding outlet stores. Here we discuss academic viewpoint separately from the business impact.

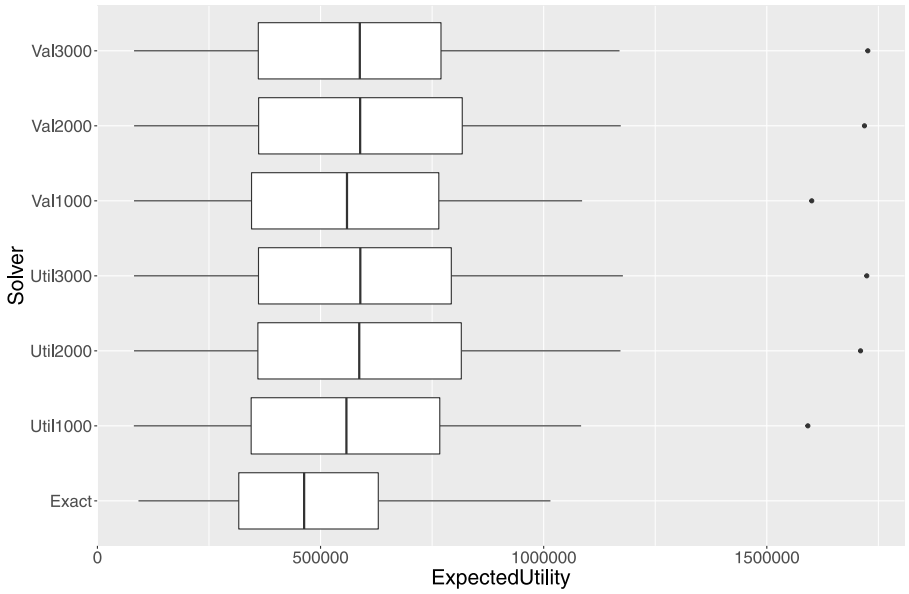


Fig. 7. Boxplot representation of expected utilities for instances above the median.

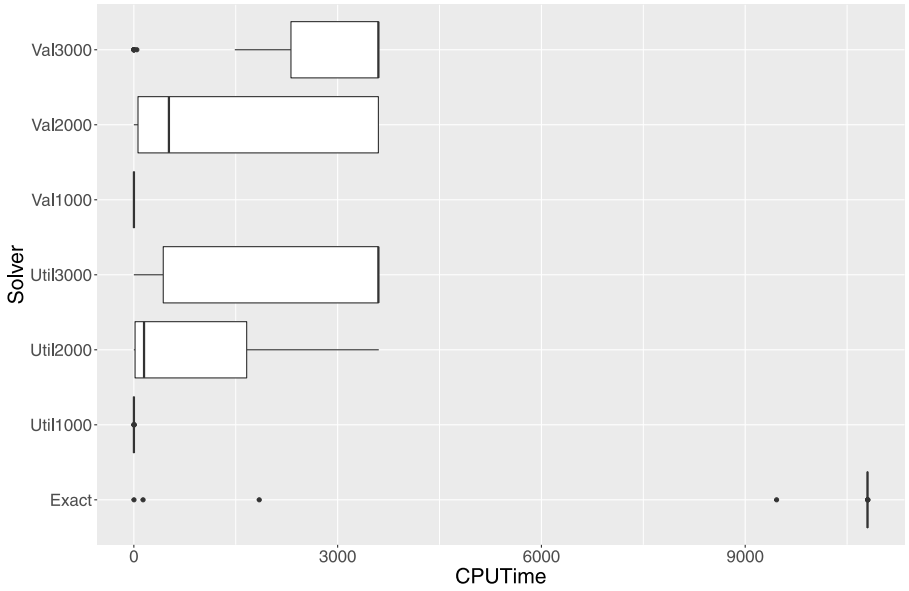


Fig. 8. Boxplot representation of computational effort for instances above the median.

