Erasmus University Rotterdam, Erasmus School of Economics

Master Thesis
Econometrics and Management Science

Operations Research and Quantitative Logistics

Uncertainty Models in Operations Planning

Abstract

In this thesis, we consider the use of uncertainty models to create an optimal operations planning for retailers. With a focus on demand uncertainty, we show that by using scenario tree building, we are able to reliably improve the sales value by on average 5%, compared to operations planning based on single forecasts. Furthermore, we demonstrate that splitting the shipment of all items of a product, such that a part of all items arrive during the sales period, does not lead to a sales value reduction by using our stochastic method. Lastly, when making the exact arrival time of such a split shipment flexible, with a decision on that arrival stage during the sales period, we prove that we can maintain the sales value as if all items would have arrived before the sales period, by using the information on the sales up to the decision stage.

R.H. Blokland
409043

October 29, 2020

DISTRICON
advisory solutions professionals

Supervisor:
R. Teixeira MSc

ERASMUS UNIVERSITEIT ROTTERDAM

Supervisor:
Dr. K.S. Postek
Second assessor:
Dr. O. Kuryatnikova

Contents

1 Introduction

Every year, retailers face the challenge of making profits with margins on sales items becoming smaller, while competition rises. Therefore, it has become even more important to maximise the efficiency of the supply chain. This applies to fulfilling as much demand as possible, while making sure that costs related to the sale of a product are minimised. It is difficult to know exactly where to place your inventory, such that customers will not enter a store without finding the product they desire, while another store still has plenty inventory left of that same product. Especially the rise of online shopping has increased this difficulty for many retailers, up to the case where some had to go out of business.

For retailers, this problem often is taken one step further. Because a collection of clothing normally only lasts for one season, they have to deal with all the above generally in up to three months, after which all unsold inventory can be disposed of for a normally unprofitable salvage value. Because the margins normally are not very large, it is impossible to give every store plenty of inventory, whereas leaving stores without some items is not desirable either. The largest problem lies in the great uncertainty of demand. From historical sales, retailer can find a range of possibilities regarding when and how much an item would sell, but it is impossible to exactly estimate what will happen. Therefore, retailers need to find a balance between responding to the occurred demand at stores, while taking into account the amount of items to be distributed and thus also produced.

To stay profitable, many larger organisations have adopted the use of a process called Sales and Operations Planning (S&OP). This process focuses on integrating business strategy and operational planning, such that all relevant business function and supply chain elements are well aligned. Together with advanced inventory allocation methods, this process can be especially useful to improve the efficiency of the supply chain, in the sense that the management team is better able to create a strategic plan based on their current performance level.

1.1 Literature Review

This paper will focus on the uncertainty in the operations side of a SOP process. Because we want to give a clear picture of the entire process, we split this section into the following three parts. First, we will go into further detail regarding so far developed S&OP strategies to give the reader an idea of the current status. Next, since this paper will focus on applying uncertainty models, we will give a short introduction of stochastic optimisation and robust optimisation. Last, we provide an overview of applications of uncertainty models in situations from which we can learn regarding the implementation of the operations side of the S&OP mechanism.

1.1.1 Sales and operations planning

Tuomikangas and Kaipia (2014) give a clear overview of the existing literature regarding the S&OP process. They describe multiple coordination mechanisms in both academic and practitioner literature, to provide a framework regarding these mechanisms in the context of SOP. The authors explicitly focus on the coordination part of this process, since many companies find it difficult to realise expected benefits. Bower (2005) illustrates such an example, pointing out the fact that many companies do not know what the entire S&OP process entails. Also, he gives a shortlist of twelve pitfalls regarding the use of the process. He sees the S&OP process as a "dynamic collaborative planning and decision making process between functions" (Tuomikangas & Kaipia, 2014), which emphasizes the goal of aligning all business parts of a company. A second perspective on the S&OP process as described by Tuomikangas and Kaipia (2014) is a "method-oriented perspective on planning". This approach aims to maximise profits, while adhering to a defined set of constraints, where S&OP is useful as support for making fact-based decisions.

Another such research synthesis is that of Thomé, Scavarda, Fernandez, and Scavarda (2012). They additionally assemble quantitative evidence of the impact of S&OP on the performance of companies. The authors stress that S&OP can have many different goals, amongst which are reducing inventories and balancing supply and demand. They also note that only few papers have analysed the impact of an S&OP process on a firm's performance. In one of those (Feng, D'Amours, & Beauregard, 2008), it was showed that by using a fully integrated S&OP process with a mixed integer-based programming model, the financial results were better than in a partially integrated or decoupled planning case, where the sales and operations teams cooperate less before the sales period starts. Olhager and Selldin (2007) find the same conclusion. Both papers also state that S&OP is especially useful in situations of market uncertainty.

More importantly, Olhager and Selldin (2007) adds that firms with a major market uncertainty factor should choose approaches which are able to deal with this uncertainty. Current performance evaluations use mainly point forecasts, from which scenarios are created and used to create plans. While the inventory costs and stock levels already improve by using an S&OP process, a focus on anticipating on demand uncertainty in the supply plans could be even more beneficial.

For clarity in the remainder of the paper, we will not go into the implementation side of the S&OP process. In fact, we assume the retailer has its business parts well aligned and we focus on the uncertainty in inventory allocation models, which can only work well if the retailer executes their operations planning smoothly. This also points us to the most important contribution of this paper to existing literature, since we evaluate the usefulness of uncertainty models for an S&OP process in different settings. This paper distincts itself mainly in the setting of split shipments, which we will introduce in the problem description (Chapter 2).

1.1.2 Uncertainty models

Optimisation under uncertainty has been recognised as a relevant part already in the early stages of the development of optimisation. Sahinidis (2004) gives an overview of approaches to deal with problems with uncertainty. Nowadays, stochastic programming and robust optimisation can be seen as the two most widely spread methods. The author adds the concepts of fuzzy programming and stochastic dynamic programming to these two, although the latter can be seen as a part of stochastic programming.

In the book of Birge and Louveaux (2011), the authors provide readers with an in-depth look on how to model uncertainty explicitly. Also, they give insights regarding the implications of using such methods and the impact on expected results. We will go into more detail regarding a more detailed approach on the modelling side in Chapter 3.

While stochastic programming deals with the assumption of knowing how this uncertainty is formed and the implications it has on the uncertainty, robust optimisation uses a different perspective. Using a robust approach is mainly useful in cases where the researcher is unaware of the exact figures of uncertainty, but is able to find a suitable uncertainty set in which these uncertain parameters reside. Ben-Tal, El Ghaoui, and Nemirovski (2009) is a detailed book in this field, which can be seen as a good starting point for using this approach.

It should be noted that uncertainty models can become computationally very expensive, especially when considering models with multiple stages. Shapiro and Nemirovski (2005) even argue that "multistage problems are generally computationally intractable already when medium-accuracy solutions are sought". Even though they are not able to prove this statement, it gives one food for thought and it is also the reason why many stochastic models are solved approximately or using other algorithms.

1.1.3 Applications of uncertainty models

Uncertainty models have already been applied to a large set of types of problems. One of those, which we will investigate in this paper, is a situation where one needs to decide on the supply requirement under demand uncertainty. A typical paper, also addressing the problem in an S&OP process, is Sodhi and Tang (2011). They create a stochastic programming model which takes the demand uncertainty into account, and apply methods to make the model tractable. Notable is their use of risk metrics, where they build upon their earlier work in Sodhi and Tang (2009), which uses extensive literature on stochastic programming in the field of asset-liability management.

These risk metrics help to keep the model computationally within limits, since this can become rather difficult (Sen, 2001). Another method often used is generating scenario trees, as for example

in Kazemi Zanjani, Nourelfath, and Ait-Kadi (2010). In both Dupačová, Consigli, and Wallace (2000) and Høyland and Wallace (2001), the authors create the basis for such use, on which other work continues. Methods to, for example, aggregate these trees (Sodhi & Tang, 2011) or reduce the number of scenarios (Heitsch and Römisch (2003) with an example in Growe-Kuska, Heitsch, and Romisch (2003)) have since been developed.

As opposed to scenario tree building (SCT), one could also opt for the use of linear decision rules (LDR). Rocha and Kuhn (2012) give reasons why one could use such a method, rather than scenario trees, where such an example is the exponential growth of models using scenario trees. They even show that their LDR method is superior with regards to scalability and accuracy. Kuhn, Wiesemann, and Georghiou (2011) propose an efficient method for use of linear decision rules, which can be used in the case of a multistage stochastic program. By using both primal and dual linear decision rule approximations they are able to exploit the benefits of both methods.

Next to these options, other algorithms can also be developed, which can depend on the problem itself. For example, Monte Carlo sampling methods is seen as one of the few, if not only, reasonable way to estimate the expectation function (Shapiro, 2001) and therefore also has applications in the two-stage setting. Furthermore, Baucke, Downward, and Zakeri (2017) describe an algorithm which builds upon the earlier literature using Monte Carlo simulation to develop a deterministic algorithm to solve multistage stochastic programming problems. Also, some decomposition techniques are elaborated on in Birge and Louveaux (2011).

1.2 Research Questions and Thesis Structure

While we give a full description of the problem in Chapter 2, we already will go into the research questions we try to answer in this paper, with which we conclude in Chapter 6. Our main goal is to develop a method which take the uncertainty in inventory planning into account. Here, we focus on demand uncertainty and inventory uncertainty. The latter is made explicit by splitting the shipment of products into parts and making their arrival times flexible (Sections 2.3 and 2.4). This leads us to the following research questions:

1. Can we construct a method, which takes demand uncertainty into account, to optimise the supply planning?
2. How does this method perform in comparison with a classical forecasting model?
3. What is the impact on the sales if not all inventory at the distribution centre arrives before the start of the sales period?
4. What is the impact on the sales if we decide on a later moment during the sales period when all items should have arrived?

To answer these questions, we will first more explicitly state our problem in Chapter 2. Next, we focus on how we will answer the above questions in Chapters 3 and 4. Then, we analyse our methods in Chapter 5, where we first expand on specific settings, data and comparison material, before we continue with the computational experiments. Lastly, we will conclude by answering our research questions in Chapter 6, where we denote points up for discussion and possibilities for further research.

2 Problem Description

In this thesis, we will consider uncertainties in an inventory management setting. To be more specific, we focus on two types of unreliability. The first is the uncertainty of demand, applicable to both the size and location of the demand. The second comprises uncertainties in shipment timing. In this chapter, we will first go into detail regarding the process in an inventory management setting. In Sections 2.2 to 2.4, we go into more detail regarding the uncertainties in this setting and its complications.

2.1 Process

In a standard inventory management setting, retailers use periodical replenishments to update their inventories in store according to the development of sales. Also, depending on the type of items, the distribution centre will receive replenishments periodically. This is mainly applicable to never out of stock (NOOS) articles, such as laundry detergents, whereas seasonal products might receive their entire usable inventory in advance. In general, each products' inventory cycle can be put into the same schematic overview, displayed in Figure 1.

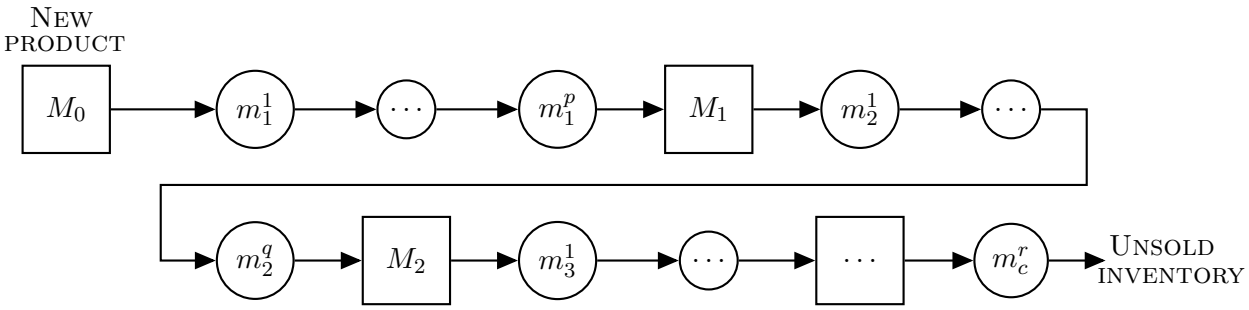


Figure 1: Schematic overview of the inventory management process for a typical retailer. Circles represent inventory replenishments, while blocks represent additional products arriving at the distribution centre.

The schematic overview in Figure 1 is characterised by the regular replenishments, where periodically new products enter the supply chain and can be used to replenish stores too. For products with no best before date, the dots in this scheme can be extended for a very long time, where at one point the product might be pulled out of sales. An additional factor in this scheme would be applicable for products with a best before date, where unsold products past their date should be taken out of inventories. As such, we could differentiate on more factors, which we disregard here for readability purposes.

An important aspect in this process is the behaviour of sales, where, for example, seasonal factors can play in important role in the replenishment sizes. Also, some products might sell very often in the beginning, with the sales numbers deteriorating over time. An example of the latter are short

life cycle products, on which we will focus our attention in this paper.

Short life cycle products are typically identified as retail products. Especially fashion retail products often have a short life cycle, which can be as short as two to three months for seasonal products. If the life cycle of an item is short, the retailer might receive all products before the sales period, or occasionally have a second shipment arrive in an early stage of the sales period. This makes the schematic view on the process considerably smaller and more specific. Figure 2 is an example of a product selling for three months, with a weekly inventory replenishment for each store. Furthermore, in the fifth week, a second shipment of the item arrives at the distribution centre.

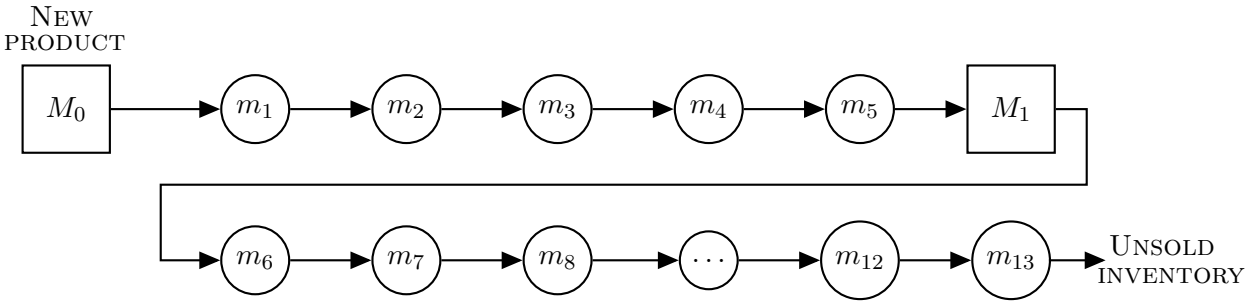


Figure 2: Schematic overview of the inventory management process for a product with a life cycle time of thirteen weeks, with weekly replenishments and an additional shipment of items arriving in the fifth week. Circles represent inventory replenishments, while blocks represent additional products arriving at the distribution centre.

2.2 Inventory Allocation

As we see in Figures 1 and 2, we make inventory allocation decisions periodically to update the inventory and react to the development of demand over different stores. Since the allocations in the second week depend on the allocations and sales in the first week, each decision is based on gathered information in the weeks before. This is especially relevant for short life cycle products. Not only will it become known which stores sell first and need replenishments sooner, also information on the to be realised sales level of a product becomes available after every stage.

Suppose we have an inventory allocation process with weekly replenishments to stores. Then, Figure 3 shows a more detailed overview of the problem faced in the first seven weeks. A truck delivers all items to the store before the first week. Then, every week, we determine how many items should go to each store.

Therefore, the main issue of the inventory allocation is the actual size of these replenishments. If a product has a hundred items available, a retailer might send ten items to the ten available stores, but it might be more profitable to divide this differently. Therefore, the retailer would like to wait with distributing all items, especially if he knows that and when it is possible to replenish again.

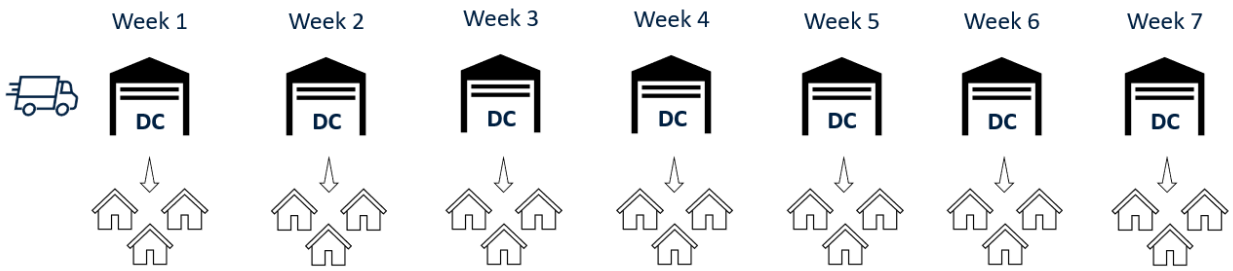


Figure 3: A typical inventory allocation problem, where one distribution centre replenishes all stores periodically.

On the other hand, it is important to not miss out on possible sales, since a number of items are likely to remain unsold at the end of the sales period. Also, selling early could result in selling more items for a non-discounted price.

To solve these problems, a well known option would be to use a classic inventory policy model. However, in many cases, a retailer can use historical sales information to make estimates of the sales progress of new products. The retailer can then opt to make use of uncertainty models. By considering what might happen to the sales in the future, knowing that the retailer is able to replenish, it could be possible to improve the inventory allocation strategy. Instead of reacting on the sales progress of a product, the retailer would be anticipating on the sales progress.

Then, in each of the stages where the retailer wants to decide on the inventory allocation, as in Figure 3, he faces a multistage uncertainty problem (Birge & Louveaux, 2011). In this paper, we adopt this problem and investigate how to optimise this decision with the uncertainty of the future taken into account.

For this inventory allocation problem, we also include multiple relevant business factors. As a service for customers, many stores include the right to return your items to the store, after which the retailer can sell this item again. This causes the inventory to be both dependent on the returns and replenishments. Furthermore, in settings such as those for fashion retailers, typically every product should have at least one item present. As an example, a t-shirt in a store would need each size present, in case customers want a certain size. More details on how to handle these more practical issues will be provided in Section 3.3.

