POLITECNICO DI TORINO

College of Engineering and Management

Master of Science in Industrial Engineering

Master of Science Thesis

Optimizing the reactive allocation of garments to stores in the fashion industry

Adviser:
Prof. Federico Della Croce Di Dojola

Co-adviser:

Prof. Giuliana Carello, Politecnico di Milano

Candidate:
Eleonora Acero

December 2018

SUMMARY

Executive Summary

This thesis addresses the problem of allocating scarce resources among several locations in the most efficient way in a fashion retail environment. This allocation is called “reactive” because it occurs in the second half of the selling season in response to the actual demand faced by the stores in the network, to rebalance the items in the system and increase the expected profit of the stores in the upcoming period.

Indeed, the Fashion and Apparel Industry is one of the pillars of the global economy, but it also represents one of the most flexible and unpredictable Industries, given the high volatility of demand and fast changes in customer tastes and trends, together with short products life cycles and long production lead times.

The top priority for almost all garment businesses is ensuring availability of products and, consequently, maintaining high customer satisfaction and brand image, while minimizing remaindered end of season stock and maximizing margins. Therefore, the most critical process for a company operating in this sector is the definition of the times and quantities to be allocated to the whole network of stores.

However, due to the inherent uncertainty of demand forecasts, it is likely that the firm will end up with some stores selling more than expected and others less. The two situations entail two different risks: where there has been an over-allocation the firm needs to markdown – loosing margins-, and where too little has been allocated it will lose sales – losing potential profits and reducing customers’ satisfaction.

Furthermore, It is also likely that, as the final weeks of the season approach, the central warehouse stock is quite limited and unable to fix the inventory shortages in all the stores of the network. The solution is to organize transshipments of units between the stores, accepting extra costs in order to benefit from higher expected revenues in the affected locations.

Therefore, the purpose of the thesis is to investigate how to optimize the reallocation from overstock stores to understock ones, both in terms of maximization of the final expected revenues and in terms of minimization of the transportation costs.

Demands for the stock keeping units (SKUs) are random, so the relative expected sales and the starting inventory position must be considered in the decision, along with transportation costs. Transportation costs are structured as seven stepwise functions of the transportation lot weight; each function is relative to a different zone, which is a proxy of the distance of the destination store from the origin of the transfer. The actual routing of the vehicles is done by the logistic carrier that provided the transportation costs information and thus is out the scope of the work.

The mathematical model faithfully representing the problem of profit maximization shows a non-polynomial function and thus it is not solvable to the optimum using the commercial solvers available. Thus a Mixed Integer Linear Programming (MILP) version of the model has been proposed to approximate the original one. However, the dimension of the problem is very large, involving hundreds of SKUs and stores and all their combinations, and thus it cannot be solved entirely by means of an linear optimizer.

Hence the MILP model was used iteratively on smaller instances, created by dividing the SKUs into subsets, to generate an initial solution. Right after, the MILP was inserted in a Matheuristic framework with a neighborhood search heuristic structure, in which small portions of the initial solution are relaxed and re-optimized at each iteration to improve the results.

Computational results using one such adaptation show that the algorithm is fast enough for practical work, and that substantial improvement in expected profit can be achieved with this approach. Indeed, the tests run on the main SKUs set show an increase in expected revenues of about 78%;at the net of transportation costs, the overall benefits of the operation for the considered SKU set are roughly 59% of the actual expected revenues. This percentage gain, extended to the whole SKU set, can bring significant monetary gains to the company.

Chapter 1

Introduction

1.1 Origin and scope of the project

In the specific context of this thesis, a case of reactive allocation of cloths between stores in a network is faced. The allocation is called *reactive* because it comes after the pre-season and in-season allocation and try to “react” to the mid-season inventory state in the stores, improving the overall resources balance(?).

The opportunity rose from the collaboration with O.R.S. for a big international apparel retail client. The work could be the starting point for a new project enlarging the system of software developed by O.R.S. for managing the supply chain of this client.

For the very practical nature of the problem, the approach followed throughout the work has been “pragmatic”, focusing on achieving the goals satisfying the constraints. This, with the peculiarities of the problem structure, has limited the research and in-depth analysis of the theoretical methodologies: in fact, almost no research have focused on the problem of optimizing the reallocation of merchandise between stores during the selling season, when transportation costs functions are given.

1.2. Thesis disposition

1.3. The Fashion & Apparel industry

Even though a wide range of literature focused on both retailing replenishment optimization and supply chain management, only few researchers have directed their attention on the particular case of the fashion retail industry (Iannone, et al., 2013). In fact, such industry presents specific characteristics that cause several issues in the supply chain management.

First of all, fashion apparel is categorized as innovative product. Whilst functional products typically do not change swiftly over time and have a stable and predictable demand, innovative products, instead, are characterized by novelty, great variety (Vaagen & Wallace, 2008) and customization (De Felice, et al., 2012).

Companies supplying innovative products like fashion apparel are obliged to continuously bring newer innovations in the market to oppose imitators that gradually erode the competitive advantage of the current innovative products, which therefore have a short lifecycle (Barnes, 2009).

This implies the need to have lower production volumes (De Carlo, et al., 2013) and higher flexibility with respect to other retailing industries, making demand unpredictable (Wang, et al., 2012). Demand fluctuations are also caused by the inherent seasonality of fashion goods, whose purchasing behavior is subject to impulsiveness, driven by the product popularity within fashion market and influenced by shelf availability (Lanzilotto, et al., 2014) as well.

All the above suggests that an efficient Supply Chain strategy, which focuses on cost minimization, is more appropriate for functional products, while innovative products Supply Chains should follow a responsive/demand-driven strategy aiming at assuring product availability to match the marketplace with clients demand (Lam & Postle, 2006).

From the supply side, the industry is characterized by long and complex demand-driven supply chains which often are global, including suppliers located in several different Countries (Bruce, et al., 2007).

A traditional textile and apparel supply chain consists of four segments (Sen, 2008) (Fig.1). At the top of the supply chain, there are fiber suppliers who are in charge of collecting the natural or synthetic raw materials; then the second segment, the textile mills, work the rough fiber in different ways to make the fabric. The fabrics are the input to the third segment, the apparel manufacturers - or the manufacturers of industrial textile products – who cut and sew the fabric into finished cloths. The final segment constitutes the distribution and sales part of the chain and includes the warehouses and the final retailers. The warehouses are the hubs of the distribution network where final products are received from suppliers, often located in far countries, and then allocated to stores and clients in accordance with their specific orders (Coraggia, 2009). Finally, the last ring of the supply chain is represented by the Retailers which offer the clothing items and other textile products for sale to final consumers, trying to satisfy their demand.

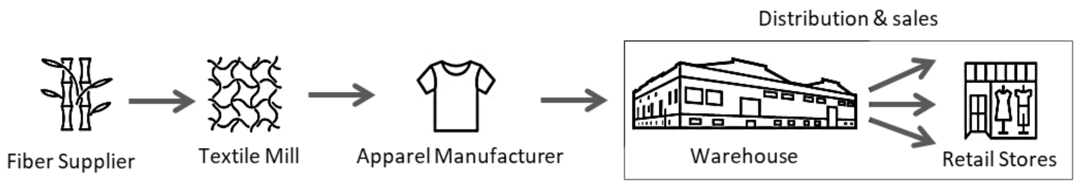


Fig. 1 The clothing and textile supply

The widespread choice of adopting an international supply chain has the benefit of reducing labor costs but it contributes to extend lead time, thus making production inflexible and not able to adapt to demand changes. Indeed, a further feature of the traditional apparel industry is the long time-to-market, especially if the retailer does not have in- house production: it takes almost one year to pass from the definition of the clothing item to its delivery to the stores, after. Such long supply process forces producers to estimate the already volatile demand in far advance, further increasing the uncertainty of the forecasts.

The main processes performed in a typical Fashion Retail Supply Chain are divided in three chronological phases:

1. *Pre-Season* phase: it involves all the activities performed before the beginning of the real sales season, starting from the creation of the

collection and ending with the deliveries of the finished product to clients and stores. (Martino, 2015)

Forecasting of market demand and future orders is one of the pillars of this phase since all the planning activities are based on it. As mentioned in the previous paragraph, this process is made particularly complex in the Fashion & Apparel industry due to demand unpredictability and high volatility. Demand predictions are based on historical sales data, whose information are integrated with characteristics of the new collection and the stores to update with respect to new trends.

Usually, the forecasts are made for an higher aggregation level than the single item code. Single articles are grouped into product categories (Thomassey & Hapiette., 2007) to better reflect consumers purchasing behaviors and, above all, achieve more accurate predictions. Indeed, making aggregated forecasts for few clusters of products, instead of detailed projections for thousands of codes, allows reducing the prevision error.

Classification of single items into wider product categories is also useful when planning for brand new products. Indeed, new fashion products to be offered in the upcoming season have no real historical sales data available to draft forecasts. In these cases, the buying and pricing decisions can be based on past record of similar products.

2. *In-Season* phase: it starts with the first sales recorded in the stores and involves all the selling season, including the end-of-season clearance sales aimed to liquidate all stocks before the end of the season, until the remaindered stock is returned to the central warehouse.

   Even if a first replenishment schedule is defined earlier, stores are not stocked in a unique solution before the selling season. Indeed, depending on the length of the season, it is advisable to allocate only part of the central stock to stores in the first replenishment event and wait to gather actual sales data in order to decide whether to allocate additional units. In fact, it is important to react according to how different products perform in different locations.

Thus, the first replenishment draft before the selling season is based on previsions on past sales data while the following ones, during the in-season phase, are driven by both forecasts and real sales data, which allow to assess deviations between actual demand and projections.

3. *Post-Season* phase: it involves all the activities necessary for the correct management of the unsold items, included their delivery to factory outlet stores, where they will be sold at discounted prices next seasons.

The most valuable asset an enterprise can record is the satisfaction of its clients; this is true for the Fashion Industry as well. Customer satisfaction refers to a customer's overall assessment of the extent to which product or service performance matches the expectations (Anderson & Sullivan, 1993) (Davis-Sramek , et al., 2007). This factor is important to be taken into account because higher satisfaction within the customers basis has the potential to increase clients' loyalty (Martino, 2015) and, in turn, make sales grow. Customer loyalty refers to a customer behavior and positive attitude toward the service/product provider firm. Such behavior includes repeating purchase activity, expressing positive word-of-mouth, having the intention to continue the relationship, not switching to competitors or committing on the long-term (Davis-Sramek , et al., 2007) (Giese, et al., 2004). On this basis, customer satisfaction is considered a key driver for increasing company sales.

Moreover, customer satisfaction is important because of its inherent link with the firm reputation in the eye of the customers.

In the retailing sector – and not only – what causes low customer satisfaction and raises a negative perception of the firms in the eye of customers is the event of low availability of the desired product. The negative perceptions affect the firm's image, hence reducing future visits and total sales, in general.

This is especially true for the Fashion and Apparel Industry in which, as mentioned before, market demand is significantly affected by the availability of products in the stores (Martìnez-de Albèniz & Boada Collado, 2014), that is directly linked to the stores' inventory level.

With low inventory levels, customer may not find the demanded item available in the color or size she needs and thus can decide to avoid purchasing anything or switch to a similar product; in any case she feels a certain level of dissatisfaction. With regard to this, some researches have shown that unavailability of highly demanded items has higher likelihood to negatively impact firm perception if few appropriate substitutes are available (Boatwright & Nunes, 2001) (Broniarczyk, et al., 1998) (Campo, et al., 2000) (Sloot & Verhoef, 2005).

The parameter that better measures availability is the out of stock, which is defined as the number of orders that cannot be fulfilled. Previous research has shown that stock outs (a case of low inventory level in which some codes are missing, i.e. are out of stock), at the same time lower the appeal of the product category and make the customer uncertain on which item to choose or which action to take. Indeed, a customer can decide to act in different ways: to buy another product in the same store, to buy the same product in another store or through other channels (e.g. web and mobile channels) (Lanzilotto, et al., 2015), to wait until the product is available or not to buy at all. In all these cases stock-outs generate customer dissatisfaction; this discontent translates into different economic consequences in the different situations, as out of stock costs are related to the possible lost sale and relative revenue.

All that above does not imply that overstock is the answer to the shelf availability problem. Indeed as fashion items suffer a strong depreciation over time, overstocking would expose stores to higher holding costs and more unsold units at the end of the sales season; the remainder units will be highly marked down in the last season weeks, significantly reducing contribution margins and thus profits.

Thus, the general objective of Retailing Industry actors is dynamically optimizing stores assortment trying to ensure high product availability and to minimize overstock or out of stock events (Iannone, et al., s.d.).

To perceive this goal in the fast changing environment of the Fashion & Apparel industry as well, the ability of being responsive to market fluctuations and react promptly to deviations from forecasts is imperative, especially for companies that manage an extended network of stores and try to satisfy separate demand streams in different locations.

Coping with the demand-supply mismatch on time is especially relevant in the last ring of the supply chain, i.e. retailers, where late corrective actions may be difficult and more expensive.

Chapter 2

Description and analysis of the problem

To create an algorithm for an optimization problem, it is paramount to understand the functioning of the real system, such to distinguish which features have to be modeled in the next phase and which characteristics can instead be overlooked for the purpose of the optimization.

Therefore gathering data and collecting information about how the system of stores and items is organized is a critical phase of the analysis preceding the realization of the algorithm.

Furthermore, it could be useful to classify the case study with respect to the noted problems in literature, such to benefit from the theoretical tools that already exist in reference to it. Nevertheless, even if the literature on supply chain management and inventory allocation problems applied to the fashion industry is substantial, the case on hand cannot be attributed to a class of known problems because of its peculiarities (the location in time of the problem, the presence of transportation costs but the absence of vehicle routing, the fact that the transfers are made between stores and not between a distribution center and the single stores).

To comprehend the problem boundaries and dimensions, it is appropriate to start from the description of the market the case study is dealing with, the taxonomy of store and products used within the system and the products flow during and after season.

2.1 Products and Channels Classification

The reference market is the North America division of the retailer, which encompasses almost 300 stores all around the U.S.A., Canada and minor north American regions. This market is supplied by the retailer through three different sales channels:

1. *Retail Stores* are the channel which the current season products are mainly designed for and in which they are sold at higher prices than other channels (price promotions happen during special events or are applied to special customers, always within a certain threshold). The assortment includes the more recent seasonal products as well as the mainstream basic ones. At the end of a certain season, the products of such season are moved from retail floors to the web.

2. The *web e-commerce* is pretty different from other traditional sales channels. It follows separate rules, sales are differently spread along time and also best seller products are distributed in a diverse manner with respect to traditional stores. At the beginning of a season all the products sold in a retail store are also sold in the e-commerce channel. At the end of the following season, (*Fashion*) products are taken to *Factory Stores*.

3. *Factory Stores* are outlets to which out of season apparels are shipped, including web e-commerce leavings and products selling below expectations on retail floors. In such outlets prices are marked down with respect to the original prices and can be further decreased by promotions.