7.1. Methodological contribution

The model that we have proposed is not meant to be a substitute for classical approaches to lateral shipments, based on stochastic inventory control theory. Rather, it is a complement aimed at dealing with many facets of the problem, which do not lend themselves to traditional stochastic modeling, as well as approximate dynamic programming. This is partly due to the sheer size of the problem, in terms of number of stores and SKUs, and partly to the need to involve store managers into the process, which implies some intangible aspects due to their incentives and knowledge about their specific store location.

The model is remarkably flexible and it can be integrated within a decision architecture supporting a range of different business processes. Unfortunately, it requires the solution of a very difficult pure binary linear program. The LP relaxation itself is computationally difficult, which precludes the application of a whole range of matheuristics based on solving a sequence of LPs or restricted BIPs (Maniezzo et al., 2021). We have resorted to a model reduction strategy, whose performance has been analyzed by experiments on real-life instances.

Both the experience that we have described and the actual use have shown that the model reduction approach has merit. However, there are some open issues. For instance, a moderate increase in the number of possible shipping pairs included in the reduced model may result in an abrupt increase in the CPU time. This is somewhat related to performance variability, which is a well-known issue in MILP models; see, e.g., Lodi and Tramontani (2014) for a related discussion. Also the choice of other hyperparameters, like the fraction of selected pairs among those resulting from solving a weighted matching problem is subject to discussion. Moreover, we have set the allowance parameter β to zero, as a strictly positive value resulted in a sharp increase in computational effort. However, a better strategy would be to allow more shipping pairs from a carefully selected subset of stores, leaving the final selection to the solution of the reduced model. From a very practical viewpoint, a safe strategy is to solve the model with a conservative setting of parameters, in order to obtain a good solution

Table 5
Relative performance of `Exact`, `Val2000`, and `Util2000` in terms of expected utility on instances above median.

#	Exact	Util2000	Val2000	#	Exact	Util2000	Val2000
1	100%	88%	89%	20	80%	99%	100%
2	100%	89%	89%	21	71%	99%	100%
3	100%	92%	93%	22	56%	99%	100%
4	100%	94%	95%	23	97%	100%	100%
5	100%	95%	95%	24	79%	100%	100%
6	100%	95%	96%	25	94%	100%	100%
7	100%	98%	98%	26	49%	100%	100%
8	67%	100%	100%	27	80%	100%	100%
9	100%	99%	100%	28	80%	100%	100%
10	77%	100%	100%	29	81%	100%	100%
11	96%	99%	100%	30	47%	100%	100%
12	95%	99%	100%	31	83%	100%	100%
13	99%	99%	100%	32	73%	100%	100%
14	73%	99%	100%	33	75%	100%	100%
15	74%	99%	100%	34	82%	100%	100%
16	84%	99%	100%	35	77%	100%	100%
17	77%	99%	100%	36	83%	100%	100%
18	75%	99%	100%	37	83%	100%	100%
19	83%	99%	100%	38	87%	100%	100%

with a limited computational effort. Then, since the problem need not be solved in real time, it is possible to start another computation to see if a better solution can be found within a predefined time window. Needless to say, this is a practical and sensible workaround, but it is neither elegant nor quite satisfactory. A promising approach is to apply machine learning techniques in order to find a suitable set of problem features, besides the sheer size of the specific problem instance, and to figure out a way to automate the choice of algorithm hyperparameters.

Last but not least, one might wonder whether alternative solution strategies, like metaheuristics not relying on a formal optimization model, could outperform the proposed approach. Clearly, despite the large-scale nature of the problem, we cannot rule out this possibility. Nevertheless, we argue that a matheuristic approach may entail two advantages:

1. It may be easier to incorporate additional restrictions, which may occur when the model is applied to a different firm within the same industry (as we discuss below).
2. By relying on commercial solvers for mathematical programming we may leverage improvements in performance without the need to adapt code.