2.3 Split Shipment

Often, in the problem as described in Section 2.2, all inventory arrives at the distribution centre before a sales period starts. Then, retailers redistribute their items over stores. However, retailers could receive such large amounts of inventory that it could flood the distribution centre. Therefore,

retailers might split their shipments such that the second half arrives at a later time during the sales period.

In practice, the problem is similar as to the schematic overview in Figure 3. However, the main difference is that the truck in the first week does not ship all items to the distribution centre. In fact, we could for example see a second truck arrive in week five. In Figure 4, we display the adaptation to the problem is this case.

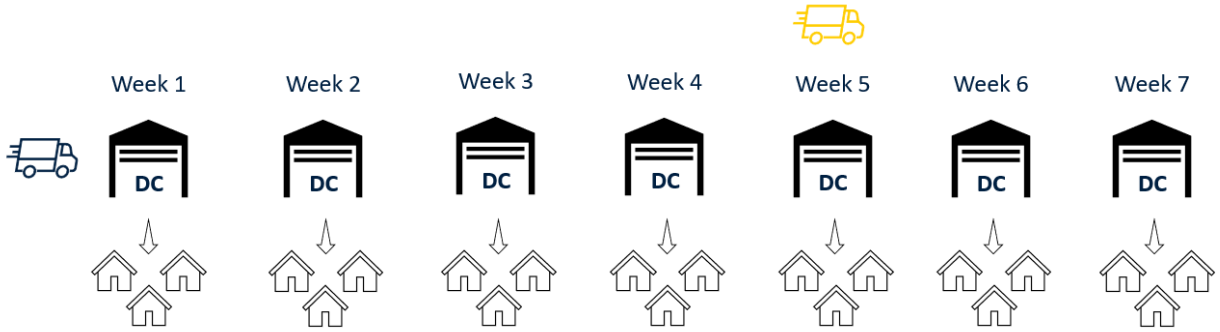


Figure 4: An inventory allocation problem with a split shipment.

The main point of interest is whether limiting the number of items which can be distributed from the start, will impact the results. Having less items available could result in being more cautious, which could impact the sales negatively. If it does not, this opens the door for a more rigorous approach, which we will elaborate on Section 2.4.

2.4 Flexible Shipment Arrival Time

In the previous section (Section 2.3), we introduced the possibility of splitting shipments due to flooding of the distribution centre. We could broaden this perspective by considering the problem by allowing that a second shipment comes in at a to be determined point in time. This raises our last research question, whether we can make split shipments flexible while maintaining a similar sales level.

We can think of multiple reasons for doing this. First, holding off from having at that time superfluous inventory can save money in the form of time and space. We not only reduce pressure at the distribution centre, but a more constant and lower handling volume can also be beneficial with regards to distribution centre size and workforce.

Second, a shipment method with a longer transportation time is often considerably cheaper. For example flying products by plane or shipping on a vessel can make a major difference. Therefore, we would like to choose the optimal transportation method for each product. We could accelerate

the planned arrival if a product sells well, or delay the planned arrival if it is unpopular.

Suppose we consider a similar inventory allocation problem as in Figures 3 and 4. However, we now allow the second shipment arrival by truck to be flexible. Since retailers often have long lead times, products are already on their way such that we can not let the truck arrive earlier or later. Instead, we could choose to fly products into the distribution centre by plane from a given point, or let our products sit on an inland vessel, causing a longer transportation time, which can also be translated into outsourcing inventory space.

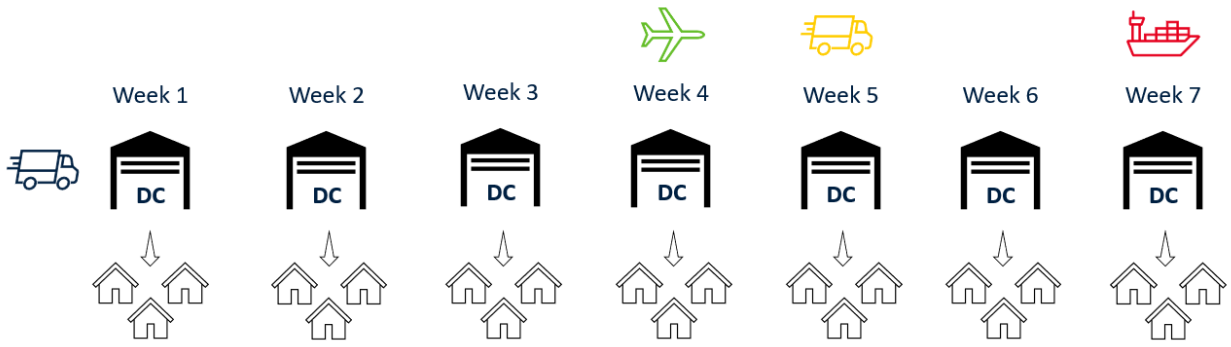


Figure 5: An inventory allocation problem with a flexible shipment arrival time.

We display this situation in Figure 5. Instead of the definitive arrival of the truck in week five, we now allow for the possibility that it either arrives by plane in week four, or on the vessel in week seven.

We do emphasize that at a certain moment the shipment decision does need to be made. This idea is only useful if we know something about the development of the sales level of a product. Therefore, we could for example set a hard deadline for the decision on when the shipment should arrive after the second week. For the remainder of this paper, we will refer to this point as the shipment decision. Also, for the purpose of describing the problem, we limit the arrival options. However, in reality, the retailer is probably able to choose out of more options, ending up with choosing the moment which fits the product best if this is practically doable.

3 Methodology

In Chapter 2, we introduced our problem and the uncertainty comprised in an optimal inventory management plan. We can apply multiple methods to deal with these uncertain factors, as described in Section 1.1. We first illustrate the applicability of these methods to our problem in Section 3.1, where we also specify the reasoning for our method of choice. In Section 3.2, we then introduce the modelling technique used in stochastic optimisation. Lastly, we construct the mathematical models which form the basis of our solution method in Sections 3.3 to 3.5.

3.1 Uncertainty Models

We introduced multiple methods to handle problems with uncertainties in Section 1.1. These can mainly be divided into two *frameworks*, robust optimisation and stochastic optimisation. In Section 3.1.1, we denote their differences and explain our reasoning behind continuing with one of the two. Then, in Section 3.1.2, we explain how we can and will choose to concretise the uncertainty to be able to solve our problem.

3.1.1 Uncertainty optimisation frameworks

We must note that robust optimisation can be seen as a part of stochastic optimisation, but typical characteristics of problems can be distinguished. While both types of problems are similar, they can be distinguished by two key factors. First, in robust optimisation we generally assume that we do not have specific information on the distribution of the random parameters, while this is known in the form of scenarios or probability distributions in the case of stochastic optimisation. Second, the objective functions of both types of modelling will normally be different. In stochastic optimisation, we want to for example maximise the expected profits, but in robust optimisation we seek to find solutions which would for example be optimal in a worst-case situation.

Our problem can be translated into both frameworks for uncertainty models. Demand is unknown, and we want to make sure we can satisfy the customers' needs for as long into the sales period as possible. On the other hand, we do have some information regarding what can be expected regarding sales, and the eventual goal for a retailer is to maximise profits in the long run. Especially since we have information on potential demand and are able to make this information explicit in the form of different possibilities, a stochastic optimisation framework seems more fitting. This also allows us to change our non-deterministic problem into a deterministic one, by creating scenarios on demand occurrences. Furthermore, retailers want to maximise their profits in any case, thus maximising over the expected profit would also be more fitting.

Even though the method we choose to tackle our problem will mainly correspond to those used in stochastic optimisation, it would be interesting to approach our problem using a robust optimisation

framework as a follow-up research. This will likely give new insights in how our methods would perform in worst-case scenarios, or how much a retailer has to sacrifice in order to reach different goals such as customer satisfaction. However, this is outside the scope of this thesis and could be the basis for further research.

3.1.2 Concretising the uncertainty

In our literature review, we describe multiple methods on how to concretise the uncertainty in an attempt to solve the actual problem (Section 1.1.3). Already for a longer time, methods with scenario trees have been widely used, mostly due to the ease with which it can be explained and implemented. Furthermore, one can control the size of the problem by controlling the size of the scenario tree, which is the centre of corresponding mathematical models. Therefore, we continue with using scenario trees as well.

Because of the clean way of creating a tree, we can easily construct our problem into scenarios representing the possible outcomes of the known uncertainty. While it is not possible to capture all uncertainty in scenarios, we can be very specific on the included information. In fact, we can view this method as some sort of advanced forecasting, where we can modify the degree of uncertainty of these forecasts. If we compare this with a classical forecasting method often used in the operations planning, this stochastic method is an upgrade to that classical idea. We go into more details regarding this idea in Section 5.2, where we introduce the benchmark we will use in our simulations, but we already state something about this now.

In principle, we evolve this classical idea in two steps. We illustrate this process in Figure 6. In the first step, we go from using one forecast for each store to creating cases, which would all be solved separately. This means we would for example have three cases, ranging from *strong* sales to *weak* sales, and based on these one can optimise the supply plan.

Ideally, we want to create a supply plan based on all cases, evaluated all together. We can mathematically take into account the corresponding probabilities, costs and benefits of each scenario. This is exactly what the scenario tree building method entails, which is illustrated in the second step of Figure 6. Here, we combine the cases into a *scenario tree*, which consists of scenarios and probabilities of going from one scenario to another. Note, a scenario constitutes demand in one or multiple stores in one stage t. Then, a *scenario path* consists of a sequence of scenarios.

As we can easily see, the scenario tree does have a lot more possibilities with regards to what might happen with sales, which could become difficult with regards to computation time. However, it is more detailed and theoretically should give a better result than single forecasts and also the more classical scenario analysis.

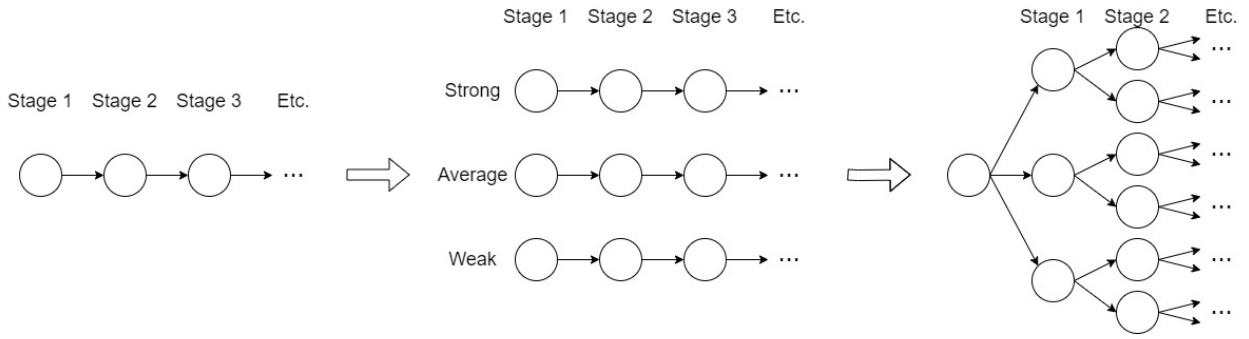


Figure 6: Evolving from a classical forecasting method to applying scenario trees.

3.2 Multistage Stochastic Optimisation

Before we introduce any modelling representing the inventory allocation problem, we briefly discuss how modelling in a multistage setting actually looks like. The use of notation in stochastic programs differs widely amongst the existing literature. Therefore, we start in Section 3.2.1 by indicating in what manner we will use notation throughout this paper. In Section 3.2.2, we then generalise the introduced type of problem to a more general setting. Lastly, we shortly go into depth on how we can solve such problems in Section 3.2.3.

3.2.1 Two-stage setting

As a basis, we will use the notations as in Birge and Louveaux (2011) throughout this paper. While we use a multistage stochastic program, we introduce the notation in a simpler form. Birge and Louveaux (2011) define a two-stage stochastic linear program with fixed recourse as the following:

$$\begin{aligned} \min \quad & z = c^T x + \mathbb{E}_\xi \left[\min q(\omega)^T y(\omega) \right] \\ \text{subject to} \quad & Ax = b, \\ & T(\omega)x + Wy(\omega) = h(\omega), \\ & x \geq 0, y(\omega) \geq 0. \end{aligned} \tag{1}$$

In this model, the program is split into two stages. The vector x will comprise the decisions in the first stage. Then, in the second stage, the uncertain factor ω in the problem becomes known (noted as events $\omega \in \Omega$), which implies that the scenario-dependent data of $q(\omega)$, $h(\omega)$ and $T(\omega)$ becomes known too. The decisions in the second stage will then be represented by $y(\omega)$.

3.2.2 Multistage setting

To extend the two-stage setting, we use a similar notation as in the two-stage model, as provided by Birge and Louveaux (2011). Note that transposes are excluded if they are clear from context. Then,

the multistage stochastic linear program with fixed recourse can generally be denoted as follows:

$$\begin{aligned}
\min \quad & z = c^1x^1 + \mathbb{E}_{\xi^2}\left[\min c(\omega)^2x^2(\omega^2) + \ldots + \mathbb{E}_{\xi^H}\left[\min c(\omega)^H x^H(\omega^H)\right]\ldots\right] \\
\text{subject to} \quad & W^1x^1 = h^1, \\
& T^1(\omega^2)x^1 + W^2x^2(\omega^2) = h^2(\omega), \\
& \qquad\qquad \vdots \\
& T^{H-1}(\omega^{H-1})x^{H-1} + W^Hx^H(\omega^H) = h^H(\omega), \\
& x^1 \geq 0, \\
& x^t(\omega^t) \geq 0, \qquad\qquad \forall t \in \{2,\ldots,H\}.
\end{aligned} \tag{2}$$

The objective function consists of expectations for each stage. Since each of those stages follow on earlier, yet to occur, stages, each stage responds to random scenarios through the variables $x^t(\omega^t)$. The variables are then linked via constraints stretching over two following stages.

3.2.3 Solution methods

While we can always solve these models exactly, this might not be the best option. Especially in large settings, these problems become very difficult to solve (Shapiro & Nemirovski, 2005). Therefore, Birge and Louveaux (2011) describe a simplification to this multistage setting into a deterministic two-stage equivalent of this problem, which could help us solve such larger problems. A clear and simple way of finding the solution in each stage, is by using backward recursion. Then, the solution for all $t \in \{2,\ldots,H\}$ would be

$$\begin{aligned}
Q^t(x^{t-1}, \xi^t(\omega)) = \min \quad & c^t(\omega)x^t(\omega) + \mathscr{L}^{t+1}(x^t) \\
\text{subject to} \quad & W^tx^t(\omega) = h^t(\omega) - T^{t-1}(\omega)x^{t-1}, \\
& x^t(\omega) \geq 0,
\end{aligned} \tag{3}$$

where $\mathscr{L}^{H+1}(x^H) = 0$ such that the recursion will terminate, since H would be the last stage. In other stages, $\mathscr{L}^{t+1}(x^t) = \mathbb{E}_{\xi^{t+1}}\left[Q^{t+1}\left(x^t, \xi^{t+1}(\omega)\right)\right]$. Then, to find the solution we actually seek, we should use the deterministic form of the first stage, which is

$$\begin{aligned}
\min \quad & z = c^1x^1 + \mathscr{L}^2(x^1) \\
\text{subject to} \quad & W^1x^1 = h^1, \\
& x^1 \geq 0.
\end{aligned} \tag{4}$$

Multiple algorithms (Section 1.1.3) to solve our problem are based on such an approach, such as the decomposition techniques elaborated on in Birge and Louveaux (2011). In reality, using such an approach is not very applicable. Above, we need to assume that the only interaction between stages is through x^t, dependent on one stage. However, in our problem, variables on both supply and returns are dependent on two stages (Section 3.3). This makes segmenting our problem into

bits similar to the method described above difficult, if not impossible. Furthermore, if the scenario tree is small (we will go into the computation time in Section 5.5), modern solvers such as CPLEX are able to solve problems of our size in an acceptable time frame.

3.3 Modelling a Multistage Inventory Allocation Problem

Inventory allocation problems are in essence rather simple. Especially in retail, the goal is to sell as many items as soon as possible, specifically at locations with a high sales price. If one would know the demand, the optimal solution would be sending the exact size of demand to each location. While we can not be 100% accurate, we can create forecasts and build expectations on this demand. Then, we want to optimise the distribution of our items given these expectations.

Since we often do not have a sufficient amount of items to fill every location with the maximum expected items to sell, we need to make choices on this allocation. These choices are not only based on the probabilities of selling in the first period after locations get new supplies, but demand expectations beyond that period are just as important. After all, we do not want to be stuck with many items at a location once sales disappoint.

This is where the multistage setting plays a big role. The size of this demand can be represented in scenarios, where each scenario will occur with an estimated probability. Now, we can allocate our items based on an objective representing the expected profit made by sending a certain amount of items to locations.

Before we introduce the model, we shortly emphasize the difference between using a setting with lost sales and a setting with backordering. When using a lost sales model, we assume that all unfulfilled demand is lost. This in contrast to applying backordering, where each unmet item will be backordered, such that no unmet demand will occur if there still is inventory in the distribution centre. The latter occurs in, for example, fashion stores, where unavailable sizes can be ordered immediately in the store. Table 1 shows an example of what the resulting variables would look like in the different cases.

This section is further split into three parts. First, we denote relevant parameters and variables in Section 3.3.1. Then, in Section 3.3.2, we introduce the model of the lost sales case. We denote the modifications to this model to get the appropriate model in the backordering case in Section 3.3.3. Furthermore, a list of all sets, parameters and variables is also included in Appendix A.

3.3.1 Notation

To explicitly model different moments of supply and to estimate demand more fragmented, we distinguish stages, with stage $t \in \mathcal{T} = \{0, \dots, T\}$. Here, $t = 0$ marks the stage *before* the selling

Table 1: Example of the difference between lost sales and backordering.