Products' assortment show a large variety and thus requires a clear codification to be managed at large scales. In fact, items are recorded with a specific hierarchy which, from the highest level of aggregation to the lowest, is:

- Department
- Group Department
- Subdepartment *e.g. Sport Shirts*
- Class *e.g. Oxford Sport Shirts*
- Subclass
- Style: a combination of Model (the design of the Garment), Main Fabric and Fit.
- Colorway : a combination of Style and Color *e.g. White Oxford Sport Shirts*
- SKU (Stock Keeping Unit): a Colorway matched with a Size.

Moreover, each product can be classified according to the length of its lifecycle. Fundamentally, garments can be distinguished between *Fashion* products (*Fashions*), which have a *shor*t lifecycle, and *Basic* products (Basics), which have longer lyficycles. *Fashion* products are told to have *short* lifecycles because they are intended to sell only in a season of a specific year, while *Basic* items have *long* lifecycles because they represent the "evergreen" items of clothing, whose preference is quite independent from temporary fashion trends; ideally *Basics* have an "infinite" lifecycle because they can be sold year after year.

For these reasons, the models developed in the next chapters have been tested on *Fashions* data, since they are more subject to demand swings and to markdowns and final season *clearance actions*[1]. Anyway, the models can be applied to *Basics* data sets as well.

2.2 Problem statement

The problem on hand places during the mid-season of sales: the distribution center is running low in stock because the end season is approaching and it is not able to refill all stores and assuring the high service level (HSL) for each SKU. Evidently, the long lead times of production do not permit to manufacture new products on time to remedy the inventory shortage. In addition, because of the volatility of demand the allocation process did not run perfectly and some store sold less than expected on some SKUs while others have sold these SKUs over expectations. In the first case the stores are holding a stock quantity that exceeds the high service level target estimated for the rest of the season, while in the second case stores do not have sufficient inventory to satisfy the probable demand of the upcoming period.

In this situation, the firm is bearing the risk of ending with a high level of unsold likely lost sales on one hand, and likely lost sales on the other. The realization of these situations would cause lost margins - due to the lost sales and to the remaining inventory markdowns - and/or extra shipping costs to return unsold units to the warehouse at the end of the season.

To mitigate such risks, the firm would like to act in advance moving the items from the stores where they are over the target level (HSL) to the stores that are understock. This practice would increase the chances to sell out the items in the overstock stores while diminishing the probability to incur lost sales in the understock stores. Obviously, this solution is not free since moving stuff across stores requires extra movement and thus extra shipping costs, that are generally

[1] Actions taken at the end of a season to move products from the retail floor to the web e-commerce. In practical terms, such products are remaindered to the warehouse, part of which virtually represents the inventory of the e-commerce.

higher than the tariffs applied for the transportation from the warehouse[2]. So the objective of this work is to find a way to strike a balance between these tradeoff, the higher expected revenues and the higher costs, to try increasing the overall expected profit.

It is relevant to note that this approach is particularly suitable for high value products, whose contribution margin is sufficiently high to justify the transfer cost, while it could be senseless for low price fashion firms [cercare referenze]. Most of the SKUs of the considered problem belong to such typology, but there are also smaller garment products, such as socks and small accessories, that lie outside this set; they are considered anyway since one of the primary objectives of the client is customers satisfaction, which in this case means to fix broken stores assortments spreading the available stock across the stores. Moreover, they are not considered alone but within batches containing higher priced products as well, so the per unit transportation cost for the small items does not result excessive with respect to their value.

2.3 Problem structure and dimensions

2.3.1 Stores and SKUs sets

The wider and more general instance of the problem includes around a hundred of stores for each division (*Factory* and *Retail*) around the U.S.A. territory. The exchanges have to be done only between stores of the same division, Factory Outlets or Retail Stores, since these two divisions do not sell the same merchandise at the same time and with the same pricing rules. This implies that the two sets of stores and relative SKUs will be treated as two separated

[2] The transportation costs from the warehouse to the single stores are given by a flat tariff and thus are independent from the weight of the moved lot. Instead, in the case on hand the weight of the transportation batch determines the total shipping costs from a store to another.

instances of the same problem, given that there are no particular features distinguishing the two reallocation problems apart for the input data.

Within every division, each store has its own colorways and SKUs sets on sales, which do not perfectly match with the other stores' sets.

2.3.2 Demand

As it is typical in the fashion industry, demand is stochastic. From the analysis made by O.R.S. on historical sales data, it emerged that sales follow a Poisson distribution, which is estimated at the colorway level and then turned on the single stock keeping units. Due to the differences among stores (location, climate, type of clientele..etc.), each colorway in each store has its own distribution parameter for sales λ (the expected sales for the colorway), which is then projected to the single SKUs according to their selling frequency within the colorway assortment – which, again, varies from store to store. Consequently, each SKU can have a different definition of the service level quantity (HSL)[3] for each store in which it is sold.

This implies that it is not indifferent to move an SKU to a store or another that has the same initial inventory, since the probability to sell that SKU could be

[3] "In inventory management, service level is the expected probability of not hitting a stock-out during the next replenishment cycle or the probability of not losing sales. Safety stock is inventory that is carried to prevent stock outs. Companies choose to keep safety stock level high as a buffer against demand variability: the safety stock level must be high enough to cover vendor's delivery times, sufficient enough to satisfy customers' demand, but not so high that the company loses money because of high carrying costs. The target service level can be therefore defined as a trade-off between the cost of inventory and the cost of stock-outs." (Radasanu, 2016)

So the high service level quantity (HSL) coincides with an high quantile of the probability distribution (e.g. 95%) which defines the target quantity a store should hold in inventory to avoid out-of-stocks with adequate confidence level.

very different in the two locations and thus could impact the overall expected profit in different ways.

2.3.3 Inventory constraints

Each SKU in the store's SKUs' set has an initial inventory Q^0, which is the stock available in store at the time the reallocation is settled. Comparing this stock-on-hand and the target inventory level (HSL) of a SKU, a store can be classified either as in-stock (if the two quantities coincide), understock (if the on-hand quantity is lower than the target one) or overstock (if the initial stock exceed the HSL) for such SKU. In case of overstocked store the objective is to remove the units in excess and redistribute them across the understock stores, which in turn can only receive the SKUs for which they are running low in inventory and send the ones for which their stock exceeds the high service level quantity. The stores should neither send or receive the SKUs that are exactly in-stock.

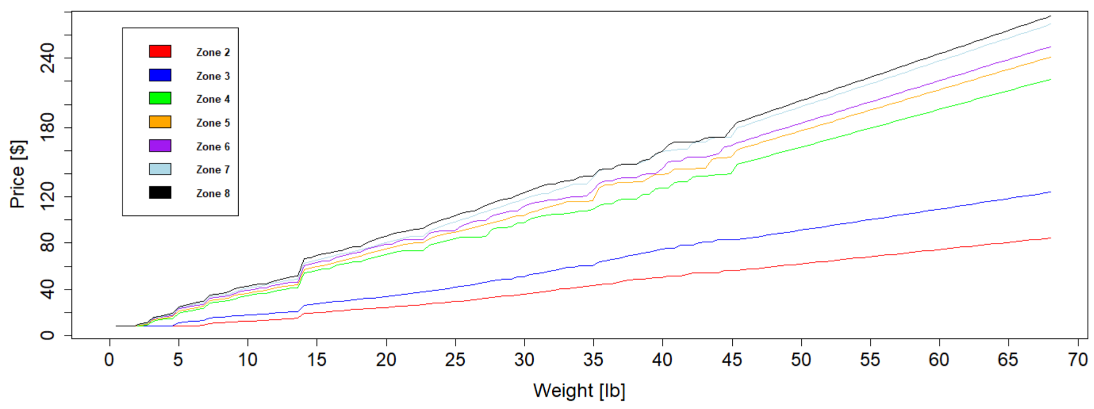
To decide how much to transfer from a store and where to move the merchandise, the decision maker should strike a balance between the selling probabilities - and thus the expected sales - in every destination store and the relative transportation cost.

2.3.4 Transportation costs

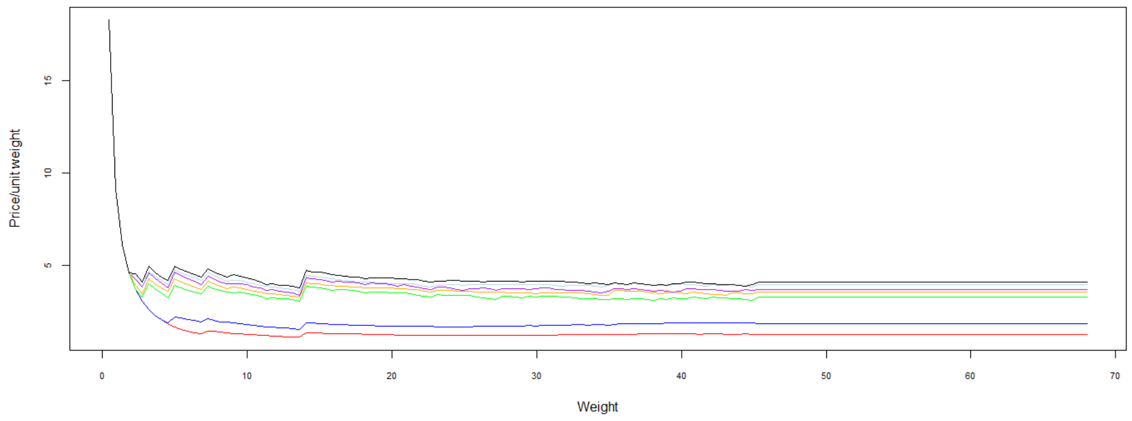
In the case on hand the transportation costs structure is based on information provided by the external logistic carrier that will actually handle the routing of vehicles, after the exchanges have been defined. For this reason, in the formulations of the following chapters, transfers are managed as single paths from the origin store to the destination, regardless of the effective vehicle routing the carrier is going to implement - which is out the scope of this work.

In logistics transportation costs are generally a function of the distance and the weight/volume of the moved batch. In the specific case, the information about the distance and the difficulty of reaching a specific location is condensed in the zone number of the destination with respect to the store of origin (always considered in zone 1). There are seven zones, indexed 2 to 8, with increasing transportation cost per batch weight as the zone number increases. Thus, if store B is placed in zone 2 with respect to the sender store A and store C is in zone 3 with respect to A, a batch with a given weight will cost more if sent from A to C than from A to B. It is noteworthy to highlight that the zone matrix (or distance matrix) reporting the zone number for each combination of sender and destination store is not symmetric, so there can be found cases in which a destination store B is in zone X with respect to A, while A is in Zone Y (Y≠X) with respect to the sender store B.

Every zone tariff is divided into 147 equal weight ranges - of approximately 0,45 lb - for which each zone number has its own transportation cost. Thus, each zone tariff shows a transportation cost function which is stepwise constant with respect



to the lot weight and overall increasing with the total weight of the lot, as shown in [fig.]. The transportation cost per pound instead is decreasing as the total lot weight increases [fig.], which implies that is more cost-effective to transfer large lots between two stores than sending small packages to multiple destination stores.



2.4 Objectives

The satisfaction of the client is the most valuable assets of an enterprise. Measuring the service level is relevant because it can affect the relationship with the customers and can determine an important impact on profitability. In the retail sector setting a high level of service (greater or equal to 95%) is crucial since the level of service is a key factor in assuring the fidelity of the clients and the maximization of sales. (Radasanu, 2016)

The client firm of this project gives lot of consideration to customer satisfaction as well and try to avoid stock-outs and consequent lost sales. With this purpose, it sets a high service level target and allows specific in store orders for customers who could not find their size in stock for the clothing they wanted to buy (which are handled by another O.R.S.' software).

Consequently, from the apparel firm perspective, the optimal situation would be to have the HSL quantity available in inventory for every SKU in every store. However, it is likely that the reallocation problem arises in a situation in which the total resources are too scares to reach this target level in every store-SKU combination, since it is probable that the warehouses are facing inventory shortages as well. In these circumstances, there will be stores with HSL quantities or even more units in stock for some SKUs but most of the stores will run low in some SKUs' inventory or even face deficits in the colorway assortment. Consequently, it is reasonable to set as objective not the achievement of the target HSL quantities in every store and SKU, but rather the reallocation of the exceeding merchandise from the overstock stores such to maximize the expected profit.

Chapter 3

Solution approaches

An *algorithm* is a step-by-step procedure for solving a computational problem. For a given input x, it generates the correct output *f*(x) - the answer for a corresponding problem solved - after a finite number of steps.

Solution methods for a general optimization problem can be divided into exact and approximated ones.

If the algorithm gives an *optimum solution*, it is called *exact* algorithm. Exact methods are usually limited to small instances; they include mixed integer linear programming (MILP), dynamic programming (DP) and branch and bound (BB) methods.

For larger instances, in order to find a "good" solution within an acceptable amount of time, two types of approximate methods can be developed: approximation and heuristics algorithms.

Heuristic algorithms can be very simple but still effective, producing "good" feasible solutions which are not guaranteed to be close to optimum. The performance of a *heuristic* algorithm is usually analyzed experimentally, through a number of runs using either generated instances or known benchmark instances. For instance, well-known examples of heuristic algorithms are the Tabu search, the Simulated Annealing and the Genetic Algorithms.

An algorithm is called an *approximation* algorithm if it is possible to establish analytically how close the generated solution is to the optimum (either in the worst-case or on average), therefore the solutions found are *guaranteed* to be within a fixed percentage of the actual optimum. *Approximation algorithms* produce solutions in *polynomial time*, but for the price of loss of optimality.

A ρ-*approximation algorithm* is an algorithm that runs in polynomial time and delivers a solution of value at most ρ times the optimum for any instance of the problem. The value of ρ is called the *worst-case ratio bound* and estimates the "goodness" of the algorithm.

Given the practical nature of the case treated in this work, it is not interesting to focus on approximation algorithm which instead can have relevance in a theoretical dissertation. Rather, next chapters will focus first on exact methods, in particular MILP models applied to restricted instances, and then on heuristics approaches that permit to deal with the real dimensions of the practical problem.

3.1. Heuristic Approaches

Heuristic algorithms are a set of solution techniques of complex combinatorial problems able to bring satisfying results within limited running times. Differently from exhaustive algorithms, these methodologies cannot guarantee optimality, but try to achieve results as close as possible to the best solution. To this aim, the resolution decisions are taken basing on experience related to the structure of the problem, this way allowing to avoid the enumeration of all the different possibilities and thus limiting the computational costs of the algorithm.

Therefore, it is necessary to deeply analyze the problem to understand how to exploit its properties in the resolution and building an efficient and effective heuristic.

Heuristic algorithms can be classified into categories which differ for both complexity and quality of the results:

- *Constructive* (or *Greedy) algorithms*.
- *Neighborhood Search Heuristics.*

3.1.1. Constructive (Greedy) algorithms

Constructive (or *Greedy) algorithms*. Constructive algorithms, as their name itself reveals, "build up" an admissible solution starting from the problem data. In practice, these algorithms start from an empty solution and, at each iteration, progressively expand the partial solution making choices that respect the problem constraints. Usually, these choices are made following very simple rules which lead to the decision that appears to be the most convenient at the moment to reach the optimum. Given that at each step the most desirable way is chosen, these algorithm are also called greedy. A characteristic of the greedy algorithm is that once the choices are made they are not later challenged in the subsequent steps, i.e. no backtracking mechanisms are provided.