7.2. Business impact

The model discussed in this paper is one component of a replenishment tool developed by Evo for the Miroglio application. During its 5+ years in operation, the tool has already managed millions of pieces of inventory. The introduction of the Evo Replenish tool, which includes the matheuristic approach, has allowed the firm to achieve €1 million/month incremental margin with a reduction in leftover inventory and 23% fewer stockouts. Clearly, we cannot claim that such a reduction can be attributed to the rebalancing module. It is impossible to disentangle the contributions of each individual module of the overall architecture. Nevertheless, the flexibility of the model played a significant role in user acceptance, due to its ability to accommodate inputs from store managers, including some of their preferences.

Besides Miroglio, the model has been successfully adopted also by other companies in fashion and other industries to improve inventory management. Since this also raised the need for model extensions, new features and constraints have been added, at the request of the companies themselves, in order to respond to different logistic needs. The main changes have been the following:

- New logistic constraints have been introduced, besides the minimum value of each shipment. For each shipment it is possible to specify minimum and maximum value, number of pieces and physical weight.
- The shipment cost, instead of a fixed parameter, can depend on the geographical location of source and destination nodes.
- The concept of *bundle shipment* has been introduced. Some items can feature a special packaging, in which more pieces of the same SKU are packed together, or even different sizes of the same item. These bundles are usually available only at the central warehouse, and they have to be shipped as a single unit. Hence, for these items, we need to handle some extra constraints to ship a number of pieces that is a multiple of the bundle size. For example, if five pieces of an SKU are packed together, we can either decide to ship nothing, or one bundle (5 pieces), or two bundles (10 pieces), and so on.

The flexibility of the modeling framework has allowed a relatively easy adaptation of the initial tool.

CRediT authorship contribution statement

Paolo Brandimarte: Conceptualization, Formal analysis, Methodology, Software, Writing – original draft, Writing – review & editing. **Giuseppe Craparotta:** Conceptualization, Data curation, Formal analysis, Writing – review & editing. **Elena Marocco:** Conceptualization, Data curation, Formal analysis, Writing – review & editing.

Acknowledgments

We thank Francesco Cavarero and Miroglio Group for the support during the development and the actual application of the model and the related tools, and Fabrizio Fantini, founder of Evo Europe Limited, for creating the connection. We also thank the anonymous reviewers for their useful comments, which lead to a significant improvement in the paper.

References

Agrawal, V., Chao, X., & Seshadri, S. (2004). Dynamic balancing of inventory in supply chains. *European Journal of Operational Research*, *159*, 296–317.

Agrawal, N., & Smith, S. (2009). Multi-location models for retail supply chain management. In N. Agrawal, & S. Smith (Eds.), *Retail supply chain management: Quantitative models and empirical studies* (2 ed.). (pp. 319–347). Springer.

Allen, S. (1958). Redistribution of total stock over several user locations. *Naval Research Logistics Quarterly*, *5*, 51–59.

Allen, S. (1961). A redistribution model with set-up charge. *Management Science*, *8*, 99–108.

Amrani, H., & Khmelnitsky, E. (2017). Optimal division of inventory between depot and bases. *Naval Research Logistics*, *64*, 3–18.

Bertazzi, L., & Speranza, M. G. (2012). Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, *1*, 307–326.

Brandimarte, P. (2021). *From shortest paths to reinforcement learning: A MATLAB-based tutorial on dynamic programming*. Springer.

Brandimarte, P., & Zotteri, G. (2007). *Introduction to distribution logistics*. Wiley.

Caro, F., & Gallien, J. (2010). Inventory management of a fast-fashion retail network. *Operations Research*, *58*, 257–273.

Caro, F., Gallien, J., Díaz, M., García, J., Corredoira, J. M., Montes, M., Ramos, J. A., & Correa, J. (2010). Zara uses operations research to reengineer its global distribution process. *Interfaces*, *40*, 71–84.