	Lost Sales		Backordering	
	DC	Store	DC	Store
Inventory before	2	5	2	5
Demand	-	8	-	8
Sales	-	5	-	5
Backordered	-	-	-	2
Unmet	-	3	-	1
Inventory after	2	-	-	-

period starts and thus no demand has occurred yet, corresponding to the start phase of a collection. On the other hand, $t = T$ marks the stage where the current collection will be replaced, and all items will be removed and sold at a fixed (and lower) salvage price.

Also, demand will occur at different locations. We can differentiate between a physical store and a distribution centre. All products will first pass through the distribution centre before being sent to a store. We will denote location $n \in \mathcal{N} = \mathcal{N}_{\mathcal{S}} \cup \mathcal{N}_{\mathcal{D}}$. Suppose n is a physical store, then $n \in \mathcal{N}_{\mathcal{S}}$. Similarly, when n is a distribution centre, $n \in \mathcal{N}_{\mathcal{D}}$. In our cases, we will use only distribution centre. Therefore, we will explicitly use the notation DC instead of n, where the DC is intended.

In each stage, the demand sizes are represented with a scenario. Each scenario $\omega_t \in \Omega$ can be different for each stage, location and stock keeping unit (SKU), but for clarity we will only explicitly state the relevant stage. Furthermore, this set Ω can be further split into small sets, with one example being $\Omega^S_{w_t}$. This set consists of all successor scenarios of scenario ω_t, being scenarios in stage $t + 1$. We will leave specific sets for a SKU out of the formulation, since this makes the formulation unnecessarily heavy regarding subscripts. Also, for computational reasons, we solve the model separately for each SKU. Results on this computation time are given in Section (5.5.4).

Furthermore, we will not consider certain parameters. Since the costs for ordering all products have already been made, production costs will not be included. Also, shipping costs will not be included in the model, since many retailers use contracts allowing them to pay for shipping beforehand, causing the size of a shipment to be irrelevant. Such costs could have a significant impact on the choices of allocating our inventory, such as not shipping to a store every day. However, taking these costs into account as well is outside the scope of this thesis.

The problem type itself does require taking inventory costs into account. If it does not cost anything to keep items in a store, we might end up sending all of our inventory to stores, while we want to wait on sending items to stores such that we can analyse the behaviour of sales. Therefore,

we will include a measure for the holding costs, $h_{t,n}$ in our model. This will make it possible to balance the number of items in stores and the distribution centre. Furthermore, it can serve as a natural capacity constraint on the stores if the holding costs in these physical stores are set higher than in the distribution centre.

In these stores, fashion retailers also deal with a minimum capacity constraint, which is called store presentation. Especially in the first weeks of sales, having at least one product in stock is critical, since customers should be able to both see and put on their desired item. We will denote this minimum capacity of an item with $m_{t,n}$. Another relevant parameter is the initial inventory size $a_{0,n}$, which equals the total order quantity for $n = DC$ and 0 for $n = 0$.

Since we work with scenarios for demand, we also deal with two parameters dependent on the corresponding scenario. These are the demand, denoted by $d_{t,n}(\omega_t)$ and the probability of a scenario occurring, $p(\omega_t)$. Lastly, we represent the sales price in stage t and at location n by $\pi_{t,n}$.

To model the flow of inventory, we need three main variables. The first represents the inventory size at *the end* of a stage t at a location under a certain scenario, $I_{n,t}(\omega_t)$. Also, we need to represent the number of items sold during t given a scenario, which we denote by $z_{t,n}(\omega_t)$. To combine the first two, we need a variable denoting the supply of items to a store. Therefore, $y_{t,n}(\omega_{t-1})$ denotes the number of items supplied to location n at *the beginning* of period t.

We have stated variables dedicated to the flow of inventory, but since we are dealing with the possibility that we can not fulfill all demand immediately, we need to add a variable taking this into account. We denote this unmet demand by $u_{t,n}(\omega_t)$.

Lastly, in reality, some items will be returned. Therefore, to incorporate returns, we also include a variable $r_{t,n}(\omega_{t-1})$, stating the number of items to be returned to the store. Since the variable is dependent on the actual sales, we bound this variable in a constraint by using an expected return rate q_n, which differs on whether location n is an online or offline store.

3.3.2 Lost sales model

In the case of a lost sales model, as explained in the beginning of this section, it is assumed that a customer will leave if the desired product is not in inventory. The demand will therefore be lost, such that we can define the model as follows:

$$\max \quad \sum_{\omega_t \in \Omega} p(\omega_t) \left[\sum_{n \in \mathcal{N}} \Big(\pi_{t,n} \big[z_{t,n}(\omega_t) - r_{t,n}(\omega_t) \big] - h_{t,n} I_{t,n}(\omega_t) \Big) \right], \tag{5}$$

$$\text{s.t.} \quad I_{0,n}(\omega_0) = a_{0,n}, \qquad \forall n \in \mathcal{N}, \tag{6}$$

$$y_{0,n}(\omega_0) = 0, \qquad \forall n \in \mathcal{N}, \tag{7}$$

$$I_{t,n}(\omega_t) + y_{t+1,n}(\omega_t) + r_{t+1,n}(\omega_t) = z_{t+1,n}(\omega_{t+1}) + I_{t+1,n}(\omega_{t+1}), \qquad \forall t \in \mathcal{T} \setminus \{T\}, \forall n \in \mathcal{N}_{\mathcal{S}}, \tag{8}$$

$$I_{t,DC}(\omega_t) + y_{t+1,DC}(\omega_t) + r_{t+1,DC}(\omega_t) - \sum_{n \in \mathcal{N}_{\mathcal{S}}} y_{t+1,n}(\omega_t) = z_{t+1,DC}(\omega_{t+1}) + I_{t+1,DC}(\omega_{t+1}), \qquad \forall t \in \mathcal{T} \setminus \{T\}, \tag{9}$$

$$\sum_{n \in \mathcal{N}_{\mathcal{S}}} y_{t+1,n}(\omega_t) \leq I_{t,DC}(\omega_t), \qquad \forall t \in \mathcal{T} \setminus \{T-1, T\}, \tag{10}$$

$$z_{t,n}(\omega_t) + u_{t,n}(\omega_t) = d_{t,n}(\omega_t), \qquad \forall t \in \mathcal{T} \setminus \{T\}, \forall n \in \mathcal{N}, \tag{11}$$

$$q_n z_{t,n}(\omega_t) - 0.5 \leq r_{t,n}(\omega_t) < q_n z_{t,n}(\omega_t) + 0.5, \qquad \forall t \in \mathcal{T} \setminus \{T\}, \forall n \in \mathcal{N}, \tag{12}$$

$$I_{t,n}(\omega_t) + y_{t+1,n}(\omega_t) \geq m_{t+1,n}, \qquad \forall t \in \mathcal{T} \setminus \{T\}, \forall n \in \mathcal{N}, \tag{13}$$

$$I_{T,n}(\omega_T) = y_{T,n}(\omega_T) = u_{T,n}(\omega_T) = r_{T,n}(\omega_T) = 0, \qquad \forall n \in \mathcal{N}, \tag{14}$$

$$I_{t,n}(\omega_t), r_{t,n}(\omega_t), u_{t,n}(\omega_t), y_{t,n}(\omega_{t-1}), z_{t,n}(\omega_t) \in \mathbb{N}, \qquad \forall t \in \mathcal{T}, \forall n \in \mathcal{N}. \tag{15}$$

In the objective function, Equation (5), we maximise the expected profit of all scenarios, corrected for the holding costs of inventory. Then, in Constraints (6) and (7), we set the initial inventory of and supply to every location n on the corresponding logical parameters.

In Constraints (8) and (9), we model the relationship between the inventory of two successive scenarios in successive stages, characterised by supplies, sales and returns. Note that the inventory level in the distribution centre is also influenced by all supplies sent to stores from that distribution centre. In Constraints (10) we therefore make sure that the total supply to stores does not surpass the inventory level of the distribution centre at that stage, as that will result in an infeasible supply plan. We purposely do not include returns of the distribution centre yet, since we would then need to assume all returns would be back before the end of the week, which is rather unrealistic. We do assume that returns from sales in week t can be sold in week $t+1$ again.

Constraints (11) link the sales and demand. Those sales can then be turned into returns via Constraints (12), where depending on the type of store the expected return rate determines the integer number of returns, as we give it an interval range where at least one integer is included.

In Constraints (13), we take care of the store presentation, stating a minimum number of items to be present at the beginning of a stage. In more extreme cases, when for a lengthy period of stages a minimum number of items should be present at each store, infeasibility could occur because of

a lack of items. Then, we can relax this constraint by adding a penalty in the objective function. The size of this penalty depends on the willingness to accept not adhering to this constraint.

Lastly, Constraints (14) ensures that no inventory will be left over at the end of the sales period, to incorporate any benefits or costs for getting rid of items. Also, all denoted variables are integer numbers (Constraints (15)).

3.3.3 Backordering model

Now, rather than losing unfulfilled demand, we assume that each occurred demand will be backordered. While this in theory does not have to cost the retailer anything due to shipping agreements, we do not want our model to choose to have everything backordered. Therefore, we will include a backordering cost in our model. Furthermore, we need to adjust the constraint controlling the inventory size at the distribution centre. Since we backorder items, a part of the size of $u_{t,n}(\omega_t)$ can be fulfilled from the distribution centre, with costs b_n. We should however incorporate the possibility of the distribution centre not having enough items in stock. Thus, we will add a variable $v_{t,n}(\omega_t)$, representing the number of items backordered.

In this regard, we need to add a term to the objective function, such that in the case of backlogging the objective function becomes

$$\max \quad \sum_{\omega_t \in \Omega} p(\omega_t) \left[\sum_{n \in \mathcal{N}} \Big(\pi_{t,n} \big[z_{t,n}(\omega_t) + v_{t,n}(\omega_t) - r_{t,n}(\omega_t) \big] - b_n v_{t,n}(\omega_t) - h_{t,n} I_{t,n}(\omega_t) \Big) \right]. \tag{16}$$

Furthermore, we need to adjust the constraint controlling the inventory size at the distribution centre, Constraints (9), by adding the backordered number of items. Also, Constraints (11) and (12) will need to incorporate this extra variable. These three constraints are adjusted as follows, in their respective order,

$$\begin{aligned} I_{t,DC}(\omega_t) + y_{t+1,DC}(\omega_t) + r_{t+1,DC}(\omega_t) - \sum_{n \in \mathcal{N}_S} y_{t+1,n}(\omega_t) = \\ z_{t+1,DC}(\omega_{t+1}) + \sum_{n \in \mathcal{N}_S} v_{t+1,n}(\omega_{t_1}) + I_{t+1,DC}(\omega_{t+1}), \end{aligned} \qquad \forall t \in \mathcal{T} \setminus \{T\}, \tag{17}$$

$$z_{t,n}(\omega_t) + v_{t,n}(\omega_t) + u_{t,n}(\omega_t) = d_{t,n}(\omega_t), \qquad \forall t \in \mathcal{T} \setminus \{T\}, \forall n \in \mathcal{N}, \tag{18}$$

$$\begin{aligned} q_n \big[z_{t,n}(\omega_t) + v_{t,n}(\omega_t) \big] - 0.5 \leq r_{t,n}(\omega_t) < \\ q_n \big[z_{t,n}(\omega_t) + v_{t,n}(\omega_t) \big] + 0.5, \end{aligned} \qquad \forall t \in \mathcal{T} \setminus \{T\}, \forall n \in \mathcal{N}. \tag{19}$$

In addition, $v_{T,n}(\omega_T) = 0$ and $v_{t,n}(\omega_t) \in \mathbb{N}$. Other constraints should remain in the model as presented from Equations (5) to (15).

3.4 Split Shipment

The models in Section 3.3 do not anticipate on the possibility of split shipments, as described as a part of the problem in Section 2.3. We can adjust for such shipments to the model by giving the distribution centre more inventory when such shipments will arrive.

In Constraints (6) of the model in Equations $(5)-(15)$, we set an initial inventory at the distribution centre. For this purpose, we use a parameter $a_{0,n}$. To include additional shipments which come in at a later phase, we will generalise this parameter to $a_{t,n}$ and let Constraints (6) be valid for all $t \in \mathcal{T}$ and $n \in \mathcal{N}$. It is therefore necessary to predefine this parameter for all indices and it is only applicable in case the arrival moment is certain and can not be changed.

3.5 Flexible Shipment Arrival Time

In Section 2.4, we introduced a flexible split shipment, where the retailer can decide at a given moment when they want that later shipment(s) to arrive at the distribution centre. To apply this, we start with the model of Equations $(5)-(15)$ and expand it with constraints allowing for a flexible shipment arriving at the distribution centre in a later stage.

We first introduce some notation. We denote the set of transportation options by an index $s \in \mathcal{S}$. Each transportation option will have its own lead time L^s. These methods also have an associated cost, denoted by c^s. We add these costs to the objective function, with a distinction per scenario.

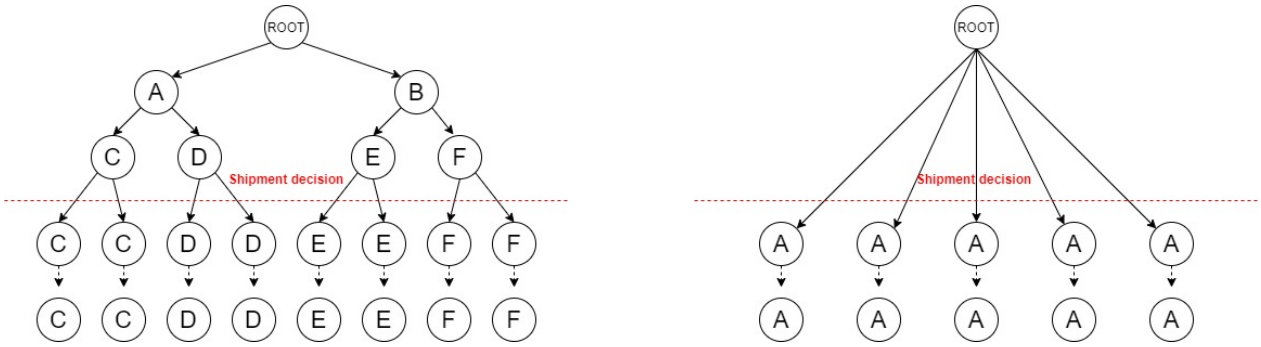
We base our constraints around the actual decision moments, which is why we introduce a new set $\mathcal{D}$. Each $d \in \mathcal{D}$ represents one later arriving shipment on which we need to make a decision. We add two parameters corresponding to each decision d. A_d corresponds to the size of the respective shipment. k_d represents the decision moment for this shipment, which is *after* stage k_d and *before* stage $k_d + 1$, thus we can include all information up to and including stage k_d for such a choice. We also add a binary variable x_d^s, denoting whether we choose transportation method s for the shipment of decision d.

We first add two constraints related to making sure the model can handle the presence of such a flexible shipment. The first, Constraints (20), make sure that we only choose one transportation method. Second, Constraints (21) make sure all items of a shipment arriving at stage t, are actually seen as arriving at the distribution centre in t.

$$\sum_{s \in \mathcal{S}} x_d^s(\omega_{t-1}) = 1, \qquad \forall d \in \mathcal{D}, \tag{20}$$

$$y_{t,DC}(\omega_{t-1}) = \sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}: L^s = t} x_d^s(\omega_{t-1}) A_d, \qquad \forall t \in \mathcal{T} \setminus \{0, T\}. \tag{21}$$

The third and last constraint regards the relationship between the choice on a flexible shipment and the scenario. As described in the problem description (Section 2.4), we make a decision on the arrival stage at stage k_d, after which this decision is final and thus cannot be changed. Therefore, all scenarios from stage $k_d + 1$ onwards should have the same choice on flexible shipment as their successor, since the available information at the decision moment is equal for all those scenarios.



(a) Well before shipment decision

(b) Stage of shipment decision

Figure 7: Example of scenario tree on non-anticipativity constraints flexible shipment

The above also means that before stage $k_d + 1$, the shipment decision can differ per scenario. These logical rules can be best expressed in Figure 7, where we distinguish between two moments. The first is well before a shipment decision and the other at the stage when the shipment decision should be taken. We can express these rules as non-anticipativity constraints, linking possible obligations to their predecessor, as in Constraints (22). Here, the model sets the shipment decision for flexible shipment d of a scenario (ψ_t) equal to its predecessor ω_{t-1} if $t \geq k_d + 1$.

$$x_d^s(\psi_t) = x_d^s(\omega_{t-1}), \qquad \forall t \in \{k_d + 1, \ldots, T\}, \forall d \in \mathcal{D}, \forall \psi_t \in \Omega^S_{\omega_{t-1}}. \tag{22}$$

These constraints hold for each decision that still needs to be taken. If the decision stage has already passed, x_d^s should be fixed for stages it is still applicable to, such that it remains incorporated in the model. Important to note is that when the shipment decision should be taken, we encounter the situation as in Figure 7b. Now, all scenarios are in a stage following the shipment decision. Therefore, they should be all equal and we effectively deal with one shipment decision.

4 Process Implementation

In this chapter, we go into more detail on how to evaluate the performance of our methods. In essence, we can not solve the model at the start of the sales period and then follow the closest path in the scenario tree. This can for example cause infeasibilities with regards to inventories in stores, or the actual behaviour is not represented in the scenario tree. To replicate a sales period, we need to simulate it by solving our model periodically and extracting the optimal here-and-now decisions on each solve. In Section 4.1, we go into detail regarding this process and also expand on relevant steps in this process, which we discuss in Sections 4.2 to 4.4.

4.1 Process

We evaluate our methods by using a process which can be mainly identified as a rolling window simulation, on which we expand in Section 4.1.1. Next, we give more details regarding the specific parts of this simulation in Section 4.1.2.

4.1.1 Rolling window simulation

In a sales period, we can differentiate on time and replenishments by using stages, as described in Section 2.1. Since we are only interested in the here-and-now decisions regarding supplies to stores, we will implement a rolling window simulation. In Figure 8, we give a schematic overview of what such a simulation will entail. For clarity, we define an iteration as one sequence of steps to find the here-and-now-decisions regarding supply or a shipment decision. We will describe those steps in Section 4.1.2.