The strategies to determine at each step which alternative is the most promising one can be multiple, like the evaluation of the objective function of the partial solutions or the estimation of a score computed basing on the reached status, but all are based on the local optimization criterion. As a matter of fact, at each iteration, the decision made optimizes only a small portion of the original problem, with the hope of reaching a global optimization. Although, the best solution is achieved in problems showing optimal sub-structures, but it rarely happens in normal problems; hence it is very common for this type of algorithm to fall back into a local optimum. In regards to this issue, a possible improvement can be reached by introducing a certain level of randomness in the choices made, for instance selecting randomly among the n best feasible alternatives or between the possibilities that differ less that a certain percentage from the best one.

3.1.2. Neighborhood Search Heuristics

The local search algorithm are methodologies that, starting from an initial solution, try to get improvements exploring the solution space around the current solution. This solution space around a starting poit is called neighborhood and represents the portion of the overall solution space which is reachable applying a well-defined operator to the current solution.

This type of algorithms is not able to start from an empty solution but need an initial feasible solution, which is generally provided by constructive-type algorithms, which are a fast way to get a feasible solution for the problem.

Moreover, in the context of neighborhood search algorithms it is necessary to define a neighborhood structure; to do this, first of all the solutions representation of the problem has to be defined. This is definitely a very critical passage since the representation of a solution heavily influences the neighborhood types that can be generated and the complexity of the objective function evaluation. Then, once the solution representation is established, a set of operators have to be selected to be applied to such representation, in order to create other solutions from the current one in an easy way. The choice of the neighborhood operators is not trivial since it influences the goodness of the local search and can even prevent the algorithm from reaching the optimum (which is the case for disconnected neighborhoods).

Once the neighborhood is defined, a strategy to explore it and select the next solution has to be chosen. Among the most known local search strategies there are:

- The Steepest Descent (or Best Improvement) strategy: the neighborhood is completely explored and the best solution found in it is chosen to become the curent solution of the next iteration.

- The First Improvement strategy: the neighborhood is not completely indagated at each iteration, often the neighbors are generated one at a time and the iteration stop as soon as a solution improving the current one is found. Consequently, this strategy allow to apply the local search also to neighborhoods that are too big to be fully evaluated

The efficacy of both methodologies is strongly linked to the type, structure, dimension and feasibility of the chosen neighborhood.

However, generally speaking, the Steepest Descent technique reaches an higher improvement at each step but it is slower in finding such improvement, while the First Improvement strategy finds lower improvements at each iteration but in a much faster way. Overall the two strategies have similar performances but, typically, the first improvement strategy reaches better final solutions because it is easier to quickly get stuck in a local optimum using the Steepest Descent approach.

Indeed, the inability to avoid getting stuck into local optima - the situation in which the exploration of the solution space is not able to find further improvements even if they actually exist - is the main weakness point of this techniques.

Neighborhood search methodologies can be divided into:

- *Classical local search*

- Metaheuristics (*Iterated Local Search, Variable neighborhood search, Tabu Search*, Grasp, *Simulated Annealing, Genetic algorithms*..etc.)

- Matheuristics: a more recent family of resolution approaches based on the hybridization of heuristics methodologies and exact methods that will be treated in further detail in Chapter 5…)

The greedy algorithms are generally more simple and fast, while the neighborhood search heuristics are more complex, in particular the metaheuristics and matheuristics. Obviously this affects the running time, which increases as the methodologies get more complex, as well as the solutions goodness, that is generally low for the greedy approaches but gets better in the neighborhood techniques, especially for the metaheuristics and matheuristics.

3.2. Computational complexity theory

The purpose of the complexity theory is to measure the performances of a given resolution algorithm with respect to the necessary computational resources, i.e. the time and memory space. Even if in some applications the measure of the memory space needed by an algorithm can be a decisive factor, generally it is considered an issue of minor importance; therefore, this paragraph will linger only on the running time, bearing in mind that an analogous reasoning can be applied to the other resource as well.

All the input parameters x of a problem have to be firstly represented through a finite series of symbols, that will determine the dimension of the input. In general, given a certain problem, a set of input parameters x of codified dimension n is called an *instance* of the problem.

The time complexity (or the running time) of an algorithm expresses the total number of elementary operations, such as additions, multiplications and comparisons, for each possible problem instance as a function of the size of the instance. Formally speaking, the time complexity of an algorithm in relation to a certain problem is defined as the function T(n) that determines an upper bound to the number of steps made by the algorithm to solve the problem instance of size n.

The measure of the computational complexity of the algorithms in terms of running time is based on their response to the increase in size of the examined problem. Often, it is not easy to determine with precision the function T(n), hence it is preferred to define the asymptotic behavior of such function, considering only the predominant terms when n tend towards infinite. So, the big-O notation is used to specify the complexity of an algorithm, which implies that:

$T(n) \in O(g(n))$ if there exists a constant $c > 0$ and a non-negative number n0 such that $T(n) \leq cg(n)\ \forall n : n \geq n0$. For instance, if the computational complexity is $O(n^2)$, the big O-symbol to stress that the number of elementary computations of the algorithm grows at the same rate as the function Cn^2, where C is a constant.

Computational complexity allows to divide algorithm into categories sharing the same asymptotic behavior:

- Polynomial time algorithms. An algorithm is called polynomial if $f(n)$ can be computed in at most $O(g(n))$ steps where g is a polynomial of certain degree, so its complexity is upper bounded by such polynomial. This is the case of algorithms having complexity $O(n^k)$, including constant (k=0) ,linear (k=1), quadratic (k=2) and cubic (k=3) ones. Algorithms having complexity function in the form of $O(n^k \cdot \log^w(n))$ are considered polynomial as well.

- Super-polynomial algorithms, whose computational complexity cannot be upper bounded by any polynomial. Exponential algorithms ($O(a^n)$) and factorial ones ($O(n!)$) belong to this category.

The difference among the various complexity levels are particular evident when the dimension of the instances grow. Indeed, in the exponential algorithms the running time sees and extraordinary increase with respect to the instance size.

Problems can be divided into *optimization* and *decision* problems. A problem is called a *decision problem* if the output range is {yes, no}. Computational complexity theory's explanation refers to the decision version of optimization problems. It may be associated with each problem a decision problem by defining a threshold k for the objective function f. For instance, for a minimization problems, the relative decision problem is: does a feasible solution S exist satisfying $f(S) \leq k$?

Note that it is always possible to find the optimal solution of an optimization problem by solving iteratively its decision problem, varying the bound *k*. Thus, optimization problems and their decision versions are strongly connected: if there exists a polynomial algorithm for the optimization problem, there exist one for the decision problem as well and vice versa. Therefore the theory developed for decision problems can be immediately extended to the corresponding optimization problems.

3.2.1. Complexity classes

Problems can be divided into computational classes that indicates their difficulty.

P, which stands for "Polynomial time complexity", is the class of decision problems which are polynomially solvable. A problem is called polynomially solvable if it can be solved by a polynomial algorithm that bring to the correct answer (yes/no}. Polynomial algorithms are sometimes called *efficient* or simply *good*.

NP is the class of polynomially *checkable* decision problems with the property that for each "yes"-answer a certificate exists which can be used to verify the "yes"-answer in polynomial time. In other words, given a certain hypothesis of solution for the decision problem, it is possible to verify if such answer is correct or not using a polynomial time algorithm. *NP* stands for "non-deterministic polynomial" because the first phase of resolution is executed through an non-deterministic procedure.

One of the biggest unsolved problems of modern mathematics is to determine if the class P is strictly included in NP or if the two sets coincide. In the last case, a great number of high complexity would be solvable by means of polynomial algorithms, so it is supposed not to be true, but it has not been demonstrated yet.

Reducibility, NP-complete and NP-hard problems

An important concept in the computational complexity theory is the polynomial reducibility of one problem to another. For two decision problems P and Q, it is said that P reduces to Q (denoted by $P \propto Q$) if there exists a polynomial-time computable function g that transforms inputs for P into inputs for Q such that n is a "yes"-input for P if and only if g(n) is a "yes"-input for Q. Polynomial reducibility allow to compare the complexity of one problem with respect to another one, since saying that $P \propto Q$ equals afforming that Q is at least as difficult as P and, consequently, if Q can be solved by means of a polynomial algorithm this is valid also for P.

Polynomial reducibility allow to introduce the class of NP-complete and NP-hard problems. A decision problem Q is called NP - complete if $Q \in NP$ (first condition) and, for all other decision problems $P \in NP$, we have $P \propto Q$, i.e. all problems in NP can be polynomially reduced to Q (second condition). Therefore, all NP-complete problems have the same difficulty and the NP-complete class includes the set of the most difficult problems of the NP class. Lot of problems belong to the NP-complete class, but so far no polynomial time algorithms are known to solve them.

An optimization problem is NP- hard if its decision version is NP- complete. So the NP-hard class definition coincide with the definition of the NP-complete class where the first condition is relaxed. So, problems belonging to this class are at least as difficult as the NP-complete ones but not necessarily belong to the NP class, therefore they comprehend also non-decision version of the NP-complete problems, like the classic optimization ones.

For such problems, no polynomial-time algorithms are known and it is generally believed that these problems cannot be solved in polynomial time, and therefore they should be treated by other methods.

As far as the present problem is concerned, the computation class it belongs to is not known, however it is believed that it is "hard" to solve, meaning that it is not solvable by means of an exact polynomial algorithm.

Chapter 4

Mathematical models

[frase introduttiva capitolo?]

4.1. Notation

This paragraph introduces the symbols employed to model constants and variables in the subsequent models. The notation for parameters follows the classification and the problem features anticipated in the previous chapter and it is held unvaried throughout the whole work. The variables chosen for the mathematical formulation of the problem are presented here for the first time and their definitions are maintained in the following chapters as well.

Constants

N = total number of stores (Retail or Factory Outlet)

S = total number of SKUs

T = total number of tariff slots (equal to 147 in every problem instance)

E = total number of units exchangeable in the problem

p_{si} = price of SKU s in store i[4]

w_s = weight of SKU s

[4] As is typical in the retail industry, the selling price of a colorway may vary across stores, but it is identical for all sizes of the same colorway sold in the same store.

λ_{si} = Lambda of SKU s in store i (expected sales in the next period until end-of-season)

Q_{si}^{0} = initial inventory of SKU s in store i

$\mathrm{HSL_{is}}$ = high service level quantity of SKU s in store i

$$\mathrm{U_{si}} = \begin{cases} 1 & \text{if } \mathrm{Q_{si}^{0}} < \mathrm{HSL_{is}}, \text{ i.e. if store i is understock for the size s} \\ 0 & \text{if } \mathrm{Q_{si}^{0}} \geq \mathrm{HSL_{is}}, \text{ i.e. if store i is not understock for the size s} \end{cases}$$

$$\mathrm{UU_{si}} = \begin{cases} 1 & \text{if } \mathrm{Q_{si}^{0}} < \lambda_{si}, \text{ i.e. store i is seriously understock for the size s} \\ & \text{(initial quantity would not be enough to cover expected sales)} \\ 0 & \text{if } \mathrm{Q_{si}^{0}} \geq \lambda_{si}, \text{ i.e. store i is not seriously understock for the size s} \\ & \text{(initial quantity would be enough to cover expected sales)} \end{cases}$$

CT_{ijt} = cost of slot t in the transportation tariff between I and j

b_t= maximum weight available for transportation in tariff slot t[5]

Variables

x_{sij} = units of SKU s moved from store i inventory to store j

$$C_{ijt} = \begin{cases} 1 & \text{if for the transfer from i to j the slot t of the tariff is applied} \\ 0 & \text{otherwise} \end{cases}$$

4.2. The reactive allocation problem

The following model is the literal transposition of the objectives and constraints expressed in the previous chapter in mathematical terms.

[5] Note that such weight slots are equal for every zone tariff, the difference among tariffs lies in the costs CT_{ijt} relative to each b_t.

$$max \sum_{s=1}^{S} \sum_{i=1}^{N} \sum_{q_{si}=0}^{Q_{si}^0 - \sum_j^N x_{sij}} (\frac{\lambda_{s,i}^q e^{-\lambda_{si}}}{q_{si}!}) q_{si} (1 - U_{si}) p_{si}$$

$$+ \sum_{s=1}^{S} \sum_{j=1}^{N} \sum_{q_{sj}=0}^{Q_{sj}^0 + \sum_i^N x_{sij}} \frac{\lambda_{sj}^q e^{-\lambda_{sj}}}{q_{sj}!} q_{sj} \left(U_{sj}\right) p_{sj} \tag{3.1}$$

$$- \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{t=1}^{T} C_{ijt} CT_{ijt}$$

Subject to

$$\sum_{j}^{N} x_{ijs} \leq (Q_{is}^0 - HSL_{is}) (1 - U_{si}) \quad \forall i = 1..N, \forall\, s = 1..S \tag{3.2}$$

$$\sum_{i}^{N} x_{ijs} \leq \left(HSL_{js} - Q_{js}^0\right) \left(U_{sj}\right) \quad \forall j = 1..N, s = 1..S \tag{3.3}$$

$$x_{iis} = 0 \;\; \forall i = 1..N, \forall\, s = 1..S \tag{3.4}$$

$$C_{iit} = 0 \;\; \forall i = 1..N, \forall\, t = 1..T \tag{3.5}$$

$$\sum_{t}^{T} C_{ijt} \leq 1 \;\; \forall i = 1..N, j = 1..N \tag{3.6}$$

$$\sum_{t}^{T} C_{ijt} \leq \sum_{s}^{S} x_{ijs} \quad \forall i = 1..N, j = 1..N \tag{3.7}$$

$$\sum_{s}^{S} w_s x_{ijs} \leq \sum_{t}^{T} C_{ijt}\, b_t \;\; \forall i = 1..N, j = 1..N \tag{3.8}$$

$$x_{ijs} \in \mathbb{N}^+ \forall i = 1..N, \forall\, j = 1..N, \forall\, s = 1..S \tag{3.9}$$

$$C_{tij} \in [0,1] \;\; \forall i = 1..N, \forall\, j = 1..N, \forall\, t = 1..T \tag{3.10}$$

Recalling the notation in the previous paragraph, the primary decision variables x_{ijs} represent the shipment quantities of each size s ∈ [1..S] from each sender store i ∈ [1..N] to each destination store j ∈ [1..N], where N and S are respectively the total number of stores and the total number of SKUs in the problem instance. These variables are constrained to be integer in (3.9) and are bounded by the inventory available/receivable for transfer in each store for each SKU:

- Constraint (3.2) limits the quantities of each SKU that can be sent from every store: if a generic store i overstocks for the SKU s ($\mathrm{U_{si}} = 0$), its total shipment of s to other stores must never exceed the quantity $Q_{is}^0 - HSL_{is}$ which insures that its target service level quantity is preserved after the reshuffling while, if store i understocks for such SKU ($\mathrm{U_{si}} = 1$), it cannot send it to any other store. Therefore, this constraint coincide with the logical implication

$$\mathrm{U_{si}} = 1 \rightarrow \sum_{j=1}^{N} x_{ijs} = 0 \quad \forall i = 1..N, \forall\, s = 1..S$$

$$\wedge$$

$$\mathrm{U_{si}} = 0 \rightarrow \sum_{j=1}^{N} x_{ijs} \leq (Q_{is}^0 - HSL_{is}) \ \forall i = 1..N, \forall\, s = 1..S$$

- Similarly, constraint (3.3) limits the quantities of each SKU that can be received by every store. Every destination store j which is understock for s ($\mathrm{U_{sj}} = 1$) can acquire units of this size from all the other stores up until its inventory reaches the $\mathrm{HSL_{is}}$ quantity. On the contrary, a store overstocking s ($\mathrm{U_{sj}} = 0$) cannot get additional units for that size from any other store.

$$\mathrm{U_{sj}} = 1 \rightarrow \sum_{i=1}^{N} x_{ijs} \leq \left(HSL_{js} - Q_{js}^0\right) \quad \forall j = 1..N, \forall\, s = 1..S$$

$$\wedge$$

$$\mathrm{U_{sj}} = 0 \rightarrow \sum_{i=1}^{N} x_{ijs} = 0 \quad \forall j = 1..N, \forall\, s = 1..S$$

The secondary decision variables C_{tij} help modeling the stepwise transportation cost function between store i and location j. Specifically, every binary (3.10) C_{tij} indicates if between i and j the *t-th* tariff with upper weight bound $\mathrm{b_t}$ is activated

($C_{tij} = 0$) or not ($C_{tij} = 1$), which depends on the total heaviness of the lot transferred from i to j, $\sum_s^S w_s x_{ijs}$ (3.8). Clearly, only one tariff can be applied to the batch movement (3.6) and no tariff will be on if there is no transfer between two stores (3.7). The corresponding logical implications are:

$$\sum_s^S x_{ijs} = 0 \rightarrow \sum_t^T C_{ijt} = 0 \quad \forall i = 1..N, j = 1..N$$

$$\wedge$$

$$\sum_s^S x_{ijs} > 0 \rightarrow \sum_t^T C_{ijt} = 1 \quad \forall i = 1..N, j = 1..N$$

Since the costs appear with negative sign in the objective function (3.1), the solver will tend to activate the cheapest tariff that respects the lower bound given by the total lot weight.

Lastly, constraints (3.4) and (3.5) state the obvious condition that no store can exchange items with itself.

Expression (3.1) defines the objective function to be maximized. As anticipated in the previous chapter, the ideal goal would be to satisfy the high service level condition in every store for every SKU. Nonetheless it is likely that the total resources available in the whole stores network are too scarce to allow the achievement of such target in every location. Consequently, it is reasonable to reallocate the exceeding merchandise present in the overstock stores with the purpose to maximize the expected profit, minimizing the transportation charges and maintaining the high service level (HSL) requirement only where it is possible and costless, i.e. in the stores that already record at least HSL units in their initial SKU stock (which are the in-stock or overstock stores).

Thus the objective function is given by the expected profit, computed as the expected revenues after the transfer operation minus the transfers costs (last term of (3.1)). The expected revenues are computed as the probability of selling a quantity q times the quantity itself times the price of the relative SKU in the store (first and second term of (3.1)), summed over all sales scenarios from 0 to the final stock quantity – i.e. the inventory after the reshuffling – which is equal to

$Q_{si}^{0} - \sum_{j}^{N} x_{sij}$ for the stores i that are overstock for SKU s, and $Q_{sj}^{0} + \sum_{i}^{N} x_{sij}$ for the stores j that, instead, understock for SKU s.

As the probability calculations reveal, each article have been considered independently from the others. Nevertheless, this problem may involve connections between different articles or between different SKUs of the same colorway.

In the former case, there can be substitutions or complementarities[6] among the products at the store level: the sales probability of an item in a store can be boosted by the absence of a similar product (which means that the customer is willing to switch among the two cloths, which therefore can be considered substitutes) or by the presence of complementary products (for instance, a tie sales distribution could be positively related to the sales distribution of a particularly well matched shirt).

In the second hypothesis, the interrelations appear between SKUs belonging to the same colorway set, i.e. within a colorway assortment. In this case, it could happen that the completeness of the assortment affect the selling probabilities of the single colorway SKUs. It could occur because of the psychological impact a more complete sizes assortment can have on the final customer entering the shop: for instance, the more the SKUs displayed in the store the more the visibility for

6 In economic theory, two goods are substitutes if when the price increases for one good, the demand for the substitute product will increase (assuming that price remains constant). Instead, complementary goods literally complement each other, they are items that "go together", so if the price of one increases the demand for the other will decrease. (Munson, 2014)

However, in this treatise complementarity and substitution meanings are not linked to the elasticity of one good's demand with respect to the variation in another good's price, they are considered as the elasticity of a SKU demand with respect to the presence or the absence of another SKU in the stock. So the magnitude of substitution effect indicates how much a consumer is willing to switch to another product if the first choice SKU is missing instead of leaving the store without purchasing anything. While the complementarity effect measures how much the two goods can be considered complement and thus are purchased together by the customers.

the whole colorway set and the more the client could be induced to notice and try on that product.

On the contrary, it could happen that the single SKU sales are increased by the absence of the SKUs right close in the size scale, in particular when the measures difference among subsequent sizes is not so remarkable.

However, for the above mentioned hypothesis to be tested, a lot of accurate data at the SKU level in all combinations (in this case, every SKUs pair) are needed to draw statistically significant results.

Thus, because of data availability and for computational reasons the implemented formulation considers each article independently. Moreover, the conditional probability contribution to the overall expected profit is expected to be negligible with respect to the independent probability term, thus it can be ignored in first approximation.

4.2.1. Nonlinear optimization issues

Il can be seen that the objective function, unlike the constraints' expressions, is not linear neither polynomial. Moreover, the integrity constraints for the variables add further complexity to optimization of this model. Indeed, while the research on continuous optimization is in an advanced state, the literature on mixed-integer nonlinear optimization is still in progress: nowadays there exist few solvers handling non-polynomial problems and, generally speaking, they are not so efficient. That is because nonlinear optimization is intrinsically very difficult to solve.

Indeed, in convex optimization problems, which include linear programming ones (LP), the places to look for the optimum are limited to the extreme points (corner points) of the feasible region polytope [Fig.]. This optimum is the point with the best value of the objective function anywhere in the feasible region, thus it is called global optimum.

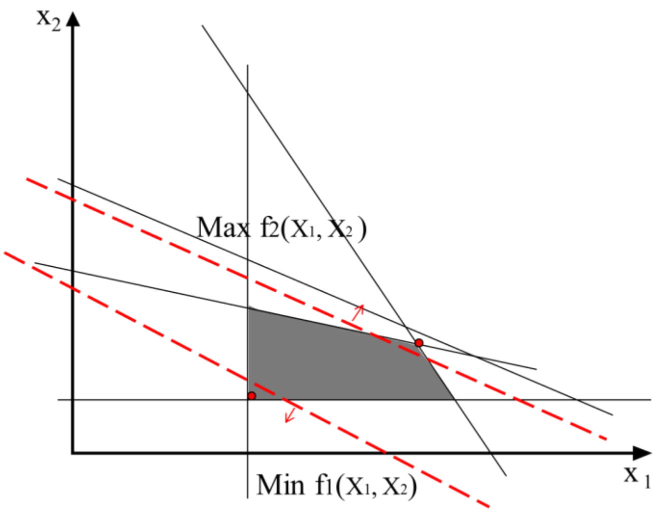


Fig. 2 Feasibility polytope and optimum point position with respect to the objective function direction.

On the contrary, in nonlinear programs optima are not restricted to extreme points, they can be anywhere in the feasibility region. Moreover these optima are not necessarily unique and optimizing nonlinear objective functions do not assure to reach the *global optimal* solution: the solver can get stuck in a *locally optimal* solutions that have better objective function values than any other feasible solutions in their “vicinity", but do not coincide with the *global optimal* solution (as distinct from convex problems) [Fig.].

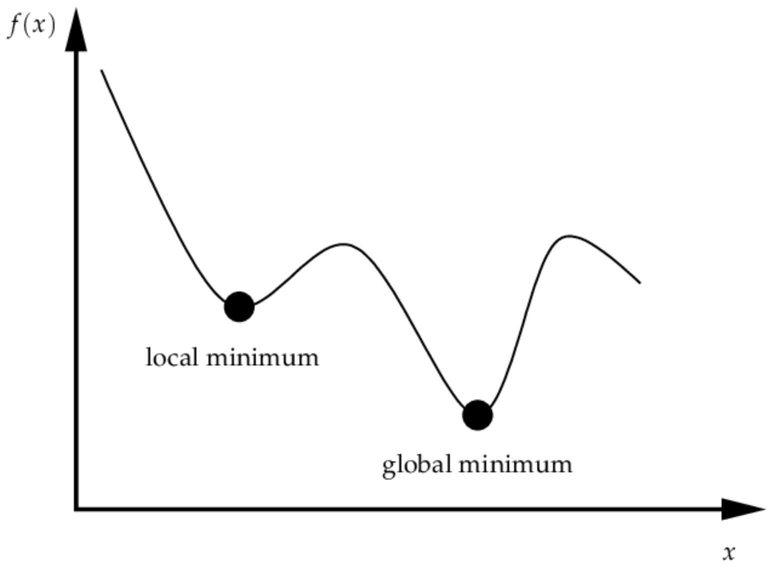


Fig. 3 Local vs global optima in a non-convex function

This difference among the two categories influences also the time efficiency of the optimization problem: in linear (more generally convex) problems it is “fast” to find the optimal solution or, alternatively, to prove the absence of a feasible solution, whilst it can take a lot of time to verify whether a nonlinear (nonconvex) problem has no solution of if a local optimum is also global.

4.3. A MILP formulation for the problem

For all the issues non-linear optimization poses, it is useful to try linearize the problem. The most common approach is the piecewise linearization[7] of the non-linear objective function, which encompasses different methods to do this conversion. However, in the case on hand the non-linear function is not even polynomial and it is quite complicated, combining powers, exponentials and factorials altogether, which makes it also difficult to be represented graphically, so the piecewise linearization appears not to be the best strategy.

A more straightforward approach is the exploitation of some characteristics of the specific problem and a priori knowledge to write a different objective function that leads to similar results.

[7] A piecewise linear approximation is a method of constructing a function g(x) that fits a nonconvex objective function f(x) by adding extra binary variables, continuous variables, and constraints to reformulate the original problem. The specific goal is to approximate a single valued function of one variable in terms of a sequence of linear segments. Optimization problems with piecewise linear costs arise in many application domains; for example, the transportation cost, inventory cost, and production cost in a supply chain network are often constructed as a sum of nonconvex piecewise linear functions due to economies of scale. (Lin, et al., 2013)

Specifically, one can use the information about the probability distributions of the SKUs sales and the knowledge about the positioning of each store inventory with respect to the HSL level and the expected sales λ_{si} for each SKU. Indeed, the objective function aims at reallocating units among stores such to reach a final inventory quantity that optimizes the revenues of each SKU (minus the cost of the batches transferred between stores), balancing the resources such to assure availability to the greatest number of stores and, at the same time, fostering the allocation to the more "promising" ones – the stores who are expected to sell more of a given SKU, thus contributing to higher expected revenues - while respecting the high service level requirement (when it is sensible) to minimize lost sales.

But the probability distributions are known at the beginning of the reshuffling: sales follow a Poisson distribution with a parameter λ_{si} for each SKU, coinciding with the expected sales in the next selling period of the season. The shape of the Poisson distribution is well known in statistics; its density function shows a peak in correspondence of the lambda value, which represents both the mean and the median of the distribution [Fig.3].

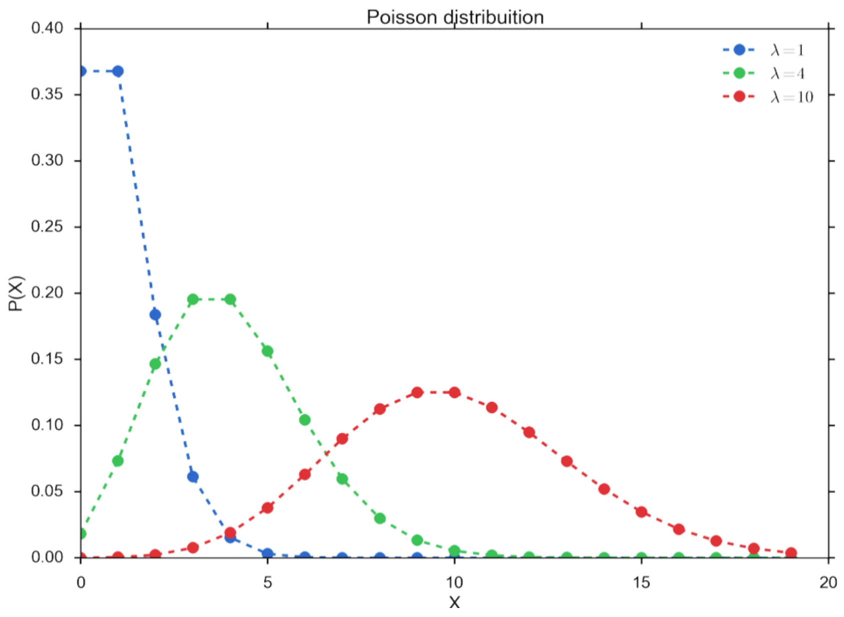


Fig. 4 Poisson probability density function with different lambdas

This implies that the marginal contribution of an additional unit to the expected revenues – given by the q-th quantity times the probability of selling q units times the relative price - increases as the stock approaches the lambda quantity and starts decreasing after the value $\lambda + 1$. Consequently it can be more convenient, both in terms of availability and incremental expected gain, to move one unit to a store that has not reached the λ level, rather than allocating an additional unit to a store that already overcomes the λ level in its SKU inventory, and thus it is building safety stock. This is also intuitively sensible: a store that already holds λ or more units in its stock for a certain SKU on average will be able to satisfy its expected demand, and any unit added will incur an higher probability to remain unsold; on the contrary, a store holding less than λ units for a SKU will not likely able to satisfy its expected demand and thus has high chances to incur in lost sales and customer dissatisfaction.