Choi, T. M. (2014). *Fashion retail supply chain management: A systems optimization approach*. CRC Press.

Cordeau, J. F., Laganà, D., Musmanno, R., & Vocaturo, F. (2015). A decomposition-based heuristic for the multiple-product inventory-routing problem. *Computers & Operations Research*, *55*, 153–166.

Feng, P., Fung, R. Y., & Wu, F. (2017). Preventive transshipment decisions in a multi-location inventory system with dynamic approach. *Computers & Industrial Engineering*, *104*, 1–8.

Fisher, M., Hammond, J., Obermeyer, W., & Raman, A. (1994). Making supply meet demand in an uncertain world. *Harvard Business Review*, 83–93.

Fisher, M., Rajaram, K., & Raman, A. (2001). Optimizing inventory replenishment of retail fashion products. *Manufacturing and Service Operations Management*, *3*, 230–241.

Fisher, M., & Raman, A. (1996). Reducing the cost of demand uncertainty through accurate response to early sales. *Operations Research*, *44*, 87–99.

Grahovac, J., & Chakravarty, A. (2001). Sharing and lateral transshipment of inventory in a supply chain with expensive low-demand items. *Management Science*, *47*, 579–594.

Gupta, S. (2019). Miroglio fashion (B). HBS Business Case 9-519-070. https://hbsp.harvard.edu/product/519070-PDF-ENG.

Gupta, S. (2019). Miroglio fashion (C). HBS Business Case 9-519-072. https://hbsp.harvard.edu/product/519072-PDF-ENG.

Gupta, S., & Lane, D. (2019). Miroglio fashion (A). HBS Business Case 9-519-053. https://hbsp.harvard.edu/product/519053-PDF-ENG.

Lodi, A., & Tramontani, A. (2014). Performance variability in mixed-integer programming. *INFORMS Tutorials in Operations Research*, 1–12. http://dx.doi.org/10.1287/educ.2013.0112.

Maniezzo, V., Boschetti, M., & Stützle, T. (2021). *Matheuristics: algorithms and implementations*. Springer.

Meissner, J., & Senicheva, O. V. (2018). Approximate dynamic programming for lateral transshipment problems in multi-location inventory systems. *European Journal of Production Research*, *265*, 49–64.

Morabit, M., Desaulniers, G., & Lodi, A. (2022). Machine-learning–based arc selection for constrained shortest path problems in column generation. *INFORMS Journal on Optimization*, http://dx.doi.org/10.1287/ijoo.2022.0082.

Paterson, C., Kiesmüller, G., Teunter, R., & Glazebrook, K. (2011). Inventory models with lateral transshipments: A review. *European Journal of Operational Research*, *210*, 125–136.

Powell, W. B. (2011). *Approximate dynamic programming: solving the curses of dimensionality* (2nd ed.). John Wiley & Sons, Inc..

Seidscher, A., & Minner, S. (2013). A Semi-Markov decision problem for proactive and reactive transshipments between multiple warehouses. *European Journal of Operational Research*, *230*, 42–52.

Sirovich, R., Craparotta, G., & Marocco, E. (2018). An intelligent fashion replenishment system based on data analytics and expert judgment. In S. Thomassey, & X. Zeng (Eds.), *Artificial Intelligence for Fashion Industry in the Big Data Era* (pp. 173–195). Singapore: Springer, http://dx.doi.org/10.1007/978-981-13-0080-6_9.

van Wijk, A., Adan, I., & van Houtum, G. (2019). Optimal lateral transshipment policies for a two location inventory problem with multiple demand classes. *European Journal of Operational Research*, *272*, 481–495.

Wang, S., Zhang, H., Chu, F., & Yu, L. (2022). A relax-and-fix method for clothes inventory balancing scheduling problem. *International Journal of Production Research*, http://dx.doi.org/10.1080/00207543.2022.2145517.

Williams, H. (2013). *Model building in mathematical programming* (5th ed.). Wiley.