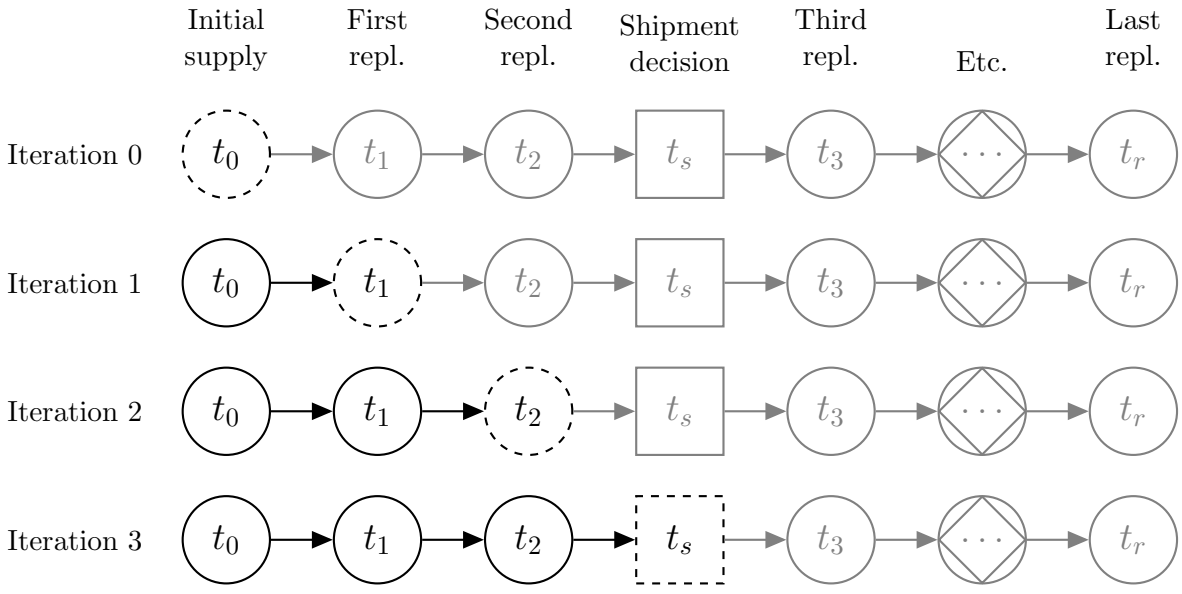


Figure 8: Iterative process of a rolling window simulation in an operations planning setting.

Essentially, we are simulating a regular sales and operations planning network, to optimise the decision in every iteration. During this simulation, we not only make decisions on the inventory, but also on the flexible shipment if applicable. In iteration 0, we find an optimal operations plan for the entire sales period, given the uncertainty on demand. However, we are only interested in the initial supply, which we implement. Then, we do the same in iteration 1, of which we only implement the first replenishment. We continue until we simulated the entire sales period and collect the summary statistics. Important to note is that the supply sizes can differ dependent on the created scenario tree in each iteration. To investigate the robustness of our methodology, we need to replicate this simulation a number of times on the same demand realisations.

4.1.2 Steps during an iteration

In each iteration, we distinguish between three main steps to take before we move on to the next iteration. We denote these steps in Algorithm 1, where we present the pseudo-code of an entire simulation run, corresponding to an extension of the schematic overview as in Figure 8.

Algorithm 1: Steps to take in the rolling window simulation

Input: Historical data, characteristics of sales product, scenario tree branching

Output: Realised profits and decisions on distribution of items

Pre-determine the demand in all stores in each stage

for *iteration* $f \in \mathcal{F}$ **do**

- **Step 1:** Compute a scenario tree ▷ See Section 4.2
  - *Step 1a:* Retrieve relevant historical sales data
  - *Step 1b:* Create scenarios corresponding to possible demand patterns
  - *Step 1c:* Build a tree by computing successive scenarios for each scenario
- **Step 2:** Determine optimal decision, being either supplies to stores or a shipment decision ▷ See Section 4.3
  - *Step 2a:* Build the model, dependent on the type of decision
  - *Step 2b:* Solve the model, extract and implement the here-and-now-decision in the next step
- **Step 3:** Finalise the iteration by updating relevant variables, such as inventory levels, or fixing the shipment decision ▷ See Section 4.4
  - *Step 3a:* Extract the demand in each store for the current stage
  - *Step 3b:* Update the inventory levels in each store

Compute all relevant summary statistics, such as total sales and total sold items

In the upcoming sections, we will go into more detail regarding these steps. Here, we stress two important factors. The optimal outcomes of the model, step two, are heavily dependent on the scenario tree built in the first step. Therefore, we need to perform this simulation multiple times to

investigate the stability of our method, such that we can make statements on the expected worst and best performances for a certain demand pattern.

Also, it could occur that certain demand patterns perform very well. On the other hand, the opposite can also be the case. Therefore, we need to run simulations with different demand patterns to investigate the performance of our methods in the long run. To make this distinction clear, we will refer to simulations where we use the same demand realisations in each stage as *replications*.

4.2 Creating the Scenario Tree

In Section 3.1, we expanded on the use of scenario trees. Now, we explain how we translate our historical data into a scenario tree using a nonparametric method. In Algorithm 2, we denote a detailed pseudo-code of how these scenarios are created. In the basis, we create scenarios based on the total number of sales in all stores combined. We then cluster the historical sales into groups by minimising, for each point, the total distance to the mean value of each cluster. Then, we divide the expected number of sales over the stores, based on the sales probability in that stage for a store.

Algorithm 2: Characteristics of building the scenario tree

Input: Historical data on total sales, sales probability stores, number of branches in each stage, sales progress so far (if applicable)

Output: Scenario tree

```
Create a root scenario to represent the stage before the first stage, which serves as the basis for the tree, with no demand
Adjust the data set if applicable, by for example merging stages                ▷ See Section 4.2.1
for stage t ∈ 𝒯 do
    for scenarios s ∈ Ω_{t−1} do
        Retrieve sales data in stage t of SKU's represented by scenario s
        Cluster sales data in stage t into a pre-defined number of branches, where each SKU is represented to the most nearby cluster mean   ▷ See Section 4.2.2
        for branch b do
            Determine the number of items sold and divide these over the stores by using the sales probability of stores in stage t   ▷ See Section 4.2.3
            Set the probability of an occurrence of b after s equal to the number of SKU's which fall into the total sales cluster of b
            Add s_b to the scenario tree, with b being a successor of s
```

4.2.1 Historical sales data

While we give more information on our data set in Section 5.1, the sales volume in stores is generally low. This makes it difficult to distinguish whether a product has a strong or weak sales progression in each store. However, if we aggregate the sales volume of all stores, we are able to distinguish a general sales progression. Therefore, we aggregate the sales of a product over all stores before we continue with creating the scenario tree. Also, we compare the sales progress on a level relating the number of sales to the number of ordered items, since products of which larger quantities are ordered are also likely to sell more.

Because the sales period could last for a long time, considering all stages in our scenario tree will have a significant impact on the solving time of the problem. Furthermore, one could argue that after a certain amount of stages, adding additional stages might not have a significant impact on the behaviour in the first stage, which we are actually interested in. We can use two methods to keep our scenario tree sizeable, by either chopping off scenarios after a certain stage, or by merging all stages after a certain time into one.

While both options could theoretically work, the latter will give us desired behaviour. It is not a downside if items sit in the inventory of a store for a few stages, since these could also get sold afterwards. Therefore, we aim to incorporate this into the model. On the other hand, we do not want items to be in the inventory of stores for too long, or with too many at the same time, which is why we using holding costs. However, items could still be stuck in a store's inventory. Deleting all stages after a certain point will give a signal to the model that it is not costly to keep inventory in the last stage of the tree, which is actually not the case. We could solve this by setting higher inventory costs in that last stage, but the model then aims for an empty inventory at each store. This causes difficulty to control the behaviour of choices made by the model, such that we choose to merge stages near the end of the sales period.

4.2.2 Create demand scenarios

In Figure 9, we see examples on the progression of total sales relative to the ordered quantities. While some products only sell 75% of the total products, others sell as much as 120%. This number surpasses the logical 100%, because items can be returned and resold, counting as another sale for that product. It can easily be seen that already in the first weeks large differences in sales exist.

We take a closer look at how our scenarios are constructed, by using an example of computing the branches in the first stage, where we can use our entire data set. The root scenario assumes no demand, from which branches on total demand will follow. We compute these branches by using our clustering method. These branches are chosen such that the total distance from each historical data point to its corresponding closest branch is minimised.

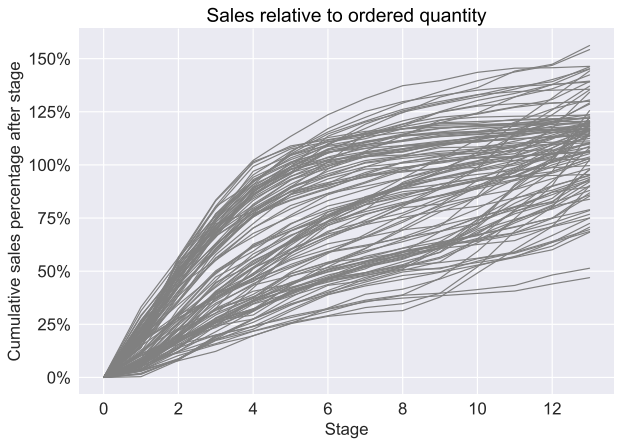
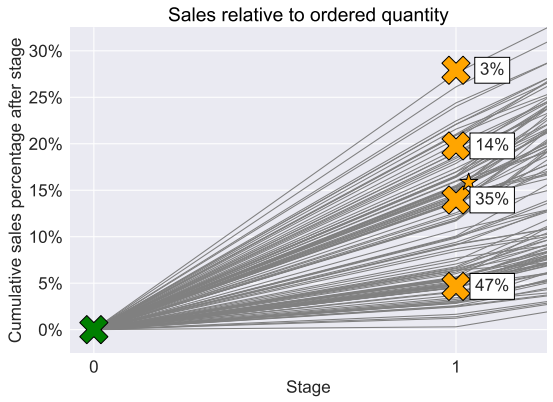


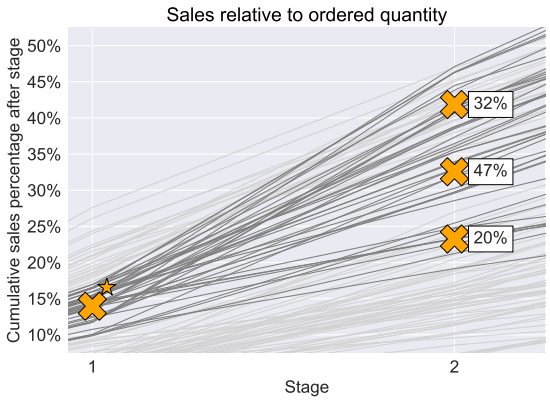
Figure 9: Example of the different possibilities on the sales progression of similar products.

In Figure 10a, we divide all data point into four branches. We mark the the root scenario at the left side of the figure in green. This scenario is the so-called parent of the computed four children. Then, we compute the probability of a branch occurring by calculating the number of data points in that cluster. Take for example the branch featured with a star. The probability of its occurrence equals 35%, meaning that 35% of all relevant historical data points fall within that cluster.

Each branch now consist of an occurrence probability and a total number of items that will be sold, but we need to divide that number of sold items over the stores. We will expand on this method in Section 4.2.3. After we completed this step, we created a scenario which we can add to our scenario tree. We do this for all four branches in Figure 10a, after which those four branches will be divided into successor scenarios too. In Figure 10b, we consider the process of creating children for the scenario marked with a star in Figure 10a.



(a) Creating children for the root scenario



(b) Creating children for a scenario in the tree

Figure 10: Example on the creation of successor scenarios for a given scenario.

The first step consists of retrieving all relevant data. By relevant data, we consider all historical data points in which total sales progressed similarly as to the branch we are currently working with. In the second stage, this means we are interested in the sales progression of historical sales which were divided into this respective cluster in the first stage. In Figure 10b, such data points are given a light grey colour, indicating their existence. However, only the darker grey data points are taken into account. Again, we create scenarios by dividing the number of items expected to sell in each branch over the stores before we add each branch as a scenario to the tree.

We repeat this approach until we are satisfied with the number of branches in and size of the tree, which is specified beforehand. It could occur that at some point during the branching only one or two items follow a certain sales progression, due to for example a very unique progression. In Figure 10b, we see for example very high sales in the second stage for some items. When we want to branch on the sales progression in the third stage, it might be a good idea to also include items which already had a high sales percentage in the first stage.

A similar problem is present when the starting stage of our model is not the first stage, such that we can not include all historical sales data. Here, one can try to find items with similar sales progression, but it is also possible to include all items which have a similar sales percentage at the current stage. We also use such data adjustment approaches in our computational experiments. In Appendix C.1, we go into more detail regarding enlarging the available dataset.

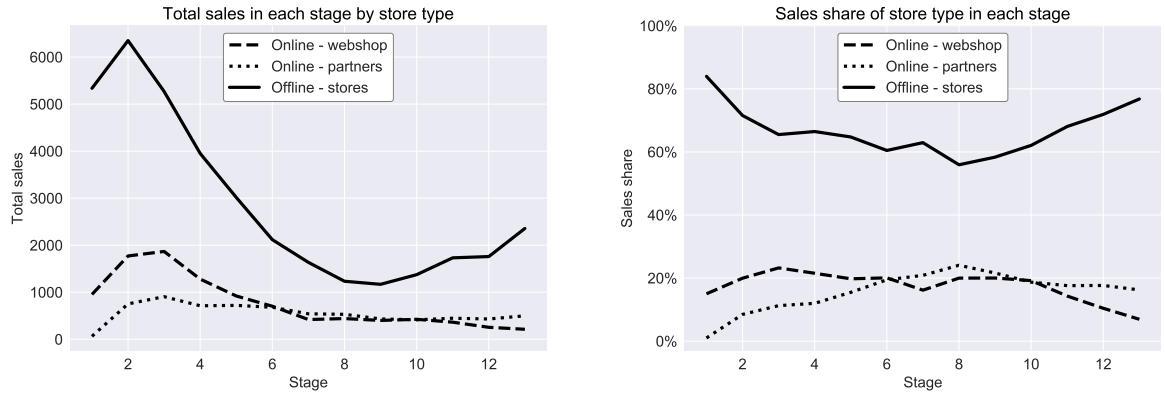
Lastly, the observant eye might have seen a difference between Figures 10a and 10b, considering the *best-case* scenario. In the first stage on which we branch, we also include a total sales percentage which equals the maximum in our historical data points. This is added to include a scenario with possible outliers, in an attempt to limit the probability of not satisfying all demand. This is an extra measure which can be viewed at in a similar fashion as to the minimum required number of items present at each store during the beginning stages of a sales period.

4.2.3 Sales probability in stores

In this section, we expand on how we develop a branch with a number on the expected items to sell into a scenario. This specifically entails the division of the expected sales over stores. Even though the historical data will normally consist of sales specified by store, we need to make some adjustments to the sales probabilities in order to create realistic simulations.

Most retailers can identify three different types of stores, two online channels and one offline. The offline store type are physical stores, of which there can be tens to hundreds. The online stores can be either sales directly on the retailer's website, or online sales via a partner. The first can be directly fulfilled from the distribution centre, while partners require their own inventory.

These types of stores could have different behaviour regarding their sales progress, which is not necessarily stable over time. An example of the sales progress compared to different types of stores can be seen in Figures 11a and 11b. Since the share of types of stores can change over time, we want to divide the expected sales in a scenario by setting the probability an item will sell in a certain type of store equal to the share of those stores in that stage.



(a) Absolute sales

(b) Sales share

Figure 11: Sales per store type in each store.

Because the sales behaviour at locations does not correspond with the average of all sales, we want to divide items based on probabilities an item will sell in a certain store. This information is known considering the historical sales data. We then correct these probabilities such that the total probability of selling in a certain type of store corresponds to the historical data. For example, in the second stage, all physical stores have a combined sales probability of 72%, while this is approximately 57% in the eighth stage. After correcting, we iteratively assign an item to one of the sales locations, based on these adjusted probabilities, until we divided the number of expected sales.

4.3 Use the Appropriate Model

In the beginning of this Chapter, we provided a schematic overview of the process in our simulation. There, we gave an insight in the difference per iteration. For the purpose of demonstrating the goals in each iteration, we repeat this schematic overview for iterations two to four in Figure 12.

It is clear that in iteration two and four, we look to find the optimal here-and-now-decisions regarding inventory allocation. In both cases, we look to solve the inventory allocation model. For clarity, we stress the subtle, but important, difference caused by the logical rules the model should adhere to as explained in Section 3.5. This means that in iteration two, this shipment decision is not yet final and can be different for scenarios occurring before the stage of the shipment decision. However, in stage four, this shipment decision is final and should be treated as such.

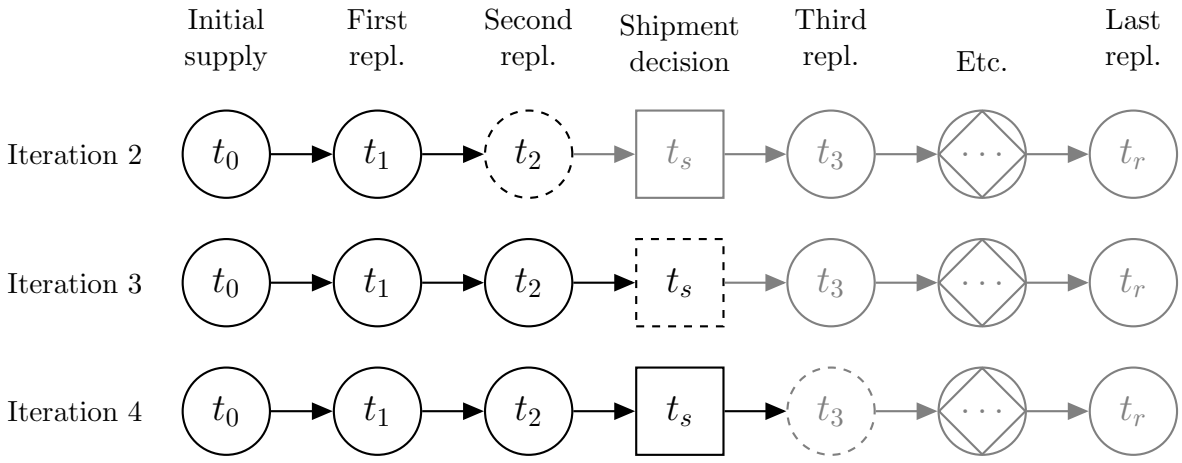


Figure 12: Iterative process of a rolling window simulation in an operations planning setting.