Combining the information about the likelihood distribution and the initial inventory available, one can recognize that each store-SKU combination can be in one out of three different situations with respect to the initial stock, each of which requires a different intervention. [Fig. 4]

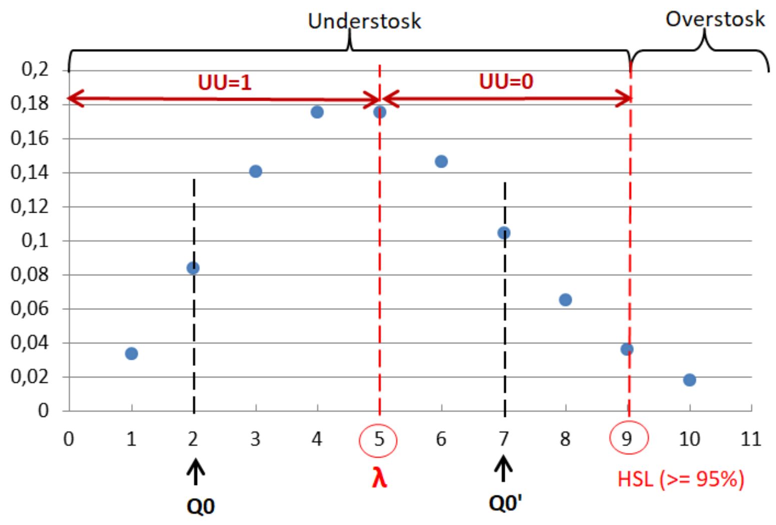


Fig. 5 Initial inventory states with respect to the Poisson distribution and its parameters

1. Overstock: as stated in the previous paragraphs this situation occurs when the initial inventory of the SKU in the store overcomes the high service level target. In this case the objective is to move all the exceeding units to other stores ending with the HSL quantity in the sender store. As it is apparent from the graph, the probability to sell a high service level quantity is still low – and thus the chances to incur in remaindered stock is high - but it satisfies the customers' satisfaction constraint minimizing the expected lost sales.

2. Critical understock ($\mathrm{UU}_{\mathrm{si}}$ =1): the stock-on-hand for the SKU is insufficient for the store and would not be able to satisfy the expected demand of the upcoming period. Thus, the objective is to receive some units from other stores to fix the stock-out and, if possible, reach at least a quantity λ_{si} of units in the final stock to cover the average future demand and maximize the tradeoff between probable lost sales and .remaindered stock.

3. Non-critical understock ($\mathrm{UU}_{\mathrm{si}}$=0): understock with initial inventory higher than the expected sales λ_{si} ($\mathrm{UU}_{\mathrm{si}}$ =0). The stock-on-hand for the SKU is enough to cover future expected demand but still too low to meet the high service level requirement. In a system with unconstrained items availability the objective would be to move some units in these stores to fill in the gap between the starting inventory and the high service level quantity (restoring the *safety stock*). But in case of limited total resources it would be more reasonable to discourage the allocation in non-critical locations in favor of the very understock ones, spreading the available floating units to the highest number of critically understock stores which, otherwise, would have a very high probability of incurring lost sales. Indeed, handling this last situation differently the system could end up with some non-critical understocks that have reached their HSL quantity and thus have high probability of remaining with unsold merchandise at the end of the season, and some critical understock locations that keep on being too low in inventory and thus face dramatically high chances to lose sales.

Putting all the above together the model can be rewritten as follows:

$$Max\left(\sum_{i=1}^{N}\sum_{s=1}^{S}\left(m_{si}(1-U_{si})\sum_{j=1}^{N}x_{sij} - \alpha m_{si}U_{si}(1-2UU_{si})\left[\sum_{j=1}^{N}x_{sji}\right]\right) - \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t}^{T}C_{ijt}CT_{ijt}\right) \tag{3.11}$$

Subject to

$$\sum_{j}^{N}x_{ijs} \leq (Q_{is}^{0} - HSL_{is})\,(1-U_{si}) \quad \forall i = 1..N, \forall\, s = 1..S \tag{3.2}$$

$$\sum_{i}^{N}x_{ijs} \leq \left(HSL_{js} - Q_{js}^{0}\right)\left(U_{sj}\right) \quad \forall j = 1..N, s = 1..S \tag{3.3}$$

$$x_{iis} = 0 \;\; \forall i = 1..N, \forall\, s = 1..S \tag{3.4}$$

$$C_{iit} = 0 \;\; \forall i = 1..N, \forall\, t = 1..T \tag{3.5}$$

$$\sum_{t}^{T}C_{ijt} \leq 1 \;\; \forall i = 1..N, j = 1..N \tag{3.6}$$

$$\sum_{t}^{T}C_{ijt} \leq \sum_{s}^{S}x_{ijs} \quad \forall i = 1..N, j = 1..N \tag{3.7}$$

$$\sum_{s}^{S}w_{s}x_{ijs} <= \sum_{t}^{T}C_{ijt}\,b_{t} \;\; \forall i = 1..N, j = 1..N \tag{3.8}$$

$$x_{ijs} \in \mathbb{N}^{+}\forall i = 1..N, \forall\, j = 1..N, \forall\, s = 1..S \tag{3.9}$$

$$C_{tij} \in [0,1] \quad \forall i = 1..N, \forall\, j = 1..N, \forall\, t = 1..T \tag{3.10}$$

In this mixed-integer programming (MIP) formulation, notation for variables and parameters remains the same as the non-linear model, just like the constraints expressions.

The very difference is the objective function definition which is now linear and follows the logics expressed above. The first term represents the total units moved from overstock stores, that is the first goal to be maximized.

The second term is the quantity moved to understock stores and can assume different signs depending on the value of the parameter UU_{si}. Indeed, in case a store is very understock for a SKU, the relative parameter UU_{si} is equal to 1 and thus the term assumes a positive sign; consequently, the quantity moved to the understock will be maximized. On the contrary, if a store is not heavily understock, i.e. UU_{si} equals zero, it is not necessary to move additional units in stock hindering their allocation in more needing locations, so the transfer of units is not avoided but penalized through a negative sign before the quantity received.

Finally the last expression takes into account the movement costs as in the previous formulation.

As it is evident, there is no term in the objective function that accounts for the change in selling probabilities, and thus expected revenues, of each allocation decision. Thus, since the linear model is not able to discriminate if a transfer is worthy or not based on the difference in expected revenues and cost in the two scenarios – the current one and the eventual allocation-, it is reasonable to force the model to allocate all the units that can be moved instead of letting it choosing how many total units to transfer. Thus it is useful to add a constraint that fixes the total number of units moved in the problem, *E*:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{s=1}^{S} x_{ijs} = E \tag{3.12}$$

The number E of total units exchanged obviously takes into account the total units offered by overstock stores and the total units receivable by understock stores for each SKU. The constraint is written in an aggregate way for the whole problem without specifying the units to be transferred for each SKU; indeed, this would be redundant due to the presence of constraints (3.2) and (3.3) that already limit the units transferrable for each code.

The addition of constraint (3.12) de facto makes the first term of (3.11) redundant because the total units to be moved from overstocks are decided a priori. As a consequence the related term is constant and could be removed from the objective function. However, it was decided to formally keep it because it could turn to be useful in some cases to discriminate among the overstocks to be selected as sources of the transfers, as it will be explained in next chapters.

In any case, it is clear that the linear objective function value is no more meant as the expected profit generated by the reshuffling operation. Thus some coefficients have to be added to partly compensate the lack of information about the expected monetary gain and to make the benefits' terms and the costs' one comparable.

To this aim both the two first sums are multiplied by a parameter m_{is} which has a double function. First of all it is useful to scale up the terms with respect to the transportation costs; otherwise the solver could decide not to move units between stores because the mathematical balance of benefits and costs would be unfavourable, even if in reality the marginal units moved would bring an increase in expected revenues exceeding the required costs. Indeed, this could happen because, without the scale correction, every unit moved will increase the objective function value by 2 (1 unit moved from an overstock plus 1 unit transferred to an understock) at a unit transportation cost which can be higher than 2$, especially for small batches, leading to a negative balance.

The second purpose of the m_{is} coefficient is to differentiate the SKUs, weighting more the more profitable ones. Indeed, while in the raw model it would be the same to move one SKU or another, in reality it makes a great difference on the overall profit to transfer a high priced SKU with respect to a low priced one in terms of increase in expected profit.

For these reasons, even though it is not the unique way for calibration, it seems sound to set m_{is} equal to the price p_{si} or to the margin of the SKU s in the store *i*.

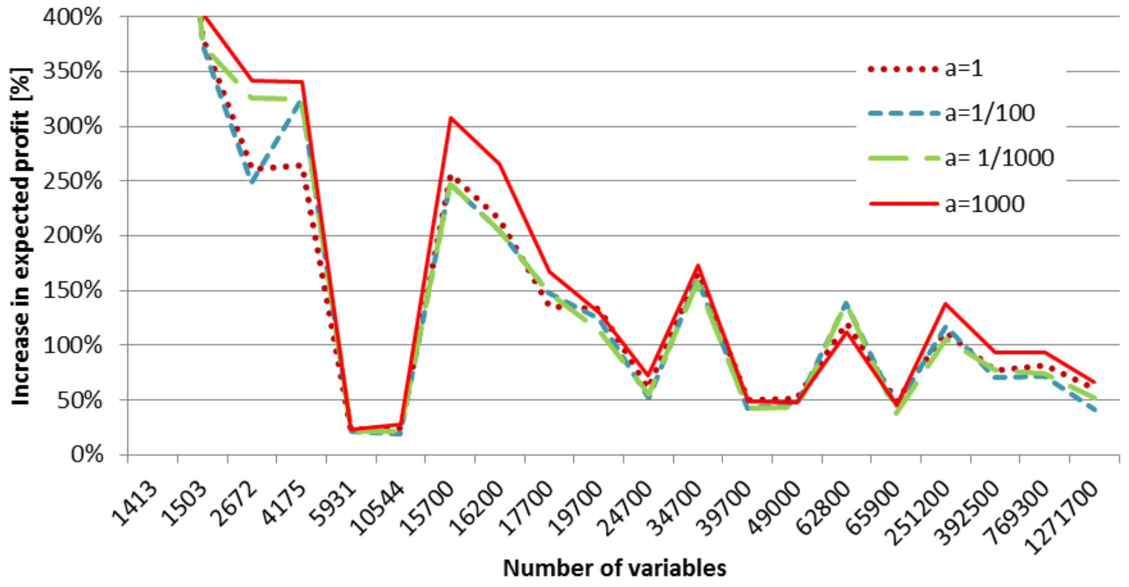
Furthermore, setting the weights of the two first sums equal means to totally penalize and practically avoid the transfer to non-critical understock locations, since moving to one of these destination would completely cancel the benefits of taking away stock from one of the overstocks locations. As it was mentioned before, this is judicious when one deals with a very limited resources situation.

However, in practice, the apparel firm can face more or less critical circumstances that can justify the relaxation of this penalization allowing also the allocation in non-critical understock stores. For this motivation the contingent coefficient α has been introduced: when α equals 1 or a higher threshold the penalty of moving to low priority understocks is maximum; on the contrary as α diminishes the disincentive for the allocation gets smaller and smaller.

4.4. Test Results

In this paragraph the results of experiments conducted to test the performances of the MILP model are presented. The instances of different dimensions have been built in R starting from real data provided by O.R.S, which have been used to compute the missing input parameters. Then the model has been coded in mosel language and solved by using the solver Xpress IVE 1.24.22., on an AsusTek PC with a Dual Core 2.16 GHz processor and 3 GB RAM.

The first trials have been done to verify overall if the solution proposed by the model effectively improves the actual situation in terms of expected profits. Such solution depends on the value of the parameter alfa in the objective function. Thus the verification was done using different values for this constant, such to determine how this number influences the solution of the model: five values, from very high penalty to almost no penalty, were chosen to cover a wide range of possibilities. Each variant of the model has been run on 20 instances, varying the number of stores (N) and keeping the number of SKUs (S) constant in some cases, and viceversa in others. The results of this preliminary analysis are reported in the table [tab.]. In the table, the generic instance named NnSk contains n stores and k SKUs.



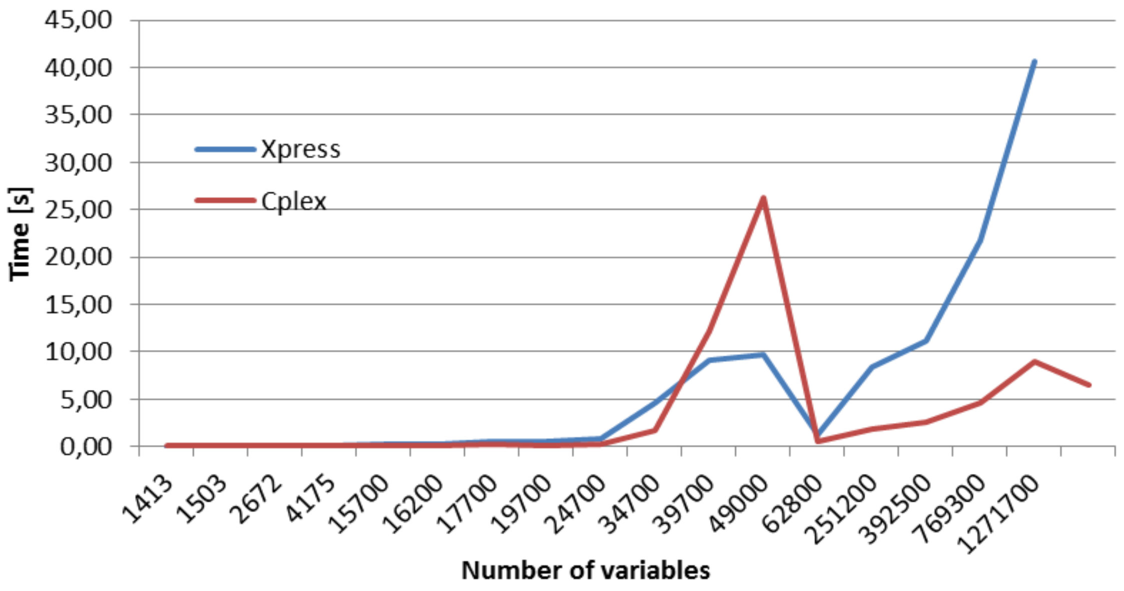
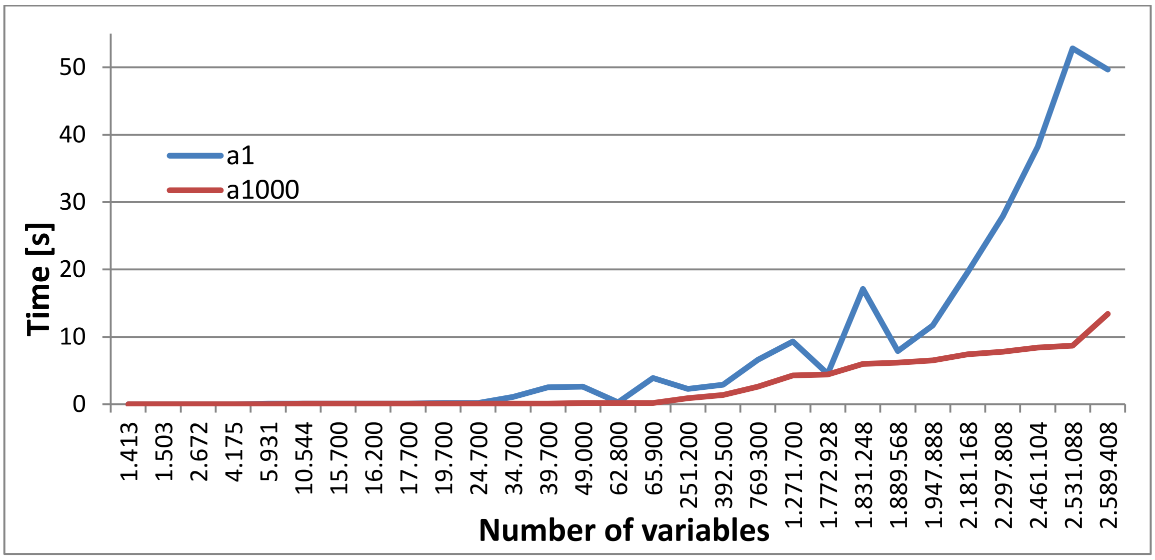