In iteration three, we make a decision on the flexible shipment. Here, we create the situation as in Figure 7b. In essence, the model remains the same as for the inventory allocation problems. However, we are now only interested in the shipment decision. The inventory allocation problem and shipment decision can technically be solved at the same time, because the problems are equal, since the shipment decision will be fixed for every scenario anyway. However, we choose not to due to the use of optimality gaps in the computational experiments.

4.4 Finalise Iteration

Before we can move on to a new iteration, we need to finalise the iteration by creating a realisation of the demand, if we are considering a replenishment. When we know the occurred demand in a stage, we update the inventory levels and returns, next to other relevant statistics such as sales and unmet demand. If we decided on the shipment, we only need to establish the certainty of the shipment arrival.

4.4.1 Create realisation of demand

The performed simulation is always based on the sales progression of a certain product. We use the historical data of the remaining products to create our scenario tree. Since we perform experiments in different settings, including a changing number of sales locations, we can not copy the exact sales information of products. Furthermore, we want to analyse the long term performance of our methods, thus we need to test with sales occurring in other similarly performing stores.

Again, we divide sales at the locations into three store types, similar as to in Section 4.2.3. We then rearrange the total sales of each store type over stores belonging to that type, where we use historical data to determine the sales probability in a certain store, on which we base this division. We do this separately per stage. In Table 2, we see examples of such rearrangements.

Table 2: Examples of rearranging demand to simulate different demand realisations.

		Stage 1			Stage 2		
		Original	Ex. 1	Ex. 2	Original	Ex. 1	Ex. 2
Webshop	A	4	4	4	3	3	3
Partner	B	1	2	0	1	3	1
Partner	C	1	0	2	2	0	2
Store	D	1	0	1	0	1	1
Store	E	2	1	2	1	1	0
Store	F	1	1	0	2	1	0
Store	G	0	1	1	0	0	1
Store	H	1	2	1	1	1	2

While the webshop is the only one of its type and thus will not see changes, the other two store types do. The two main visible takeaways are the unchanged total number of sales over the store types, and the fact that the division of the sales over stores does not necessarily need to change, as can be seen for the partner type stores in the second example of stage two.

4.4.2 Moving on to next iteration

The last step of the iteration corresponds to updating, among others, the inventory levels. We split the occurred demand into sales and unmet demand, after which we can start setting up all parameters for the next iteration. We finalise the decisions and occurrences during the stage handled in this iteration and go back to step one as in Algorithm 1.

5 Computational Experiments

In this section, we expand on the results of our computational experiments. First, we introduce details regarding our data set, while one can find our parameter settings in Appendix C. Also, we go into more depth regarding the benchmark in our experiments and possible modelling methods. In Sections 5.4 to 5.7, we will discuss the results of the different cases. For each case, we start by introducing the case and continue by giving more detailed results.

5.1 Data

In Section 4.2.2, we introduced an example of historical sales data. In our computational experiments, we will use the data set which also created the sales patterns in Figure 9. This data set contains the historical sales data of 164 similar products, divided into weekly sales in 221 stores. We also know the ordered quantity of each product, setting a more natural limit on the number of items which can be sold.

5.1.1 Data information

We introduced different store types in Section 4.2.3. Especially in our computations, this will become relevant, since we will not experiment on settings with all stores to control the computation time of our experiments. We always have one distribution centre, seen as an online channel from the retailer. Furthermore, for every fifteen stores included, we use one online channel from a partner, to compensate for the difference in selling rates of the store types. In the context of replenishing, they are treated the same as offline stores supplied by the retailer.

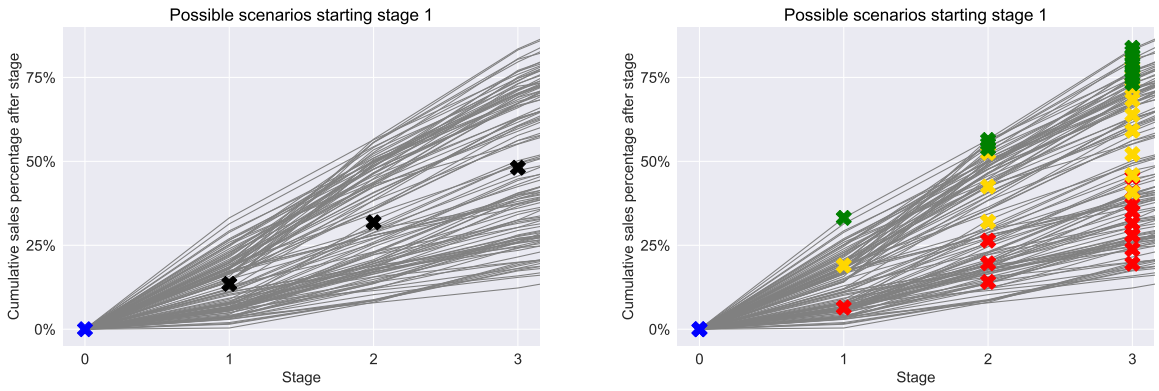
The products in our data set have a cycle life time of thirteen weeks. We modified the data as such that the best selling products have approximately 10% more demand than available items. The ordered quantity corresponds to offline stores being able to sell only a few items in demand during the sales period. This differs per product and per store, but generally ranges from two to five (roughly uniformly distributed). The online channels will then have a demand size adapted to the offline stores, based on the sales probabilities of each store type. Lastly, the ordered quantity is then corrected by the expected number of returns, depending on the return rate for each store.

5.2 Benchmark

Initially, we test our method against a benchmark, to identify how well our method performs. In this section, we give more details regarding the difference between our method and the benchmark. We already described the idea of the benchmark in Section 3.1.2, explaining how scenario trees are constructed. In all cases, we refer to this adaptation of our method as our benchmark. Also, we could denote the benchmark as *B.M.*, and our own method as *S.O.* (Stochastic Optimisation).

Here, we recap the difference between our stochastic method and classical forecasting (Section 3.1.2) and also state the consequences in our implementation. In essence, we can view the stochastic modelling of the inventory allocation problem as an advanced forecasting method. Rather than using one forecast, we use multiple and find an optimal allocation based on probabilities of each forecast occurring. Often, retailers use point forecasts, on which they base their allocation scheme. Since we branch on a number of scenarios in each stage, we can also branch on just one scenario. Then, we actually create point forecasts of the possible demand. We will use such point forecasts as our benchmark, since they represent what is used often in current inventory allocation systems.

Suppose we would predict possible sales at the start of the sales period, where we for example look three stages ahead. In Figure 13a, we show what would happen if we would adopt the point forecasting method. In this case, out of all options, we find the total demand which suits the possible scenarios best, given what happened in the past. We can quite easily see that this corresponds to finding the average of all scenarios (the black crosses), as opposed to our method in Figure 13b. Here, we give an example of a split into three scenarios in each stage, similar to Figure 10.



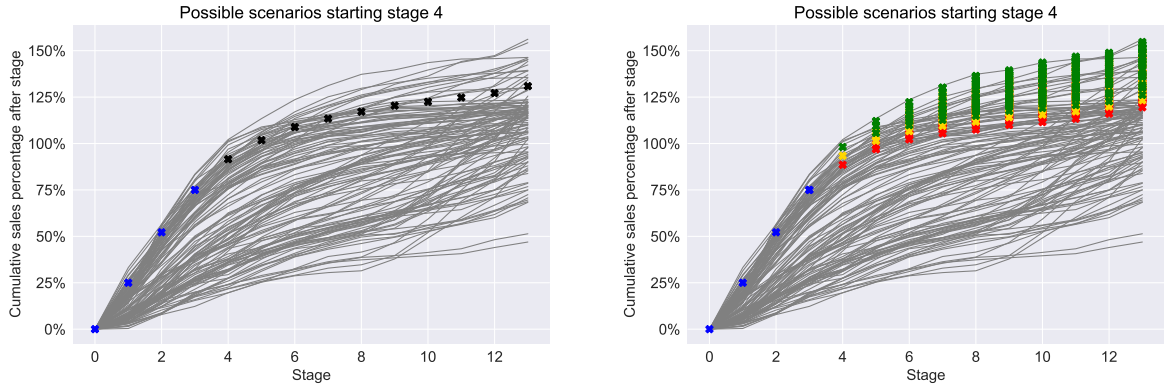
(a) Benchmark

(b) Stochastic method

Figure 13: Example on the creation successor scenarios for the original blue scenario.

In these figures, we can explicitly see the difference between the benchmark (on the left) and using the scenario tree (right), where creating a scenario tree results in many more possibilities the model should take into account. Similar to what we described in Section 4.2, also the possible scenarios of the benchmark method are updated based on the actual demand realisations in the past.

In Figures 14a and 14b, we show an example of such updated trees for both the benchmark and our method, for a above average selling product. The blue crosses represent demand realisations. These demands have already occurred before the creation of the scenario tree and this information on the sales progression can therefore be taken into account. While the benchmark methods also adapts is tree, it again only consists of one scenario, in contrast to our stochastic method.



(a) Benchmark

(b) Stochastic method

Figure 14: Example on the creation successor scenarios if there already is information on the sales progression, given in blue.

5.3 Modelling

In Section 3.3, we introduced two models with a slight difference. To recap, the first model can be seen as a lost sales environment, where every unmet demand is lost. The second, backordering, amounts to being able to fulfill unmet demand by backordering from the distribution centre. Nowadays, especially the latter is relevant, since retailers often offer the option to order the desired product online. For example, fashion retailers let a customer order a shirt online if their size is not in the store, but they know which size fits them by trying other similar shirts.

In our rolling window simulation, we can distinguish two dimensions. The first corresponds to choosing the model to use when determining the optimal inventory allocation scheme. It could, for example, be beneficial to be aware of the fact that backordering from stores is possible, even though each store would prefer to be able to sell a product to the customer directly.

Furthermore, we can alter what is possible regarding the actual demand realisations. In certain cases, for example for fashion retailers, many customers might order their desired product online in the store itself. On the other hand, bakeries can hardly sell anything they do not have in store anymore, thus resulting in any unmet demand to be lost, were it not be resolved by ordering something else.

Since many retailers would like to sell as many products to customers immediately, we can also combine the two models. We could for example use a lost sales model, while allowing backordering in reality. This way, we would expect the model to send as many items to stores as possible, to maximise the possible direct sales, while still allowing for backlogging. This is probably also the most viable option for use in practice if backordering is possible.

We investigate the results of these combinations in our first computational experiments, comparing our methods with different parameter settings against our benchmark (Section 5.5). In Table 3, we give an overview of the combinations in the two dimensions and abbreviations we can refer to in future result sections.

Table 3: Combinations of modelling in the rolling window simulation

Abbrevation	Used model	Reality
LS	Lost Sales	Lost Sales
BO	Backordering	Backordering
CLB	Lost Sales	Backordering

5.3.1 Controlling distribution centre inventory size

During the experiments, we encountered empty distribution centre inventories in early stages of the sales period. This occurred because initially we did not prefer keeping stock in our distribution centre or store, by setting equal holding costs. However, there actually is a difference in terms of when we would like our inventory in which type of store. Important to note is that we can not send items from stores back to the distribution centre.

Ideally, we would like our stores to only be able to fulfill demand of the upcoming two or three stages. Then, each store can anticipate on unexpected demand, while not holding too much inventory such that we can not solve other issues with inventory from the distribution centre. In terms of when to keep stock in which type of store, the distribution centre is then preferred from a few stages after the current stage onwards.

We can convert this into holding costs in such a way that it is cheaper in the first stage to keep inventory in stores, while in the last stage of the scenario tree it is cheaper to keep inventory in the distribution centre. Then, the model punishes for any leftover inventory such that it will try to minimise the inventory sent in the first stage.

Another option is adding a constraint controlling the inventory size of the distribution centre. By this constraint, we keep at least the number of items in the inventory of the distribution centre we expect to sell in the upcoming stages. Here, we do pay close attention to stockouts of stores within one day. In that case, the store can be supplied with the necessary number of items. The ultimate goal is to prevent stockouts.

We simulated both implementations of this idea, and while the minimum requirement of the distribution centre could be optimised further, differences are large. By adjusting the holding cost parameters, the total sales value is in general approximately 25% to 35% higher. Because this

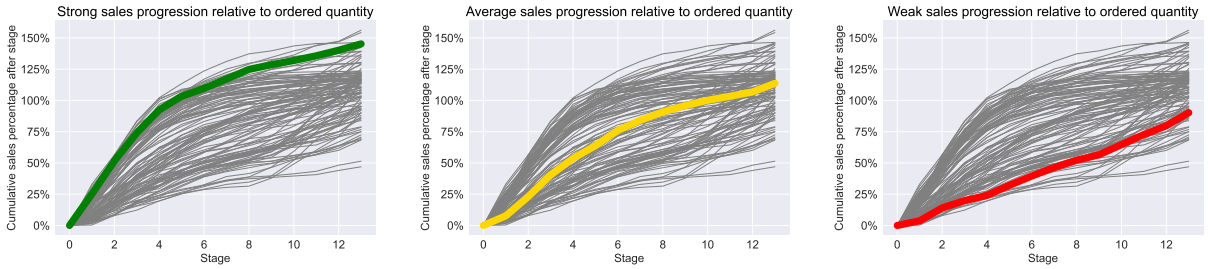
significant difference, we continue the remainder of our experiments with setting the holding costs as described above.

Also, the distribution centre might have an unlimited space, the stores generally have not. Although this practically is likely not an issue, we do want to take some capacity constraint into account because we solve the problem for each product separately. Therefore, we also include a capacity constraint requiring a maximum supply of two times the expected demand in each store.

5.4 Case: Benchmark Comparison - General Performance

The first case comprises a benchmark comparison. Besides denoting the average possible improvement in sales now, we later on test for stability of replications (Section 5.5.1) and sensitivity to parameters such as number of stores, sales volume and tree size (Sections 5.5.2 and 5.5.3). Furthermore, we comment on the computation time to identify key factors in applicability in practice (Section 5.5.4). These tests also form the basis of the other cases in Sections 5.6 and 5.7.

In the following results of our experiments, we mainly refer to three different products. These developed differently with regards to sales progression. In Figure 15, we show their sales progress compared to the entire data set. We refer to the best selling item of these three as having a *strong* sales progression (Figure 15a), while the other two are at an *average* (Figure 15b) and *weak* (Figure 15c) level of sales progression. If necessary, we comment on behaviour and results of products progressing differently.



(a) Strong sales progression (b) Average sales progression (c) Weak sales progression

Figure 15: Sales progression of different products.

For readability purposes, we will not comment on any stability and sensitivity results yet. We compare the methods in equal and representative circumstances. We use multiple simulation runs differentiating on the exact division of demand over stores, but we aggregate the results of all replications to an average. All results in this section results use 20 stores. Also, we use a tree depth of five stages, where we branch three times in the first three stages, and two times in the fourth and fifth stage.

While we show the results of simulations under similar circumstances, we do differentiate on multiple factors. We test products with different sales progressions, meaning both products selling above average and below average. Furthermore, we check how the methods perform in the different modelling combinations as presented in Section 5.3. Also, we evaluate the performance in low and high sales volume settings.

5.4.1 Lost sales model

In Figures 16 and 17, we show the improvement in direct sales to customers in the modelling combinations LS and CLB (Section 3). We run four simulations with different demand in the stores. We replicate the entire process (Chapter 4) eight times and aggregate those results to get to four approximate improvements in percentages. We plot the aggregate results of the same product on the same row, where the product displayed on each row corresponds to the type of sales progression denoted between the split on the low and high volume case.

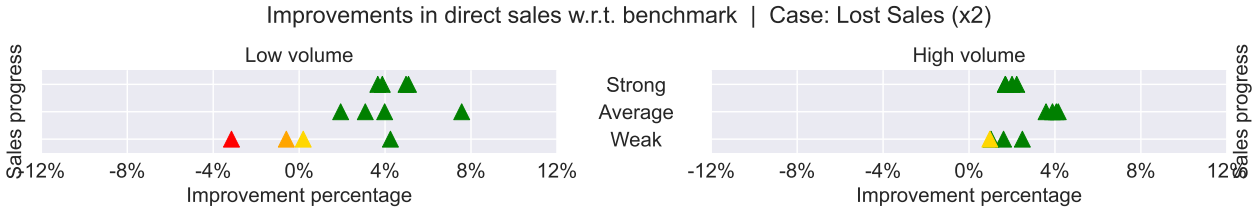


Figure 16: Improvement in direct sales with regards to the benchmark.

In case all unmet demand is lost immediately, we clearly see an improvement in sales to stores. This holds for both low and high volume cases, even though the exact percentages are more variable for low volume cases. This can be explained by the fact that the diversion in demand over stores is greater in case a retailer would only sell a little over one hundred items.

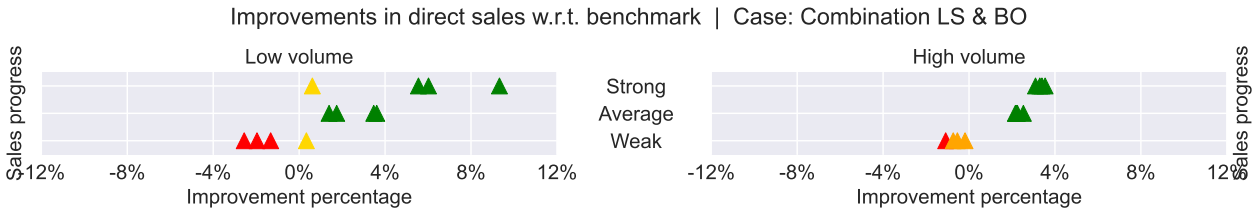


Figure 17: Improvement in direct sales with regards to the benchmark.

When we allow to fulfill any unmet demand via ordering at the distribution centre, we see a similar pattern. The results of the product with a strong sales progression are even more diverse. However, these remain mainly positive.

For both modelling combinations, the product with a weak sales progression does not show improved sales. In the low volume case, the performance even declines by a few percent. In general, the performance declines if a product sells less relative to the ordered quantity, thus having a weaker sales progression. This can be explained by the nature of our method, where we do not send items to stores if it seems unnecessary. Consider for example Figure 18. With the black line, we draw the total number of supplies to stores using our benchmark, while the red line displays the same cumulative number using our stochastic method.

(a) Strong sales progression (b) Average sales progression (c) Weak sales progression

Figure 18: Cumulative supplies to stores in a low volume setting while allowing backordering.

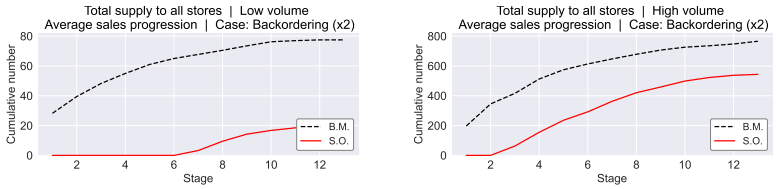
We can easily see that for the product where the sales progression is strong, the number of supplies of both the benchmark and our stochastic method are similar. However, when an item is less popular, our method seems to put more weight on not sending items to stores. The retailer is then able to fulfill less demand directly. In reality, the distribution centre still has plenty of inventory in stock. In fact, we could send extra items to stores while not depleting the distribution centre. Doing so in reality will make us able to fulfill any unmet demand in our simulations caused by simply sending too few items to stores. Then, the performance of the items with a weaker sales progression will be at least similar to the benchmark.

We did not include adaptations as above to the supplies to stores in our modelling, since we purely want to look at the performance of the method. We keep the division of items to stores as generic as possible, while we can improve on the exact supplies to stores by for example adapting for any illogicalities in practice.

5.4.2 Backordering model

If we use a backordering model, the total number of supplies sent to stores diminishes significantly. Even though we modelled the costs of backordering such that we always prefer sending to stores if the sales probabilities are high (Appendix C.2), it seems that the level of uncertainty causes the model to prefer keeping items in the distribution centre. We show this in Figure 19, using an example displaying the total supply to stores, similar to Figure 18.

The observant eye will notice that the supplies of our benchmark remain equal, where the low volume cases of Figures 19a and 18b are comparable. Since we only have one forecast, we will make



(a) Low volume (b) High volume

Figure 19: Cumulative supplies when modelling backordering, for an average sales progression.

an optimal plan according to those forecasts, causing the option for backordering to be useless for the model. In our stochastic method however, the total number of supplies diminishes largely, especially in the first stages. In a high volume setting, the relative effect is smaller, but still undeniable.

Similar to the weak sales progression case using a lost sales model, we can correct for such behaviour in reality. This could be done by setting the backordering costs almost as high as the sales price, such that we essentially revert back to the lost sales model. This, however, is not the purpose of our comparisons and could make the model vulnerable to the parameter values. The latter is undesirable, since we would like our method to perform well in a general setting.

In case it does not matter whether backordering occurs, we can compare the modelling combinations using all sales. In Figures 20 and 21, we show the simulations results, as in Figures 16 and 17. Since we want to check for some level of backordering, we decrease the sales value of backordered items by 1. If using backordering in the model causes the total sales to remain equal or higher than using a lost sales model, there is a clear benefit.

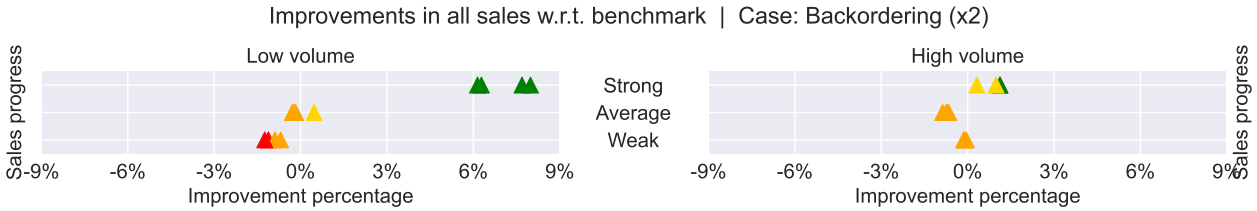


Figure 20: Improvement in all sales with regards to the benchmark.

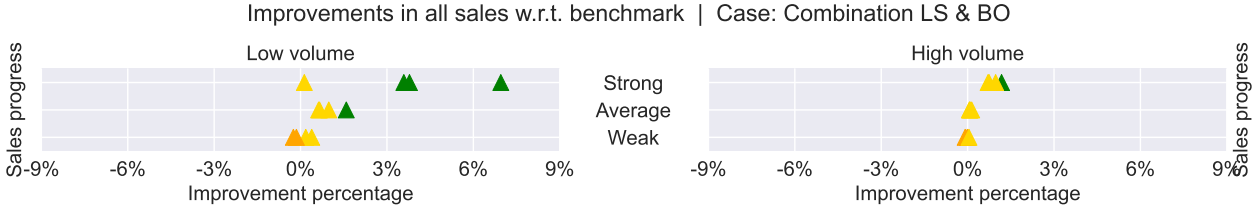


Figure 21: Improvement in all sales with regards to the benchmark.

However, allowing for the possibility of backordering in our model does not seem to positively impact the total sales. Therefore, we can conclude that it does not really matter whether we do or do not allow backordering in the model. Only in a low volume setting and for a product with a strong sales progression, a backordering model probably performs better. With the decreased supplies to stores due to the backordering possibility in the model, the total sales value of backorders is higher than in the lost sales model. The decreased value per item then causes the total sales to be lower.

When comparing the performance of our method to the benchmark, we see that the percentage improvements decreased for both the low and high volume cases. This is caused by allowing the retailer to fill up any gaps left by not having enough inventory, in both the benchmark and stochastic method, since we compare the total sales including backorders. This automatically decreases the difference between the two methods.

5.4.3 Conclusion

From these results on the general performance, we gather three main insights. First, the lost sales model ensures a better performance then a backordering model, with regards to effectively allocate inventory to achieve direct sales. Furthermore, our stochastic method works best if the remaining inventory size in the distribution centre is not sufficient enough to supply each store with an error margin, such that we for example have to choose between supplying either store A and B. Last, if the sales volume of a product is higher, the expected improvement on the direct sales value with regards to the benchmark becomes more stable, but not necessarily better or worse.

5.5 Case: Benchmark Comparison - Sensitivity

Now, we investigate the sensitivity of the stochastic method. In Section 5.5.1, we discuss the stability of the results. Also, we consider the impact of the two main factors with a significant impact on the computation time, the number of stores included (Section 5.5.2) and the scenario tree size (Section 5.5.3). These two factors are responsible for a growth in the number of variables, and apart from these two, only the number of items included in the model seems to have an impact on this computation time. We therefore also comment on differences between a low and high volume setting in Section 5.5.4.

5.5.1 Stability

The figures displayed earlier show aggregated results on the average performance on the long term. However, it is also important to investigate if the results remain constant or consist of highs and lows. A more reliable method can be implemented easier in practice, for example because a retailer is able to draw up a business plan more easily in case of a more predictable result.

Figures 22 and 23 show the results of eight replications of four different simulations using the lost sales modelling and reality. The grey circles represent the benchmark replications and are centered around their own average. The coloured triangles represent the improvement of the stochastic method compared to that average performance of the benchmark and are our main points of interest.

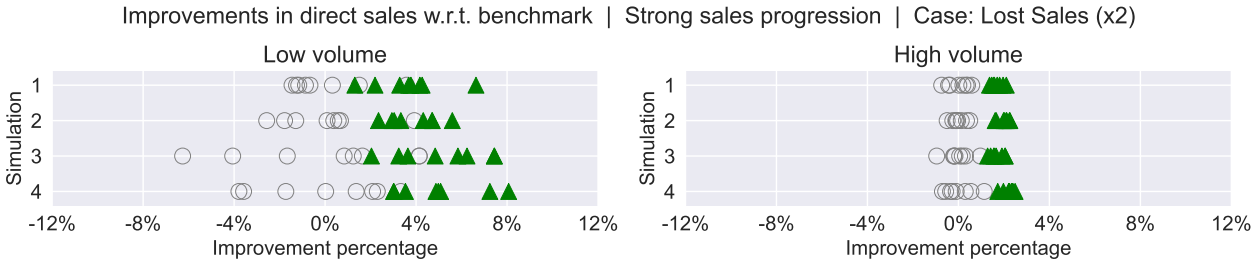


Figure 22: Improvement in direct sales with regards to average of the benchmark for different replications in each simulation.

In case of a strong sales progression, we see that all simulations have positive results, which we would expect given the results in Figure 16. Only a few replications of the benchmark outperform the stochastic method. Especially for the high volume case, this rarely seems to be the case.

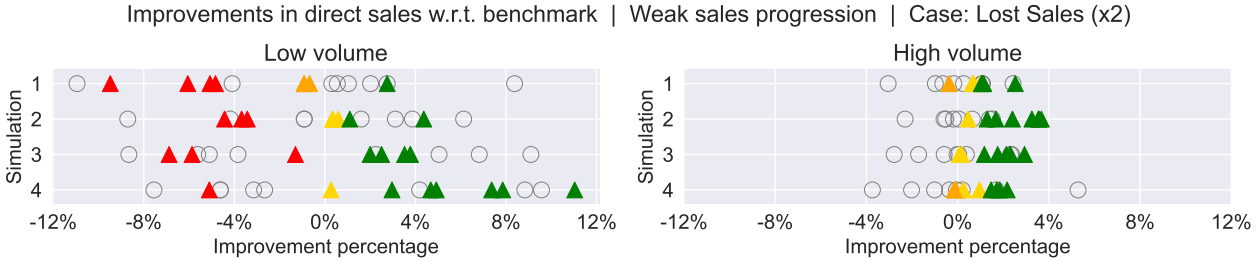


Figure 23: Improvement in direct sales with regards to average of the benchmark for different replications in each simulation.

However, this is different for a weaker sales progression (Figure 23. The performance is widely spread, and ranges in the fourth simulation for the low volume case even by over fifteen percent. This emphasizes the not so realistic performance caused by selling few items. If we for example consider the high volume cases, the results are less diverse and even more positive.

The stability results of other products and also the other modelling combinations are rather similar. The high volume cases are stable in the sense that if the average outperforms the benchmark, most replications will too. The low volume cases are more diverse, where the performance of the benchmark replications and stochastic method are more intertwined.

Needless to say, this is less desirable. However, even in its variety regarding this improvement,

our stochastic method performs better in the long run. For example Figure 24, showing the results of a product with an average sales progression using the modelling of lost sales, while allowing for backorders in reality, displays this very well. In many replications, our method performs better than nearly all benchmark replications. If not, it mostly performs better than the average of the benchmark, whereas the benchmark could also always perform worse than the average.

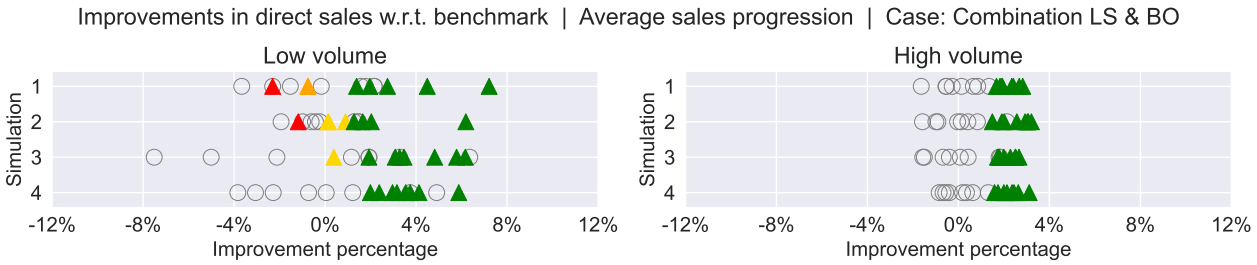


Figure 24: Improvement in direct sales with regards to average of the benchmark for different replications in each simulation.

5.5.2 Locations

All results shown so far were computed using twenty locations. Now, we show results of ten and thirty locations, to see if these will change. We also ran a few simulations with a larger number of stores, but the combination of increased computation time and the indication that those results do not differ compared to the results below, refrained us from further testing.

In Figure 25, we compare the results of a different number of stores for products with a similar average sales progression. In each simulation, the demand in each store is approximately equal. Thus, we increase the total number of available items from a product if the number of stores go up, for a fair comparison. We could for example expect that when more stores are included, results of different simulations are closer to each other, because the risk of missing demand is spread over more stores. With such diversification, we might improve the results on average and achieve more stability. The results are aggregated over a number of replications, as in Section 5.4.

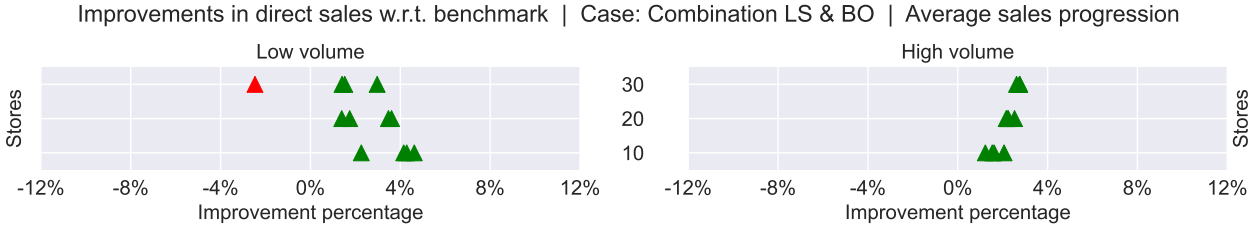


Figure 25: Sensitivity of results regarding a different number of stores for a product with an average sales progression.

We see a small difference in performances for all number of stores. While the low volume setting tends to a worse performance if the number of stores go up and vice versa for the high volume setting, this does not seem the case if we consider also other products. The exact average performance for each number of stores seem to depend on what fits the considered product best.

Also, the results of the product with an average sales progression are rather stable in terms of average. If we look at a more dispersed set of results (Figure 26), the similarity regarding number of locations does not change. Especially for the low volume setting, the results are unstable. It therefore seems that the number of included stores does not seem to have an impact on the actual performance of the model.

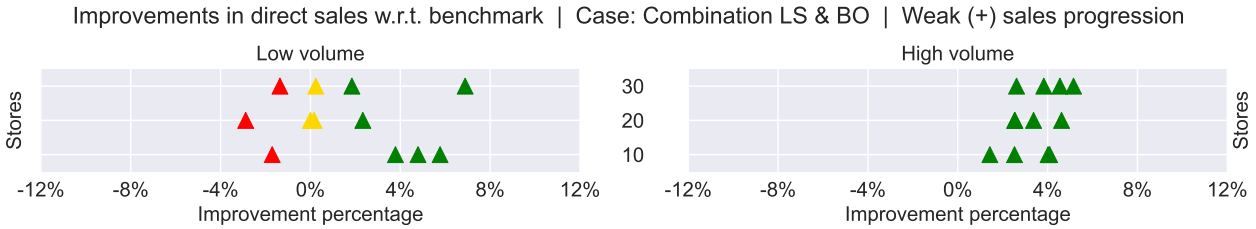


Figure 26: Sensitivity of results regarding a different number of stores with a little better than weak sales progression.

5.5.3 Tree size

All results displayed so far use the same branching pattern (Appendix C). The benchmark uses only one branch in each stage, corresponding to only one scenario per stage. In this section, we show results of other simulations where we vary the tree size. In Table 4, these variations are displayed in detail. Note that size *M* corresponds to the pattern of all earlier results.

Table 4: Example of number of branches. consequent scenarios in each stage in the scenario tree and total number of scenarios for the sensitivity analysis.

	Stage	t	t+1	t+2	t+3	t+4
S	Branches	2	2	2	2	2
	Scenarios	2	4	8	16	32
	Total number of scenarios: 62					
L	Branches	4	4	4	3	2
	Scenarios	4	16	64	192	384
	Total number of scenarios: 660					

	Stage	t	t+1	t+2	t+3	t+4
M	Branches	3	3	3	2	2
	Scenarios	3	9	27	54	108
	Total number of scenarios: 201					
XL	Branches	6	5	4	3	2
	Scenarios	6	30	120	360	720
	Total number of scenarios: 1236					

We display results of individual replications compared to the benchmark average of each simulation, to make the variance of the tree size clearer. Figure 27, results for an average sales progression, is a

good example of this influence. The high volume setting is rather stable when using more scenarios, while the low volume setting is not. Here, a larger number of scenarios has a big influence on the results. For example, both a small and medium tree show declines in sales, while the extra large tree only shows positive improvements.

Figure 27: Sensitivity of results regarding a different tree size for a product with a weak sales progression.

The maximum improvement for the extra large tree does not become better though, which could be due to this specific product. For the product with a strong sales progression, that percentage does improve. In Figure 28, we see a shift to the right for the better part of the results interval.

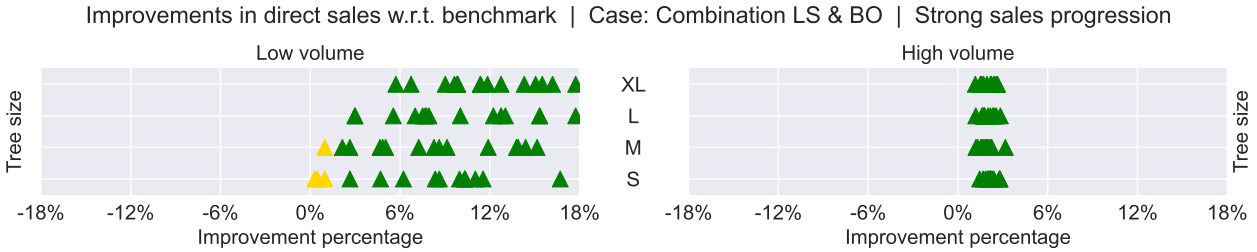


Figure 28: Sensitivity of results regarding a different tree size for a product with a strong sales progression.