Table 1 Running times for instances with different sizes, stores and SKUs number

	N	S	# variables	Time*
N3S5	3	5	1413	0
N3S10	3	10	1503	0
N4S10	4	10	2672	0
N5s10	5	10	4175	0

N10S10	10	10	15700	0,1
N20S10	20	10	62800	0,3
N40S10	40	10	251200	2,3
N50s10	50	10	392500	2,9
N70S10	70	10	769300	6,6
N90S10	90	10	1271700	9,3
N108S10	108	10	1831248	17,1
N108S5	108	5	1772928	4,5
N108s15	108	15	1889568	7,9
N108S20	108	20	1947888	11,7
N108S40	108	40	2181168	19,6
N108S50	108	50	2297808	27,9
N108S70	108	70	2531088	52,8
N108S64	108	64	2461104	38,2
N108S75	108	75	2589408	49,7
N10S15	10	15	16200	0,1
N10S30	10	30	17700	0,1
N10S50	10	50	19700	0,2
N10S100	10	100	24700	0,2
N10S200	10	200	34700	1,1
N10S250	10	250	39700	2,5
N10S343	10	343	49000	2,6
N3S512	3	512	5931	0,1
N4S512	4	512	10544	0,1
N10S512	10	512	65900	3,9



Chapter 5

Heuristic approaches

The previous chapter has presented the exact model and a MILP formulation for the resolution of the problem through exact techniques through the Xpress solver. The chapter also highlighted the limitations of this approaches, which can be summarized in:

- the approximated objective function of the MILP model, who is not able to capture the information about the selling probabilities and thus optimize the expected revenues. In other terms, the direction of optimization of the objective function of the MILP model does not perfectly match the one of the original model, except for the transportation costs term.

- the fact that the model present a numerous set of variables which limits the dimensions of the instances that can be effectively solved to the optimum in a reasonable time by a linear optimizer, so it is not feasible to run the entire problem at a time.

This considerations encourage to find a way to reduce the dimension of the problem and/or decompose it in smaller sub-problems that differ for some characteristics which can be exploited to improve the allocation decision of the solver. On the other hand, they suggest to consider fester ways of tackling the problem by means of heuristic algorithms and compare the results of the different approaches.

5.1. Dimensions and Decomposition of the problem

In chapter 3, the case was presented to be very big and not entirely manageable with exact solution methodologies. In the performed tests, it was not possible to reach a solution through the optimization of the MILP model for instances encompassing more than 75 SKUs over all the stores or more than 10 stores over all 512 SKUs, keeping the number of slots tariffs equals to 147.

This is justified by the fact that this MILP formulation generates a lot of variables when the number of stores and/or SKUs in the instance increases. With the number of variables also the number of lines in the constraints matrix increases since it is strictly linked to the number of variables due to (3.9) and (3.10): the dimension of the whole problem is in the order of O(N2·T) or O(N2·S), depending on whether the SKUs number S is less than T, i.e. the number of slots theoretically applicable for each transfer (if one considers that there are T slots from 1 to T), with N the total number of stores.

These considerations are valid for the more generic case in which the analyst does not know the data in details, i.e. she is not able to predict in any way which tariff slots are not going to be used or to have a finer view of the starting situation of each SKU in the system. However, when the numbers are known and easily manageable it is reasonable to exploit some characteristics of the actual data to reduce the dimension of the entire problem.

There are three possible reshaping directions:

- Reducing the number of SKUs
- Reducing the number of stores
- Reducing the number of slots of each tariff

5.1.1. Stores set

Reducing the number of stores implies inevitably reducing the SKUs set to the codes that can be exchanged between the stores of the predefined subset, thus excluding all the other items. Therefore, if all the relevant SKUs to be reshuffled have to be included, no store can be excluded from the stores set of the whole problem formulation. Indeed, the actual store set has already been filtered during the instance construction phase, reducing the total number of locations from 116 – the total Retail stores in the raw data set - to 108 relevant Retail stores.

5.1.2. SKUs set

For what concerns the SKUs set, further analysis can be done to remove some SKUs from the problem, based on the actual situation of the single size at the moment of the reshuffling. For instance, the data used until now included all the SKUs which are overstock or understock for at least two stores, without checking if they could be actually exchanged some way, i.e. if there is at least one overstock store and one understock store in the system for such SKU. Adding this filter to the dataset, it has been possible to reduce the overall SKUs set to 390 codes.

Moreover, further analysis found that among this new SKUs set there are some items whose allocation is obvious, which means that there is exactly one understock store and one overstock one in the system for that size. In the case on hand, there are only four items belonging to this category; evidently, this fact only marginally impacts the dimension of the problem since the SKUs number is reduced just by 1%, however it is relevant because it strongly influences the allocation of the other items. Indeed, the presence of stores pairs which have to incur in a positive transportation cost regardless of the other allocation decisions modify the marginal transportation costs for moving additional units between these locations. As a matter of fact, in a model starting from scratch the incremental transportation costs for moving one unit from I to j in the first instance– for a SKU with weight lower that the first slot bound, which is the case for all SKUs – equals 8,3 dollars for each pair (i,j); on the contrary, if some pairs (i*,j*) already bear a transfer cost of 8,3 $ due to the first obvious allocation, the incremental cost of transferring one unit between these stores will be zero or slightly larger due to the cost structure, thus making this transfer choice preferred to the others in first approximation.

5.1.3. Problem decomposition

The remaining 386 SKUs can be divided into two sub-groups based on the units availability in the system: the ones for which the whole system can be considered understock and the ones for which the units available in the system are enough to satisfy the requests of the understock stores. In other words, for the first group of SKUs the total number of units that can be moved from the overstock stores are less than the total units needed by the understock stores to reach their target inventory level, so some stores will remain understock also after the reshuffling of units. In the second case, instead, the overstock units that can be re-allocated are enough to cover the understock stores' requests and fulfill the high service level target in all these stores.

This distinction is useful because the two sub-problems can be handled in slightly different ways tailoring the MILP model with small adjustments to enhance the allocation decisions. Indeed, for the first SKUs set it is known that all the overstock stores will get rid of their excess units, so the first term of (3.11) is redundant, while the second one remains important since, first of all, the allocation in non-critical understock stores should be discouraged due to the scarcity of resources in the system and, secondly, in theory the critical understock destinations could be weighted in some way to take into account their different selling probabilities and initial inventory situation.

On the contrary, in the second problem it is known that all the understocks will receive the units needed to reach the High Service Level quantity so there is no need to penalize the allocation to non-critical understock stores – i.e. the second term of (3.11) is redundant - while not all the overstock locations will send all their excessing units. Therefore for this sub-problem the first term of (3.11) could be modified introducing weights to discriminate among the possible origin stores and prefer the more convenient ones. For instance, it is reasonably more profitable to move units from stores that, after the transfer, would remain with a positive final inventory (HSL) rather than from overstocks that have a HSL equal to zero – because the expected demand is very low – and thus would have a negative change in expected profit after the complete reallocation of its overstocked units. This adjustment to the MILP objective function has been proven to be successful by the performed tests, whose results are reported in Chapter 6 where it is also explained how the weights have been computed.

5.1.4. Tariff slots number

As far as the cost variables are concerned, it is necessary to understand if all the slots t can be actually used in the solution. To answer this question one can consider the stores pair which, in principle, can exchange the heaviest lot based on the starting situation of the relative SKUs; if these stores effectively exchange such quantity, the transportation cost would be determined by a certain tariff slot t*, which is the minimum t such that the relative weight bound b_t can contain the lot weight which, as said, is the maximum possible weight transferred between two stores in the whole problem . Therefore t* can be considered as the highest possible slot applicable, since all the other stores pairs will exchange a weight equal or lower than b_{t*}, thus employing at maximum the tariff slot t*.

Talking about the actual data on hand, such t* equals 70, which means to cut down the slots number by roughly 52 %. This remains a high number for the T parameter, however it can be further reduced for the single sub-instances, as it strongly depends on the SKUs and stores involved in each sub-problem.

Another way to simplify the modelling of the transportation costs can be the linearization of the seven costs functions since, as it is apparent in fig. ,in first approximation they resemble seven upward sloping lines with different slope coefficients. This approach would permit to get rid of the variables C_{ijt} and write the costs part of the objective function as a linear 0-intercept function of the lot weight, with different proportionality coefficients for the different destination zones.

This approximation works quite well for large weight lots, but it strongly distorts the results when lower weighted lots are employed, which is the most frequent case. Indeed, the approach was tried out with very scarce results in terms of quality of the solution. The issue lies in the presence of a base cost of 8,3 $ for the first slots, that should be modeled differently from a linear function. Theoretically, one should be able to design a model that takes into account if a lot moved between two stores is "large" or not relatively to the zone tariff: if the lot weight is "small" the transportation cost should equal the base cost plus a correction factor to account for the extra costs of carrying a weight over the b_{t0} – i.e. the maximum weight bound to apply the base slot cost-; if the lot weight is "large", instead, the linear approximation is acceptable. Implementing this kind of reasoning would likely mean introducing new variables to classify the transfer lot from store i to destination j , thus reducing the benefits of the linearization itself – i.e. eliminating the transportation cost variables and speeding up the linear programming model resolution.

For this reason and for the fact that the model already approximate the exact problem formulation, it has been decided to opt for the first strategy, i.e. the evaluation of the maximum number of tariffs through exact computation of t*.

As a matter of fact, the MILP model is itself an approximation of the real problem, because it misses the information about the expected revenues of the various alternatives. However, it is perfectly able to determine in each moment the transfer with minimum cost. Therefore, the linear approximation of costs would further weaken the ability of the model to make decisions that converge toward the optimum of the original problem. Moreover, the cost structure - made of seven stepwise functions with constant weight pace but highly irregular incremental costs, in addition to a starting tariff which is constant only for the very first slots - is one of the characteristics that make this problem "hard" to solve and neglecting this complexity would likely bring to a misrepresentation of it.

5.2. Greedy Algorithm for the problem

As highlighted in the previous section, the problem is too big and the number of possible alternatives so high that it is unthinkable to apply whatever algorithm that enumerates and evaluates all possible solutions to reach the optimum, because it would take practically an infinite time to run. Indeed, for each origin store it is necessary not only to establish to which destination the surplus of items should be moved, but also how many units of each SKU to transfer to each destination store, without forgetting the impact of each of such choices on the transportation costs for every origin- destination pair. Each variations of these possibilities creates a new alternative that should be evaluated in order to obtain a solution proved optimal, generating, with the increase of the stores/SKUs number, a so called combinatorial explosion.

In these cases the strategy that is usually implemented is a constructive heuristic, which normally allows to reach a good solution in reasonable times, but that is not able to guarantee the optimality of the solution neither to give an estimate of the distance of such solution from the optimum.

5.2.1. Greedy logic

A greedy algorithm builds the solution taking one choice at a time, maximizing at every moment the small portion of problem it can envision.

In the greedy logic for the present case, at each time the algorithm considers a single SKU and the related sets of origin and destination stores and allocates one unit at a time selecting the alternative which presents a higher expected profit at the moment of the decision. Clearly, at each iteration, the transfer lots weights are updated as well, modifying the costs increments of the future alternatives that will be evaluated.

Since the SKUs are processed one after the other and the lots' weights are updated at every decision made, the order according to which the SKUs are processed is not irrelevant since the choices made upstream drive the allocation choices downstream, due to the modified incremental costs, and thus affect the overall solution. Therefore, it has been chosen to give priority to the most valued SKUs, i.e. the products with the highest price[8] because they will likely contribute more to the final overall expected profit and thus optimizing their allocation is most important. As a consequence, the SKUs have been sorted by decreasing price and sequenced following such order. As the tables results in Chapter 6 show, this choice emerged to be beneficial with respect to a random sequencing both in the application of the greedy and in the generation of the initial solution for the matheuristics, that will be explained in the relative paragraph in the next section. In any case, the algorithm starts from the results of the pre-allocated "obvious" SKUs discussed in the previous section.

In paragraph 5.1. the whole problem was divided into two sub-problems with different features. Therefore, it is reasonable to apply a slightly different greedy algorithm to the two problems to exploit each case's features and enhance the results, both in terms of expected profit and running time.

As far as the first problem is concerned, i.e. the one including the SKUs which are overall understock in the system, it is known that all the overstock units in the relative stores will be reallocated to understocked destinations, which in turn will

[8] In practice, it would be better to use the contribution margin but , above from the numbers, the logic remains unchanged.

not be all served fully. Thus, the question is to identify the destinations that are more worthy to be chosen and the number of units allocated to each of them.

At every step, the decision where to move a unit is taken computing, for each destination, the expected increase in profit (EP) after the allocation of one additional unit to the store stock. Such expected change in profit equals the difference among the expected revenues - in the scenario in which the unit is allocated to the store - and the present expected revenues of the SKU in the destination, minus the difference in transportation costs in the two scenarios. Hence, for each SKU, for each origin store and for each unit to be moved from it, all the possible destinations are ranked according to the expected profit and the unit is allocated to the most promising one. In synthesis, the procedure for the first problem can be stated in the following pseudo-code:

```
1. FOR s in SKUs set of the first problem
    a) FOR i in origin stores for the SKU
        • WHILE there are overstock units in the origin store
            o FOR j in destination stores that are still understock for the SKU
                - Compute the EP= (expected revenues of the actual quantity + 1) - (expected revenues of the actual quantity) – (cost of moving one additional unit from i to j – present transportation cost from i to j)
            o j* = destination with max EP
            o Allocate one unit to the most promising destination j*
            o Update the inventory state for store j*
            o Update the lot weight between the origin store i and the chosen destination j*
            o Update the inventory state for store i and thus the
              remaining overstocked units
```

For the second problem, the logic is equivalent but the starting point are the understock stores, because it is known that they all will receive the units needed and thus will be destination of some transfer. The question is to understand which origin store will serve each understock store, which quantity will be transferred and, eventually, which overstock stores will remain overstocked and of what amount. The expected profit (EP) for each origin to be selected is computed in similar way as before and the store with higher benefits will be chosen to transfer the unit. Thus the pseudo-code reported before can be modified as follows:

```
1. FOR s in SKUs set of the second problem
    a) FOR j in destination stores for the SKU
        • WHILE there are understock stores
            o FOR i in origin stores that are still overstock for the SKU
                - Compute the EP= (expected revenues of the actual
                  quantity - 1) - (expected revenues of the actual quantity)
                  – (cost of moving one additional unit from i to j –
                  present transportation cost from i to j)
            o i* = origin with max EP
            o Allocate one unit from the most promising origin i*
            o Update the inventory state for store i* and store j
            o Update the lot weight between the origin store i* and the
              destination j
```

5.2.2. Computational complexity

The computational complexity of a generic algorithm can be computed as the number of iterations made times the complexity of each single iteration, i.e. the number of elementary operations that are executed at each step.