We must note that these figures give a view on improvements where the models are solved with an optimality gap. If the optimality gap is set closer to 0, the results are likely to improve more. This will especially be the case for the high volume settings, where a gap of 1% could have a significant impact on the results if the scenario tree becomes bigger because of the growing number of possible scenarios.

5.5.4 Computation time

In this section, we first comment on the general computation time of the stochastic method, both in terms of number of items and a stage by stage comparison. Then, we investigate the influence of the two variables with the largest impact on computation time, the number of locations and the tree size.

In each simulation, and especially at the beginning, it takes the models a long time to reach the optimal solution. Therefore, we use an optimality gap to solve our models in each stage (Appendix C.2). Since we are only interested in the here-and-now decisions, our results are likely not impacted heavily by this decision, although this is definitely a point for further research. This does impact the results of our computation times, where an earlier stage has less of a gap to solve, which should be taken into account when interpreting the results.

The following results are computed on a laptop with an Intel i7-7500U processor. Its clock speed is 2.70GHz and it has a RAM of 8GB. Also, the simulations use the multiple cores for different experiments simultaneously, which resulted in often having only 1 core available for the solver (CPLEX Optimization Studio 12.10.0 used in a Python environment of Anaconda Spyder). Figure 29 shows the general computation time in each stage using the parameter settings as in Appendix C, solving for one product at a time.

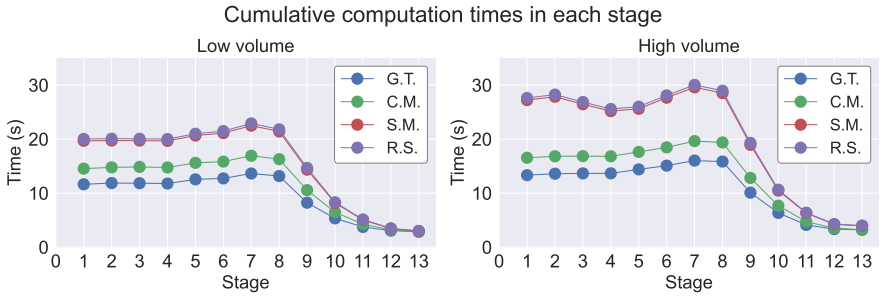


Figure 29: Cumulative computation time per stage, split for generating the tree (G.T.), creating the model (C.T.), solving the model (S.M.) and creating the realisation (R.S.).

In both settings, we see that the computations times remain stable if the last stage is not within reach of the scenario tree depth. Taking a closer look at the different parts influencing the computation time, we see that the generation of the scenario tree takes up approximately 50% of the time, while the models are solved to that optimality gap in a couple of seconds. We will go into the difference of the low and high volume setting later on.

Next to this base image on the computation times, we are also interested in the influence on the computation time of the parameters we mentioned earlier. We will split the main three parts (generating the tree, creating the model and solving the model) into different figures. Since the patterns are similar for both the low and high volume setting, we only show the results of the low volume setting.

Figure 30 shows the impact of a varying number of stores. The impact on the scenario tree generation is logical. However, the model solving time nearly doubles when ten stores are added. We see this pattern repeating itself if we include ten more stores, but we did not include this due to the eventual long simulation time for a consistent result.

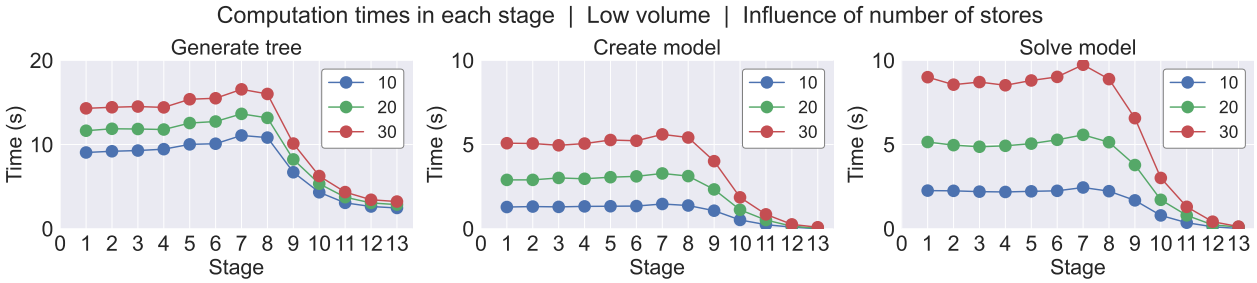


Figure 30: Computation time for a varying number of locations in a low volume setting, including 10 (blue), 20 (green) and 30 (red) locations.

Therefore, expanding this model to over a hundred stores will become very costly in terms of computation time, especially if we also for example use a larger scenario tree. To amplify, we use the same trees as in Section 5.5.3, of which we again split the computation times into parts as for the different number of locations. We show results for a low volume setting in Figure 31.

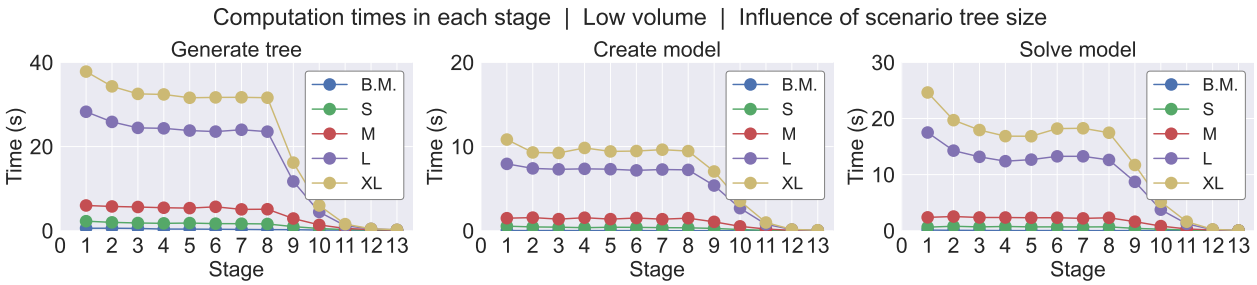


Figure 31: Computation time for a varying scenario tree size in a low volume setting.

While the number of scenarios tripled (or doubled in the extra large case) for each larger tree, the computation times show a different pattern. Using a large instead of medium tree, the computation times jump up in all aspects. However, the extra large tree does not double in computation time, which the number of scenarios does. In fact, they stay rather close to the large tree. For the solving part of the simulation, this is likely due to the optimality gap.

In the high volume setting (Figure 32), the computation times are approximately equal to the low volume setting, except for solving the model, taking twice to three times as long. In the eighth stage, that computation time spikes. Here, the model has an additional factor of the last stage in the tree, where each leftover item needs to be disposed of, albeit for a salvage price, apparently impacting the solving time significantly. This difference between the low and high volume setting shows the, possibly very significant, influence of the number of items mostly on the model solving time, especially when the tree is larger and more locations are used.

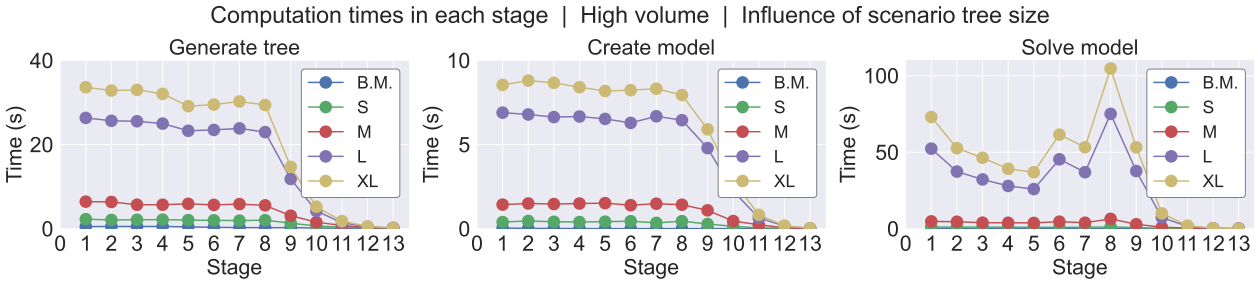


Figure 32: Computation time for a varying scenario tree size in a low volume setting.

5.5.5 Conclusion

In the stability experiments, we clearly saw an unstable performance in a low volume setting and for items with a weaker sales progression. The high volume setting shows more stable results even for all replications and different sales progression.

While changing the number of locations included in the supply network does not seem to have a significant and consistent impact on the results, the tree size does seem to have a positive impact on this total sales value. The scenario tree size is especially relevant in a low volume setting, where getting an additional sale can have a large impact on the results.

Considering the computation times, the time to get to a decision on the inventory allocation decreases when the last stages of a sales period are reached. In all stages though, generating the scenario tree generally consumes a signification portion of the total time needed, but solving the model will take a more prominent role if either the number of locations or the scenario tree grows. Also, especially the combination of a high volume setting and an (extra) large scenario tree has a significant impact on the computation time in all stages.

5.6 Case: Split Shipment

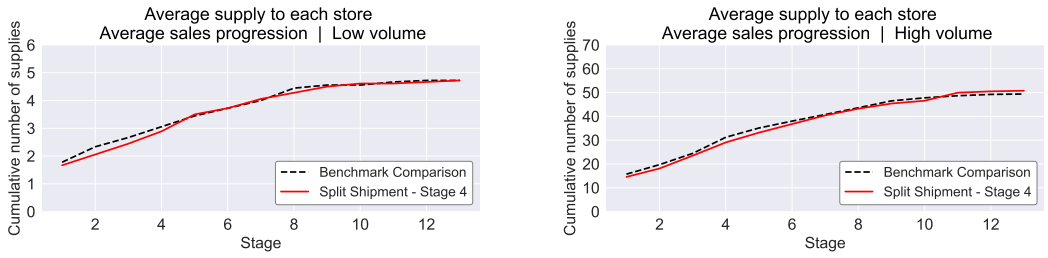
In Section 2.3, we introduced the idea of splitting the shipment of all inventory into multiple parts, because the distribution centre of retailers could be flooded in case of large batch sizes. The resulting question here is then whether the later arrival of a part of the batch impacts the sales value.

To answer this question, we apply the proposed method as in Section 3.4 and compare the results with those of our initial experiments in Section 5.4. We stick to considering the modelling combination of lost sales and backordering in reality, since this combination comes close to reality, while maintaining a high direct sales level because of the lost sales modelling.

The most important factor to consider now is the total supply to stores. If we have less items in inventory at the start of the sales period, our method might be more conservative in allocating

inventory, causing sales levels to drop. Therefore, we show the average number of supplies to each store in a low and high volume setting in Figure 33, for the product with an average sales progression.

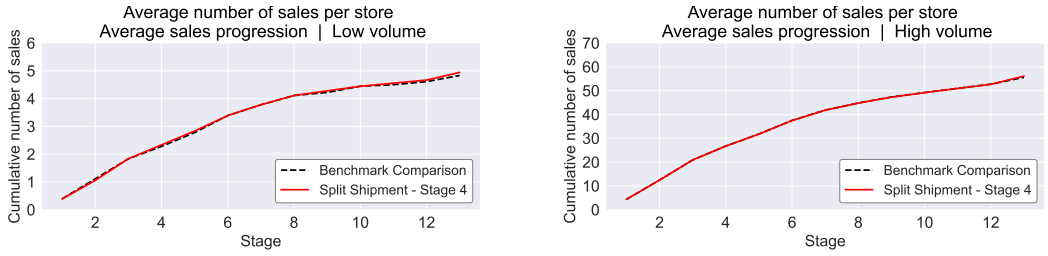
Here, items in the second shipment can be shipped to stores from the distribution centre from stage four onwards. The red line displays the average number of supplies using a split shipment, while the dashed black line shows the supplies in the base case of the benchmark comparison as in Figure 18. These numbers are averaged over multiple replications of a single simulation.



(a) Low volume (b) High volume

Figure 33: Cumulative supplies to stores, comparison base case and split shipment for an average sales progression.

In both low and high volume settings, we see no clear behaviour change with regards to the average number of supplies sent to stores. These numbers dropped by a small percentage, which in turn caused no consequences for the average number of sales in each store (Figure 34), such that the sales level remains stable. Therefore, the lower number of items in the distribution centre at the start of the sales period seems to have no impact on the behaviour of our proposed stochastic method with regards to decisions on supplies to stores.

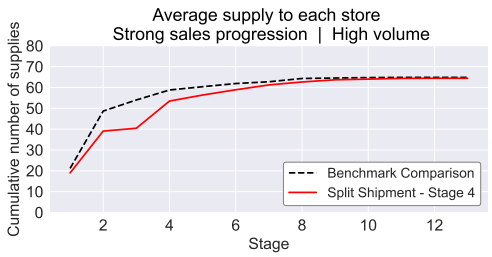


(a) Low volume (b) High volume

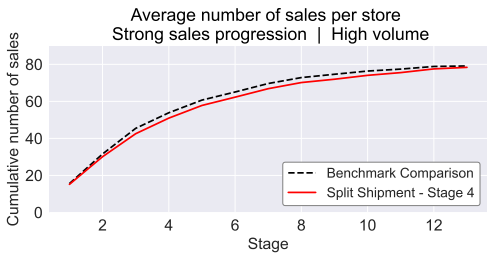
Figure 34: Cumulative number of sales in stores, comparison base case and split shipment for an average sales progression.

An important note is that for the product shown, an arrival stage of stage four would on itself be enough to let the distribution centre not be depleted before the second shipment arrives. This changes for a product with a strong sales progression, where the number of available

items quickly diminishes and we do have a stockout at the distribution centre early on. This impacts the supplies to stores (Figure 35a) and thus also the number of sales in each store (Figure 35b).



(a) Average supply to each store



(b) Average number of sales of each store (without DC)

Figure 35: Comparison base case and split shipment for a strong sales progression in a high volume setting.

The impact on the sales number in stores other than the distribution centre however remains acceptable, since the extra batch of items arrives just in time. Especially in stage three, an impact is present. The observant eye might see that the number of sales is higher than the until then provided supply, which is possible due to returns to stores, allowing to resell items. In the stages afterwards, the extra batch has arrived and the inventory and sales levels are back to the level of not having a split shipment.

In these figures, we did not include the distribution centre to prevent any haziness, since this store does not receive supplies in each stage, while it has the highest sales percentage of all stores in our experiments (Figure 11). Only if the additional shipment is not on time, as is the case when we have a strong sales progression and an arrival in stage four (in this data set), a large portion of demand at the distribution centre can not be fulfilled.

The main result of the experiments in this section is that there does not have to be an impact on the sales value in the case of a split shipment. In fact, since the sales levels remain equal, we still improve on a standard forecasting method where all items did come in at the start of the sales period. A logical, but important note here is the necessity of the additional shipment arriving in time to prevent a stockout at the distribution centre. However, in general, a split shipment does not impact the performance of our stochastic method.

5.7 Case: Flexible Shipment Arrival Time

The result of maintaining the same sales level, while using a split shipment, is very useful for the problem as introduced in Section 2.4. We can now safely split the initial shipment into parts, as long as the later shipment arrivals do not cause a depletion of the inventory at the distribution

centre. In fact, this result means we are able to choose the arrival stage of such later shipments optimally. In this case, we will investigate whether it is also beneficial to make such a split shipment flexible and maintain similar sales levels.

During initial experiments, we noticed that the model can, only in the low volume setting, decide on when to let a shipment arrive within a few minutes (with an acceptable optimality gap). This is not the case in the high volume setting. It takes the model over ten minutes, on our machine, to even find a solution. We will refer to this choice on shipment arrival as the shipment decision.

Furthermore, in the achieved results of the low volume setting, the shipment decision is very dependent on the parameter settings, especially the holding costs. Low holding costs causes it to have little impact on the decision, while high holding costs make the decision too dependent on the scenario tree. Finding the golden mean is not a generic process, which causes very different decisions for different replications. This causes the need to solve the model to optimality, of which the stability and correctness on decisions also remains questionable.

Because we want to make a stable decision in every setting, of which parameters do not need to be adapted to the specific product, we decided to solve this problem differently. The theoretical model in Section 3.5 still forms the basis of how the problem should be tackled. However, we create a fast and universal heuristic, to be applied in all circumstances. We give more details on our implemented heuristic in Section 5.7.1. Then, we split the results into commenting on the reliability (Section 5.7.2) and performance regarding sales value (Section 5.7.3) separately.

5.7.1 Heuristic

We create a fast check based around the scenario tree identifying when a significant probability occurs that we are not able to directly fulfill the total demand any longer. When we determine the arrival stage of the split shipment, we have a certain inventory size from the first shipment left. Then, we can track the probability of total cumulative demand surpassing the leftover items for each following stage. If that risk of unfulfilling demand is too high, we can conclude that we the shipment needs to arrive before that stage. In Algorithm 3, we denote the pseudo-code of our heuristic to find the last stage the flexible shipment could arrive to prevent stockouts at the distribution centre. To clarify, this decision is made after some stages have already occurred.

In this heuristic, we aim to prevent the distribution centre from a stockout. This is opposed to finding out the optimal stage in a mathematical model, where it might be better to delay if it saves costs and the consequences of maybe having too little stock are small. However, since it is not desired for any retailer to miss out on any potential demand, this is justified because retailers want to fulfill as much demand as possible.

Algorithm 3: Heuristic for finding the optimal arrival time of a flexible shipment.

Input: Scenario tree, α defining the accepted risk of having unmet demand, current inventory levels and earlier occurred demands

Output: Last stage f, where the split shipment can arrive

```
Define 𝒯 as all stages in the scenario tree
Calculate N as total inventory leftover
for stage t ∈ 𝒯 do
    Calculate total demand to and including stage t for each scenario occurring in stage t
    Order all demands and corresponding occurrence probabilities (per scenario) to create
     an Empirical Cumulative Distribution Function (ECDF) F(x), based on the demands
    Set χ as the maximum x (total demand) for which F(x) ≤ 1 − α
    if χ > N then
        The probability of total demand being larger than leftover inventory is too
         significant, thus the shipment should have arrived in stage t − 1
        Set f = t − 1 and break out for loop
```

Compared to using a mathematical optimisation model, we therefore might be a little more conservative. However, it seems like we do solve our earlier problems. First, we find the preferred stage in a fraction of a second. We should even be able to expand the scenario tree to be more accurate, without sacrificing on the side of computation time. Second, this method is not vulnerable to parameter settings, but only to the scenarios, as desired. If the decision is unstable, this only regards two subsequent stages.