As said, the algorithm starts acting on a set of SKUs, thus the complexity is firstly dependent on the number of SKUs. Secondly, the procedure run until all the units that can be transferred are actually allocated, so the second determining factor is the number of exchangeable units in the problem. Obviously, also the number of stores in the origin and destination sets matter to determine the computational complexity of the algorithm.

Basically, the number of iterations coincides with the number of SKUs. Then, for each origin/destination, for each unit to be allocated from/to the each origin/destination, an expected profit has to be computed for each destination/origin store. The computation of this number takes constant time. Subsequently, the maximum EP should be found in the destination/origin set; the sorting algorithm takes O(nlogn) operations, where n coincides with the number of destinations per SKU (D) in one problem or the number of origins per SKU (O) in the other. All the he next updating operations in the cycle take constant time.

Therefore, the overall running time of the greedy algorithm – for instance, for the first problem – is estimated as $S*O*D*u*DlogD = S*O*u*D^2*logD$, where S is the number of SKUs in the problem, O the number of origin stores for SKU (on average), D the number of destination stores for SKU (on average) and u the number of moveable units for SKU (on average). Hence, in the worst case the complexity is $O(S \cdot N^3 \cdot u \cdot \log N)$ where S and N are respectively the total number of SKUs and stores in the whole problem.

Actually, in practice, the inner loop of the algorithm, that allocates one unit to a destination, does not run over all the destination set every time since, as soon as the units are allocated allocated, the status of each store is updated and then the stores that are fully replenished are progressively removed from the list of destinations. Vice versa, in the second problem the overstock stores that have moved all their excessing units are gradually excluded from the origins list of the loop. This process, together with the fact that it is unlikely that all the SKUs are present in the majority of stores - and thus the total store number in the destination/origin set is much lower than the maximum number N-, in practice reduces the running time with respect to the worst case of complexity O(S·N^3·u·logN).

Indeed, from the data on hand it has been calculated that the average number of overstock stores per SKU is 4,2 for the first SKUs set and 24,88 for the second one, with a maximum of, respectively, 24 and 91 stores. As far as the number of understock stores are concerned, the average number per SKU is 17,65 for the first sub-problem and 5,09 for the second with a maximum of, respectively, 55 and 27 stores.

These numbers justify the choice to cycle on the overstock stores list in the first problem and on the understock ones on the second, to contain the overall number of operations made by the procedure.

The greedy algorithms presented in this section are a good method to obtain a solution to the problem in short times. Like all the constructive procedures, this heuristics does not give any information about the goodness of the results as well. However, given the dimension and the difficulty of the problem, it is reasonable to think that such solution is not the best possible and thus it is possible to obtain better results employing more sophisticated techniques .

5.3. Matheuristic approaches

As anticipated in Chapter 3, the resolution techniques for the combinatorial optimization problems are basically divided into two big categories: the exact ones, which can find the best possible solution and demonstrate its optimality but with an higher computational complexity, and the heuristics, which are able to get a good solution within a short time limit.

Recently, a new family of algorithms based on the hybridization of the two approaches is rapidly rising among the communities of researchers. These are the so called “matheuristics”, which will be first described in the next introduction paragraph and then used to deal with the present problem.

5.3.1. Introduction to Matheuristics

Typically combinatorial optimization problems can be modelled as Mixed Integer Programming (MIP) problems and the exact resolution methods are therefore applied to the relative formulations of the MIP models.

However, exact methods show a certain number of disadvantages. First of all, for several problems the dimension of the instances that can be actually solved is limited: indeed, the variance of the CPU time is normally very high even when applied to different instances with the same dimension for a certain problem, and often it grows extremely as the instances’ dimension increases. Hence If the optimal solution cannot be computed in an efficient way in practice, often the efficiency is preferred to the guarantee of optimality. In other terms, the certainty of finding an optimal solution is overlooked in favor of a good solution found through heuristic methods within reasonable times. With regard to this, neighborhood search based methods are likely the most effective class; indeed, when integrated in higher level mechanisms like in the metaheuristics, these approaches have been proven to be definitely effective to reach solutions close to the optimal one in a large number of difficult problems. However, there no exist modelling frameworks which are able to handle definitions and representations of heuristics problems, i.e. there are no general-purpose metaheuristics solvers available on the market.

Based on what stated above, it is not surprising that traditionally exact methods and heuristic ones have been considered strongly different and separated, but it is likely not completely true. Metaheuristic algorithms and MIP both have their pros and cons, but they soul dot be considered incompatibles. Actually, due to the complementarities of their features it seems more reasonable to combine these two techniques in even more effective algorithms, able to exploit the advantages of both approaches, trying to avoid their disadvantages as much as possible.

The hybridization of different techniques is already common and consolidated practice in the field of metaheuristics, but in the last years the same process has been applied also to exact methods and methaheuristics, giving birth to a new class of resolution techniques called “Matheuristics”. Matheuristics are generally based on the idea of taking advantage of the strong points of both approaches, but there does not exist an unique classification or a consolidated methology in such field to give a more precise definition. A distinctive feature common to all the matheruristics is the use of a mathematical model, or some tool derived from it, in some way in a heuristic type structure, that is why these techniques are often called “*model-based metaheuristics*”; but the structures used or the degree to which the model tools are employed can be totally different from an approach to another. For instance, there are cases in which the metaheuristics are used directly within the linear programming procedures that use search trees, where efficient metaheuristics are often used to get good solutions that permit to prune the possibilities tree faster. Other cases use the opposite process in which exact methodologies are used within the metaheuristic algorithm for the resolution of sub-problems or for an optimal exploration of neighborhoods.

So, most of the cases can be grouped into two big categories:

1. Matheuristics in which heuristics methods intervene within an exact method. This case requires that the optimization algorithms can be modified integrating heuristics strategies able to speed them up and allow them to handle bigger instances.

2. Matheuristics in which exact methods are used within an heuristic algorithm. This second case is simpler to be realized since the base components, i.e. the (meta)heuristic framework and the mathematical model, have already been implemented.

The matheuristic algorithm that is being discussed in the next paragraphs used the last approach.

5.3.2. A Matheuristic for the reactive allocation problem

In chapter 4 a Mixed-Integer linear programming model for the present problem has been presented to solve the allocation problem, in place of the non-linear original model. This MILP model however, given the complexity of the problem and the resulting impossibility to produce a more efficient formulation (for instance, eliminating the binary variables to model the transportation costs) , is solvable only for instances of limited size. Actually, this limitation exists only in the case one wants to reach the best solution. Indeed, if the optimality is ignored, for instance setting a time limit, the solvers can handle problem of larger dimensions and normally they are able to get feasible solution of good quality. In the extreme case, it is possible to get at least an information on the lower bound of the problem, solving the continuous relaxation[9] of the MILP problem, which can be done by the linear optimizers (LP solver) in a small amount of time even for big sized problems.

However, solving the linear relaxation of the present MILP model is quite senseless: allowing both variables, x_{ijs} and C_{ijt} to be real would induce the model to set most of the x variables to the maximum integer value they can get to maximize the transferred units in the objective function, while the values of the Cs, which in principle should drive the decision on which transfer to activate at what cost, would be arbitrary set to the lowest real number possible to minimize costs. So, the two kind of variables would lose their interconnection, which is the real core of the problem, and thus the resulting solution would be comparable to a random generated one which respect all the constraints. A more sensible approach to the relaxation could be the partial relaxation, i.e. the relaxation of

[9] A relaxation of a problem is a version of the problem with some requirements or constraints removed ("relaxed"). This approach is used in branch-and-bound algorithms, for instance, relaxing the constraints on the integer nature of variables (continuous relaxation); the solution found by these methods is not necessarily a solution of the original problem, but a solution of the original problem is a solution to the relaxation. It follows that in the case of a minimization problem, the value of the optimal solution to the relaxation is a lower bound on the optimal solution to the original problem, even if this bound is a not-necessarily-tight lower bound.

only the variables x, to preserve the information and consequent decision on the transportation costs; however some trials have shown that, although this last approach frequently gets to the optimal solution (i.e. the lower bound is tight with respect to the optimum), it is not convenient in terms of time, since the partial relaxed version of the model takes the same time or even more with respect to the original version, in which the x variables can assume limited integer values because of the constraints (3.3) and (3.2) instead of infinite real values in the same range.

Initial Solution

In the previous paragraph it has been noted that the continuous relaxation technique is not useful in this case to get a good initial solution with considerable savings in running time.

On the contrary the first method, i.e. setting a time limit in the solver, is fully applicable to get a decent initial solution to start with and its results are reported in the results table at the end of the Chapter.

However, a different method to generate the initial solution has been chosen, simply because it has shown to produce better results than time-limited optimization with the same total running. In particular, the whole problem has been subdivided in the two sub-problems stated in paragraph 5.1; the SKUs of each sub-problem have been ordered by decreasing price as done in the greedy algorithm of 5.2.

The initial solution has been built starting from the sub-problem in which overstocked units for the SKUs are scarce in the system. The SKUs has been divided into smaller subsets containing between 5 and 25 SKUs per instance – according of the relative difficulty of each sub-problem, approximated with the total number of units to be moved, which works as a proxy for the allocation choices number – following the decreasing price order. For each SKUs subset the relevant store subset has been identified to be part of the instance and the maximum slot number t has been computed in order to contain the overall dimension of each instance. Indeed, each instance size should be small enough to be solved quickly, but as high as possible to enhance the allocation choices of the solver, overcoming the myopic ones of the greedy, which only considers one SKU at a time.

Then, the solution is created in a constructive way: starting from the highest priced SKUs set the first instance is solved taking into account the initial lots weights – the ones derived from the allocation of the "obvious" SKUs - and the resulting new lots weights are saved; then the second instance inherits the starting lots weights from the first one, optimize the allocation based on this information and saves the lots solution to be inherited by the following instance and so on. Basically, each instance is linked to the previous one by the information about the lot weights that are already being transferred from an origin to a destination, which influences the incremental costs of the stores pairs which already exchange units and thus the allocation decisions of the subsequent instances. This approach can be seen as an extension of the greedy logic reported in 5.2. with the difference that in this case more than one SKU is considered ad a time and that the information about the probabilities/expected revenues variations is not present in the linear model.

Such procedure is then extended to the second sub-problem, the one in which the units in excess in the overstocked stores are abundant with respect to the request of the understock stores, in the same way starting from the lots weights generated at the end of the first problem. For simplicity the following discussion will focus mainly on the first sub-problem, mentioning the second one when necessary.

Heuristic Structure

In the previous paragraph it is described how to get a good initial solution of the problem, however there is no guarantee of optimality for this current solution. It is likely that such solution can be improved perturbing it and re-optimizing selected subsets of variables. This means optimizing the problem fixing some variables to a value, for instance the one of the initial solution, and let the solver runs and explores the solution space to optimize the portion of the solution variables left "free". The resolution of this simplified model can bring to a solution which is better than the previous one.

The matheuristic implemented here is based exactly on this procedure, iterated several times in a local search framework. The neighborhood is represented by the non-fixed variables, while the exploration and evaluation of the goodness of each neighbor is done by the solver itself which, evaluating all the possibilities and returning the best one of the possible solutions, is basically acting like a best improvement type local search.

The window of non-fixed variable should be large enough to allow a relevant search and escape the local optimum, but quite circumscribed such to avoid every simplified model to run for long time. In the extreme case that also the simplified MILP requires too much resolution time it is possible to set a time limit, hence turning the search strategy into a first improvement like local search. However the first approach, i.e. the limitation of the window dimension rather than the running time, has been preferred.

The heuristic procedure can be summarized as follows:

1. Initalization:
    a. Construction of an initial solution *sol* (through the constructive approach explained in the initial solution paragraph)
2. REPEAT
    a. Using the solution *sol* some variables of the MILP model of the problem are fixed
    b. The MILP model is solved by the solver obtaining a new solution *sol_new.*
    c. Since the objective function of the MILP does not coincide with the real objective function of the problem , through which the goodness of the solutions is actually evaluate, *sol_new* is not necessarily an improved solution. However, if this new solution is effectively an improved solution, the current best solution is updated setting *sol* = *sol_new.*

UNTIL all the predefined neighborhoods have been explored

Neighborhoods

The procedure just outlined in the previous paragraph is itself simple and easily to be implemented, but is yet necessary to define a critical point for its effectiveness: the strategy of selection of the variables to be fixed. Basically, these variables represent the neighborhood of the local search and thus have a strong influence on the algorithm results.

As mentioned in the previous paragraph, since an exact method is employed, the neighborhood cannot be too large otherwise the resolution duration would be again problematic. On the other side, it should contain as many feasible solutions differing from the starting one as possible, such that the algorithm can explore them, choose the best one and, this way, obtain an improvement.

The chosen formulation contains a large number of integer and binary variables, part of which have to be fixed. Basically, the variables on which the strategy can theoretically operate belong to two groups: the integer variables x_{ijs} and the binary ones C_{ijt}. As it was explained in the Chapter concerning the MILP model, the variables of the second group are logically linked to the ones of the first type through logical implication links. This means that, once the values of the variables x_{ijs} are determined, it is possible to derive the values of the variables C_{ijt} by only using such logical constraints and the fact that , for each pair (i,j) of stores, the solver will always choose to “activate” the slot with the lower t to minimize transportation costs.

At this point, it is reasonable to adopt a similar behavior in the construction of the neighborhood: the strategy of variables selection will only consider the x_{ijs} variables to be fixed, while the variables C_{ijt} will be left free or fixed depending on the decisions made for the first type variables.

Indeed, if one decides to leave some variables C_{ijt} free in a complete random way, it is likely that these will refer to different constraints and thus the solver, in the attempt to modify their value, would probably obtain unfeasible solutions and thus would return a solution which is identical to the previous one.

The strategy should aim at selecting variables which are someway related one to another, so that the software has maneuver margin to obtain solutions which differ from the original one. The relation to allow an effective exploration of other possibilities can be basically defined along two dimensions: the stores set and the SKUs set. For instance, one can decide to free all the variables concerning a specific stores subset or a specific group of SKUs. Alternatively, it can be decided to contemporarily free some SKUs and some stores and re-optimize the simplified problem. However it is difficult to arbitrary define subsets of stores which are connected enough to create alternative feasible solutions to be explored in the neighborhood. For instance, once one choses two stores to be part of a subset, then she should consider, for each SKU that can theoretically be exchanged between the two stores, at least a third store that is exchanging such SKU with one of the two locations such to assure the presence of at least an alternative allocation of the relative units. Although it is a laborious process and it is difficult to contemporary keep the dimension of the window under control and to make the exploration as complete as possible. That is why it is preferable to define the neighborhoods as subsets of SKUs whose related variables are set free.

Selection of the perturbation zone

Once the neighborhood structure has been defined, it is necessary to decide which zone of the solution is more convenient to be unsettled, i.e. which regions of the solution space have a higher potential to be improved. Indeed, given an initial solution, there are regions able to generate improvements when unsettled and others that are too bounded by the constraints to actually produce further interesting solutions.

One of the simplest ways to choose such areas is to randomize the choice, even if it means often looking over “useless” neighborhoods. As far as the present case is concerned, the problem is so bounded and the interconnections among the stores-SKU pairs so complex that a random approach is likely to produce poor results. As a matter of fact, in the tests performed randomly no improving solution was found and thus these approach has been discarded in favor of a more effective one.

A different method, which has more chances to supply improving neighborhoods, consists in selecting the SKUs to be set free starting from the ones that are present in the highest number of stores and/or have an unbalanced number of overstock and understock stores, in particular when the number of source stores is much lower than the number of possible destinations. This choice is justified by the fact that for these SKUs the number of understock stores which are excluded from the allocation for scarcity of resources is potentially large. Therefore it is expected that a release of the related variables can bring to the identification of a better distributions of the units among the understocks, excluding less profitable destinations - both in terms of expected revenues and cost savings - in the light of the transfers of all the other SKUs of the whole problem that, as usual, influence the marginal transportation cost for each origin - destination store pair.

In practice, this reasoning has been implemented in two slightly different subsequent ways, essentially dividing the procedure in two phases which differs by the criterion of choice of the perturbation region.

In the first phase, at each iteration the SKU which is present in the highest number of stores and that has an unbalanced number of overstock and understock stores – with more understocks than overstocks - has been selected. Then, it has been checked if there were other SKUs which are sold in the same stores or in a subset of the them; the idea behind this last choice is to enlarge the window of released variables to allow the solver to explore a wider portion of the solution space which is someway connected – in this case the SKUs share part of the stores set - and improve the solution, for instance pooling the transfers among store into bigger batches thus reducing costs. Afterwards, all the x variables relative to the just defined SKUs set have been left unestablished along with all the variables C_{ijt} (for the reasons explained before), and the model have been optimized by the solver to get a new solution and go ahead with the heuristic procedure.

Once the whole set of unbalanced SKUs have been examined, the definition of the region to be re-optimized has been slightly changed to explore further possibilities to improve the solution. During the first phase the neighborhoods, i.e. the relative released SKUs, that have produced an improvement with respect to the initial solution have been saved in a list of *improved_SKUs*, because for these SKUs the initial allocation have shown to be not optimal and thus it is sensible to challenge it in different ways, also combining one with another in the release. This is the incipit of the second phase which starts from the best solution of the previous stage and define the initial zone to be released as the first SKU of the *improved_SKUs* list; if this perturbation results to be effective, the current released SKUs set (*released_SKUs*) is maintained and another SKU of the list is added to it, otherwise the last SKU added before the unsuccessful perturbation is removed and a new one is added from the *improved_SKUs* list. The logic for the determination of these new reshuffling zones is summarized below:

Initialization:

- The *released_SKUs* is set initially equal to the first SKU of the *improved_SKUs* list. i=1 (the index i represents the number of the iteration and, consequently, the index of the last SKU added to *released_SKUs* from the *improved_SKUs* list at that iteration)

- The best current solution , *best_sol,* is set equal to the best solution found at the end of the previous phase

REPEAT

- If the solution found with the i-th perturbation, sol_i , is better that the current best solution *best_sol,* this last is updated setting *best_sol* = sol_i ; else the last SKU added to the *released_SKUs* set before the i-th iteration is removed, i.e. *released_SKUs* = *released_SKUs* - *improved_SKUs[i];*

- In any case a new SKU is added to *released_SKUs* from the *improved_SKUs* list: *released_SKUs* = *released_SKUs* U *improved_SKUs[i+1];*

- $i = i + 1$

UNTIL all the *improved_SKUs* list has been visited.

Therefore, at each iteration the subset of SKUs that are released is expanded if the latest solution improve the current best one or modified if this does not occur. This is done to limit the dimension of the re-optimization window which potentially can grow until it encompasses the whole *improved_SKUs* set, even if it is unlikely that all the SKUs will become part of the final released set since, as the window dimension grows, also the probability to escape the local minimum and incur in a worse solution increases.

As far as the computational complexity is concerned, it is evident that number of iterations is strictly defined by the number improving SKUs in the *improved_SKUs* list, which in turn depends on the initial set of "unbalanced" SKUs chosen in the initialization of the first phase. This implies that the whole procedure will run for O(S) times and thus the whole matheruristic will take a time roughly equal to O(S)**average resolution time of the simplified model.*

Chapter 6

Results of the tests

The present Chapter reports the results on the tests made on the heuristic procedures explained in Chapter 5. After an introduction about the data gathering process and how the instances for the tests have been constructed, the tables showing the comparative results of the different methods are presented and commented.

6.1. Data gathering and instances construction

In order to compare the performances of the proposed models we real data from an international Fashion Company which works with hundreds of franchising and direct operated mono-brand stores were used. The data collected from the above-mentioned company concern the North American market, in particular the United States. Data used for the experiments and instance construction were limited to the Fashion clothing items and to the Retail stores.

The sales reports of the past year, which are records that demonstrate information on how the articles performed in terms of sales at the colorway level, were the key source in the empirical data gathering. To split the colorway quantity sold into single SKU's sales the Size distribution report , which records the statistical frequency of each SKU sales in the colorway assortment, was used. To simulate a real situation, only the colorways that are intended to sell – i.e. which are planned to be allocated in the stores in a given period – in the current season, corresponding to the whole Spring/Summer (from February to August), where considered. The current date, at which the reshuffling operation is simulated to occur, was arbitrarily selected to be the 22 June of the current year; such date is far enough from the beginning of the season to justify the reallocation – due to a likely scarcity of resources in the warehouses - but quite far from the end of the season to allow the operation to produce significant effects, considering also that the transportation lead times can be relevant. For the SKUs related to the intended-to-sell colorways, the expected sales and the high service level quantity were computed as well as the inventory level at the current date.

As far as the sales probability is concerned, the lambda of the Poisson distribution for each SKU was calculated as the mean of the quantity sold of the SKU from the starting of the current season to the current date (all days were considered in the calculation, since the American stores never close). The choice to focus only on the present season data is legitimate by the fact that several Fashion product, being strictly seasonal and linked to the current fashion trends, have never been sold before the current season. Clearly, this is not the most sophisticated and precise way to estimate the mean of the distribution and indeed, in reality, it is done through scenarios simulation and sampling; however, for testing the goodness of the models there was no need to have a precise estimation but just a realistic one. Once the lambdas for each SKU have been found, the corresponding high service level quantities were calculated as the quantity corresponding to the 95-th quantiles of the Poisson distribution having the λ of the SKU as parameter.

For what concerns the inventory data, the stock position at the current date for every SKU in the locations was drawn from Inventory reports of the stores; these datasets record the inventory changes in the store's stock by SKU code.

In the created instance, the weight of the articles is defined at the class level, i.e. all SKUs belonging to the same class are considered to weight the same, while the price extracted from the sales reports is considered constant for the single code over all stores. Obviously, in reality the price of a single SKU can vary across the stores, for instance because of promotions, however the mark down amount is usually limited in the Retail channel and, in any case, this approximation does not affect the model performance evaluation.

Lastly, the tariffs and zones information were provided by the logistic carrier in charge for the transfer.

All that said, the SKUs which were in-stock ,and thus have no need to be reshuffled, were removed from the input data to avoid creating uselessly big datasets and only the codes for which all the required data (class, colorway, sales, inventory…etc.) were available were included in the final version of the instance inputs.

6.2. Experimental results

In this section the results of computational experiments conducted to compare the performances of the various approaches explained in the previous chapters are presented.

Most of the procedures were tested on an AsusTek PC with a Dual Core 2.16 GHz processor and 3 GB of RAM. The most memory consuming processes, including the MILP run on the whole problem with time limit and the matheuristic, run on an Acer computer equipped with a Quad Core 3.40 GHz processor and 8 GB of RAM. The MILP models were coded in Mosel language and solved by using the ILP solver Xpress IVE 1.24.22 while the constructive algorithms were written in R language using the RStudio software version 1.1.423.

Table 2 Results of the tests made on the set of SKUs for which the total units in the system are insufficient to cover all the understock needs.

	Expected Revenues [$]	**Costs [$]**	**Expected Profit [$]**	**ΔER [%]**	**ΔEP [%]**	**Time [s]**
Greedy (random order)	26018,71	4070,41	21948,30	78,46	50,54	59,15
Greedy (price order)	26073,45	3863,92	22209,53	78,84	52,34	62,67
MILP with time limit	25.338,87	2571,51	22767,36	73,80	56,16	127*
Initial solution (random order)	25333,05	2750,08	22582,97	73,76	54,90	133
Initial solution (price order)	25672,86	2688,85	22984,01	76,09	57,65	126,6
Matheuristic (first phase)	25.854,47	2645,95	23208,52	77,34	59,19	295,8*
Matheuristic (second phase)	25865,17	2645,95	23219,22	77,41	59,26	111,9*

CONCLUSIONS

Bibliography

Aarts , E. & Lenstra , J., 1997. *Local Search in Combinatorial Optimization.* s.l.:John Wiley & Sons.

Anderson, E. W. & Sullivan, M., 1993. The antecedents and consequences of customer--satisfaction for firms.. *Marketing Science,* p. 125–143..

Barnes, L., 2009. Fast fashion in the retail store environment.. *Internationa Journal of Retail and Distribution Management,* pp. 760-772.

Boatwright, P. & Nunes, J., 2001. Reducing Assortment: An Attribute-Based Approach.. *Journal of Marketing*, p. 50–63.

Broniarczyk, S., Hoyer, L. & Hoyer, E., 1998. Consumers' Perceptions of the Assortment Offered in a Grocery Category: The Impact of Item Reduction.. *Journal of Marketing Research*, 2(35), p. 166–176.

Bruce, M., Daly, L. & Towers, N., 2007. Lean or agile - a solution for supply chain management in the textiles and clothing industry?. *International Journal of Operations and Production Management,* pp. 151-170.

Campo, K., Nisol, P. & Gijsbrechts, E., 2000. Towards Understanding Consumer Response to Stock-Outs.. *Journal of Retailing*, p. 219–242.

Chu, C., 1992. A branch-and-bound algorithm to minimize total flow time with anequal release dates.. *Naval Research Logistic.*

Coraggia, A., 2009. *Il supply chain management come strumento di marketing: il caso inditex-zara.* [Online]
Available at: http://tesi.eprints.luiss.it/5795/

Davis-Sramek , B., Stank, T. P. & Mentzer, J. T., 2007. Creating consumer durable retailer customer loyalty through order fulfillment service operations.. *Journal of Operations Management*, p. 781–797.

De Carlo, F., Borgia, O. & Tucci, M., 2013. Bucket brigades to increase productivity in a luxury assembly line.. *International Journal of Engineering and Business Management.*

De Felice, F., Gnoni, M. G. & Petrillo, A., 2012. A multi-criteria approach for sustainable mass customisation in the fashion supply chain.. *International Journal of Mass Customisation,* Issue 4.

Ek, M. & Karlsson Steijffert, A., 2015. Optimization of sales in fashion retail by warehouse integration in multichannels.

Giese, J. L., Johnson, J. L. & Wallace , D. W., 2004. Customer Retailer Loyalty in the Context of Multiple Channel Strategies. *Journal of Retailing*, Issue 80, pp. 249-63.

Iannone, R. et al., 2013. Merchandise and replenishment planning optimization for fashion retail.. *International Journal of Engineering Business Management,* Issue 5.

Iannone, R., Martino, G., Miranda, S. & Riemma, S., s.d. Modeling fashion retail supply chain through causal loop diagram. 2015 May, Volume 48, p. 1290–1295.

Lam, J. K. C. & Postle, R., 2006. Textile and apparel supply chain management in Hong Kong. *International Journal og Clothing Science and Technology,* Issue 18, pp. 265-277.

Lanzilotto, A., Martino, G., Gnoni, M. G. & Iannone, R., 2014. *Impact analysis of a cross-channel strategy in the fashion retail industry: a conceptual ramework..* Senigallia (AN) - Italy, s.n.

Lanzilotto, A., Martino, G., Gnoni, M. G. & Iannone, R., 2015. *Impact analysis of a cross-channel retailing system in the fashion industry by a simulation approach.* s.l., s.n.

Lin, M.-H.et al., 2013. A Review of Piecewise Linearization Methods. *Mathematical Problems in Engineering,* p. 8.

Martìnez-de Albèniz, V. & Boada Collado, P., 2014. Estimating and optimizing the impact of inventory on consumer choices in a fashion retail setting.. *IESE Business School.*

Martino, G., 2015. *Supply Chain Management in the Fashion Retail Industry: a multi-method approach fot the optimisation of performances,* s.l.: s.n.

Munson, I., 2014. *Economics Explained: Complements, Substitutes, and Elasticity of Demand.* [Online]
Available at: http://www.econogist.com/home/complements-and-substitutes
[Consultato il giorno 21 October 2018].

Radasanu, A., 2016. Inventory management, service level and safety stock. *Journal of Public Administration, Finance and Law*, pp. 145 - 153.

Sen, A., 2008. The US fashion industry: A supply chain review. *International Journal of Production economics*, 11 February, pp. 571-593.

Sloot, L. & Verhoef, P., 2005. The Impact of Brand Equity and the Hedonic Level of a Product on Consumer Stock Out Reactions. *Journal of Retailing*, p. 15–34.

Tadei, R., Della Croce, F. & Grosso, A., 2005. *Fondamenti di Ottimizzazione.* s.l.: Esculapio.

Thomassey, S. & Hapiette., M., 2007. Neural clustering and classification system for sales forecasting of new apparel items. *Applied Soft Computing,* Issue 7, pp. 1177-1187.

Vaagen, H. & Wallace, W., 2008. Product variety arising from hedging in the fashion supply chains.. *International Journal of Production Economics,* Issue 114, pp. 431-455.

Voss, S., Stutzle , T. & Maniezzo , V., 2009. Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. *Annals of Information Systems*, Volume 10, pp. 1-270.

Wang, X., Chan, H. K., Yee, R. & Diaz-Rainey, I., 2012. A two-stage fuzzy-ahp model for risk assessment of implementing green initiatives in the fashion supply chain.. *International Journal of Production Economics,* pp. 595-606.

Fig. 6 The clothing and textile supply chain

Fig. 7 Feasibility polytope and optimum point position with respect to the objective function direction.

Fig. 8 Local vs global optima in a non-convex function

Fig. 9 Poisson probability density function with different lambdas

Fig. 10 Initial inventory states with respect to the Poisson distribution and its parameters