5.7.2 Reliability

In reality, we would like to make the decision on the shipment as soon as possible. However, as described earlier in Section 2.4, this method will definitely not work if we can not use any information regarding the sales progression of an item. Therefore we determine the optimal shipment arrival stage after two weeks in our simulation.

We evaluate the performance of the heuristic by using both our stochastic method (S.O.) and the benchmark (B.M.) as in Section 5.2. The benchmark then corresponds to applying the heuristic to single point forecasts, thus the shipment decision is equal to the stage before the total demand would surpass the leftover inventory for the single point forecasts. Because the heuristic is based on using a scenario tree, we expect the benchmark to perform worse in most cases.

In Figures 36 and 37, we show the results of applying this method for the two products with a strong and average sales progression we saw earlier (Figure 15). On the left hand side, the low volume setting, and on the right a high volume setting. The height of the bar represents the

percentage with which a given arrival stage for the flexible shipment is chosen, separated by either the B.M. or S.O. This percentage is based on six replications of two different simulations.

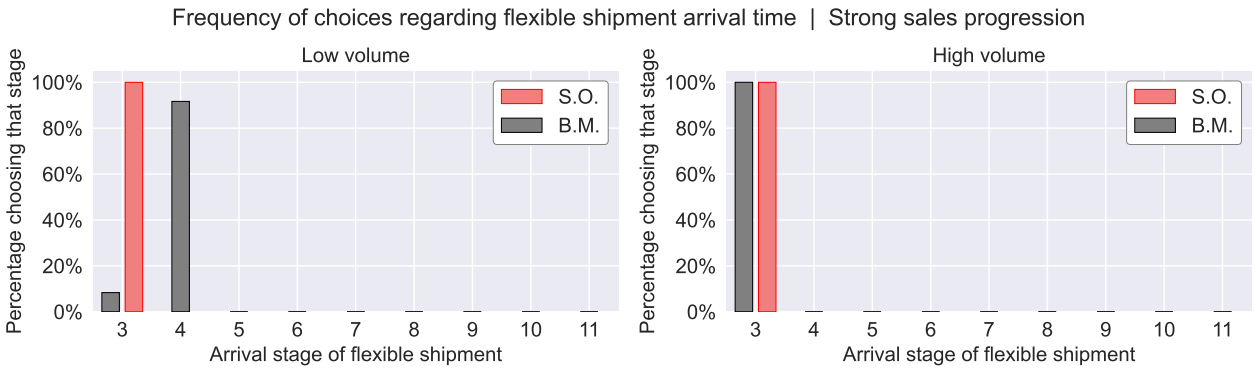


Figure 36: Percentage of times the heuristic chooses a certain arrival stage, split for the benchmark method (B.M.) and our stochastic optimisation (S.O.).

We see a clear difference between the low volume and high volume setting, where the choice in the low volume setting is not so reliable, as opposed to the high volume setting (especially Figure 37). This pattern repeats itself for other products too, being unstable for low volume settings, while it is stable for high volume settings. In that high volume setting, it often chooses the stage chosen more frequently in the low volume setting. An odd product with different behaviour is the example of the product with the strong sales progression (Figure 36). This reliability behaviour occurs both for the S.O. and B.M.

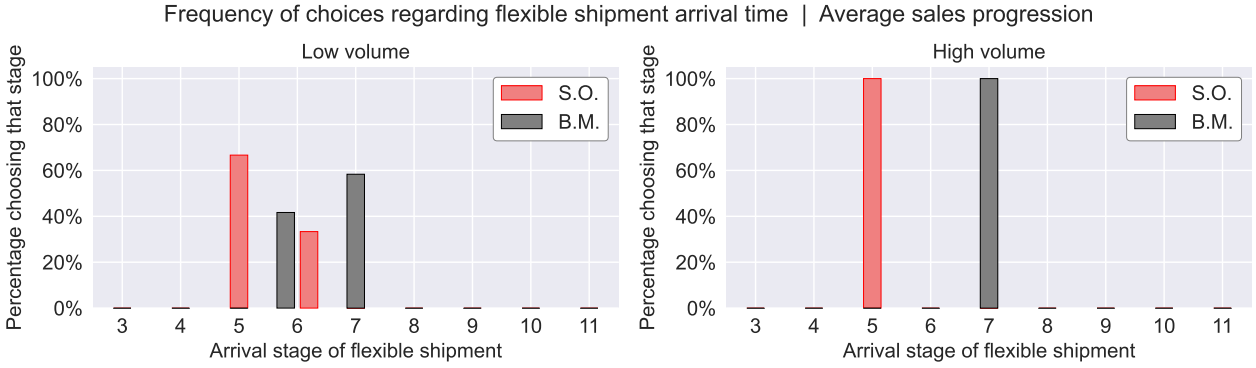
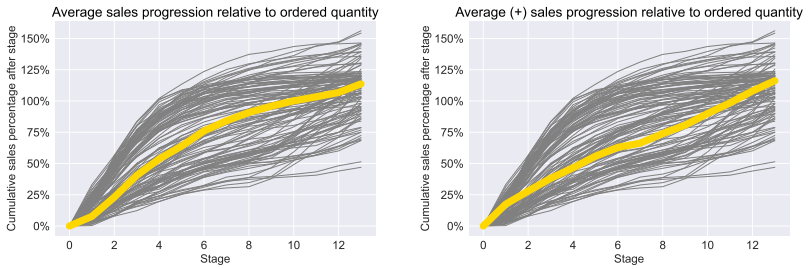


Figure 37: Percentage of times the heuristic chooses a certain arrival stage, split for the benchmark method (B.M.) and our stochastic optimisation (S.O.).

A closer look at the difference between the choices of the B.M. and S.O. (especially in Figure 37), shows us that using S.O. results in choosing one or two stages earlier compared to the B.M. This also corresponds with the behaviour regarding their differences for other products with both a similar or different sales progression.

In Section 5.7.3, we will go into whether choosing an earlier stage is actually beneficial or whether our S.O. is simply overconservative. Before we move on, we highlight another interesting fact. Our exemplary product with an average sales progression has better sales in week two and three compared to the first week, which we can see as a *warm-up period* (Figure 15b or 38a). For another product without this warm-up period, with similar total cumulative sales, we see different results. Figure 38 show the sales progression of the two products next to each other. The product with *average* (on the left) is the product we examined earlier, while the product with *average (+)* (on the right) is the product without that warm-period in stage two.



(a) Average sales progression (b) Average (+) sales progression

Figure 38: Sales progression of two products with an average sales progression.

For the product with better sales in the first week, the heuristic chooses an earlier stage for the flexible shipment (compare Figures 37 and 39). Our S.O. shifts from choosing stage five or six to stage four or five in the low volume setting. However, both products have the same choice of the fifth stage in the high volume setting. This again shows the variability in the choice of a low volume setting, where selling approximately five to ten products around stage six contributes to the not so reliable behaviour.

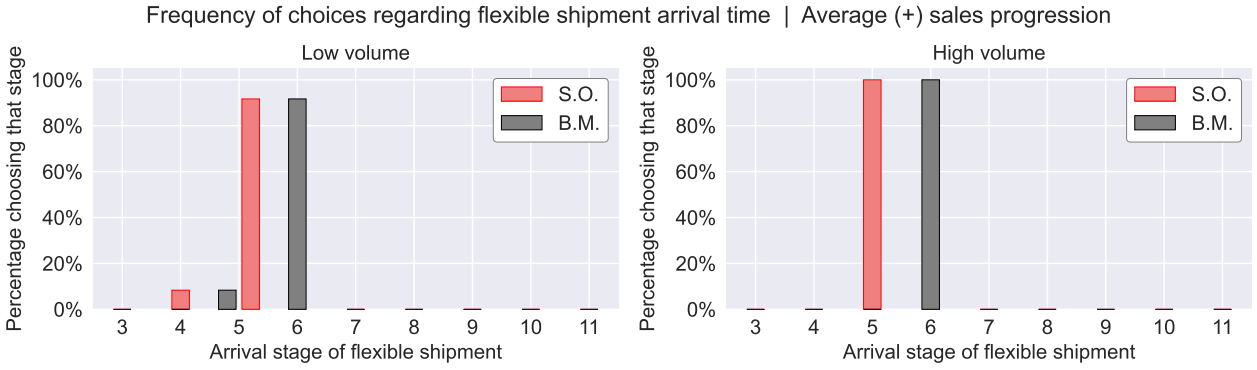


Figure 39: Percentage of times the heuristic chooses a certain arrival stage, split for the benchmark method (B.M.) and our stochastic optimisation (S.O.).

With the given examples, we can distinct between a low and high volume setting regarding reliability. The high volume setting delivers a reliable choice regarding the arrival stage of the flexible shipment,

a low volume setting has more uncertainty in it. Also, as expected, products with a weaker sales progression have a later stage chosen. Lastly, the B.M. seems to be as reliable as our S.O., but its choices are different. In Section 5.7.3, we will look into the consequences on its performance.

5.7.3 Performance

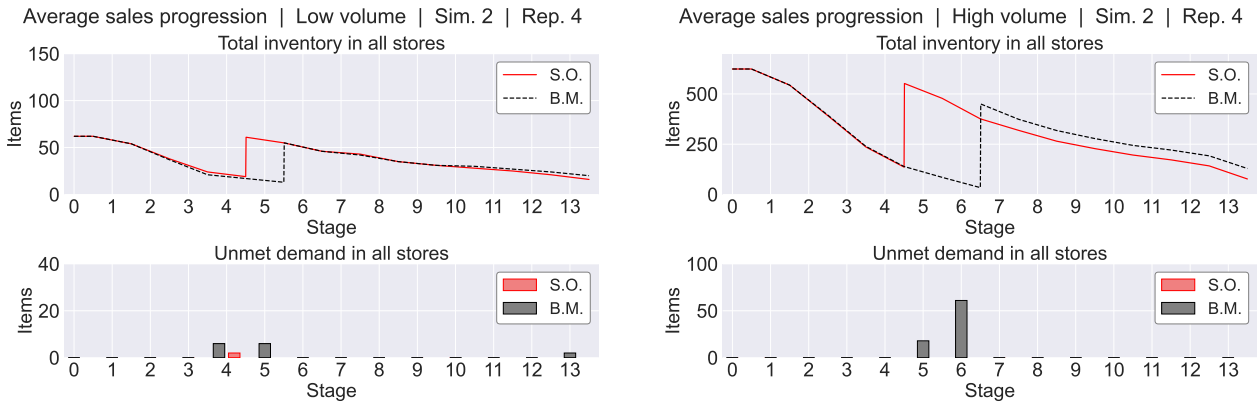
With the initial experiments on a split shipment, in Section 5.6, we established our S.O. is not more conservative with sending supplies to stores in case of a later shipment. In fact, the sales remained equal if the shipment would arrive in time. Therefore, the goal of this performance check is establishing whether this flexibility has an impact on fulfilling demand. To be even more critical, we check whether a later shipment arrival would cause unmet demand in the affected stages. If so, it means we have stretched the arrival of our flexible shipment as far as possible, corresponding to the benefits of relieving pressure on the distribution centre.

We examine this by looking at the total inventory level and total unmet demand in each stage. Since the benchmark generally chooses a later stage as arrival time, we use this as an indication to evaluate whether or not our S.O. chose the optimal arrival stage. For example, if the shipment using our S.O. arrives in stage four, and according to our B.M. (with a later shipment arrival time) unmet demand occurs in stage four, we know we chose the last possible stage for the shipment to arrive without sacrificing on the sales side.

First, we look at an exemplary result of a low and high volume setting for the product with an average sales progression (Figure 15b). In Figure 40, we display the performances regarding inventory levels and unfulfilled demand. The top subfigure shows the development of the total inventory level across stores *through* the stages, while the bottom subfigure shows the total unmet demand *in* each stage. Both figures separate on the methods, with in red our S.O., and in (dashed) black the B.M.

We can clearly see the different choice in optimal arrival time of the flexible shipment for the methods in both settings. In for example the high volume case, the flexible shipment arrives at the distribution centre two stages later for the B.M., compared to our S.O.. This causes unmet demand in stages five and six for the B.M., meaning our S.O. chose the optimal stage for the flexible shipment to arrive. Had the shipment arrived one stage later, the S.O. would have likely showed unmet demand in stage five as well.

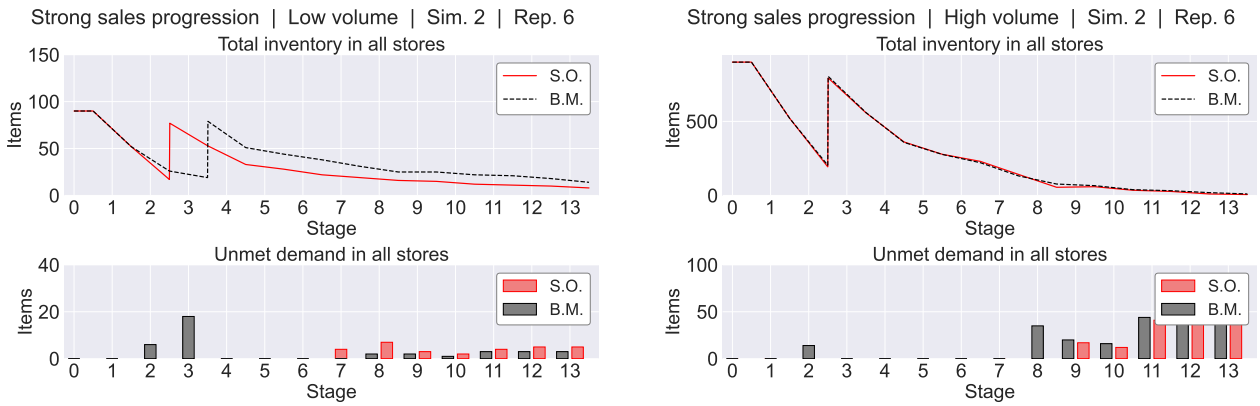
In the low volume setting, unmet demand occurs in stage four for both methods. While this is resolved one stage later in our S.O., the B.M. has to wait an additional stage on the flexible shipment causing more unmet demand. The S.O. also did not choose the optimal arrival stage, but the total unmet demand is limited, especially compared to the B.M.



(a) Low volume (b) High volume

Figure 40: Performance regarding inventory and unmet demand using a flexible shipment for a product with an average sales progression.

In the above example, we clearly saw a better performance from the S.O. This difference is also caused by the chosen optimal arrival stages of the flexible shipment. In Figure 41b, we see what could happen when both methods choose the same shipment stage. We consider both methods to have chosen the optimal stage, since it was the earliest possible after a decision following the events in stage two. However, the B.M. does show some unmet demand in stage two. We established earlier (Section 5.4.3) that our S.O. is better able to handle situations where the leftover inventory has become low and not all stores can be supplied, compared to the B.M. This is what we see in this result as well.



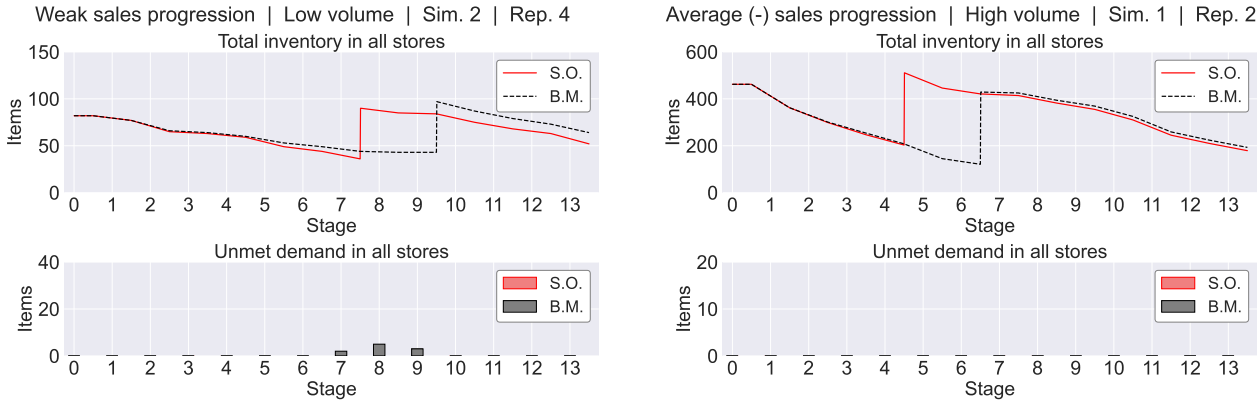
(a) Low volume (b) High volume

Figure 41: Performance regarding inventory and unmet demand using a flexible shipment for a product with a strong sales progression.

For the low volume case however, we again see the difference in choice of arrival stage. There, unmet demand occurs in stage three, a stage where the flexible shipment already would have arrived using our S.O. This again shows the superiority on the shipment decision of our S.O., which

can be seen as optimal. In later stages, more unmet demand occurs for our S.O. This can be easily explained by the significant lower leftover inventory in our S.O., which is caused by selling more items earlier on due to a better inventory allocation scheme of our S.O.

We can show more results on our S.O. being superior and not showing unmet demand where our benchmark does, but this is not always the case. To be more precise, since we built in the confidence interval bound by α into our heuristic (Algorithm 3), if sales progress in a less profitable manner, the flexible shipment could arrive too early. This is especially the case in high volume settings, while low volume settings are affected by the greater consequence of dividing the few leftover items to unfitting stores. We show examples from products with a different sales progression in Figure 42.



(a) Weak sales progression, low volume

(b) Average (-) sales progression, high volume

Figure 42: Performance regarding inventory and unmet demand using a flexible shipment for a product with a strong sales progression.

On the left hand side, the earlier showed product with a weak sales progression sees its flexible shipment arrive two stages later in the B.M. We therefore notice unmet demand for the B.M. in the three stages before the shipment arrival, likely due to the difficulty of dividing fifty items over twenty stores correctly. In for example the high volume setting of the same product, this unmet demand is not present.

We see the latter on the right hand side of the figure, in Figure 42b. We have not seen this item earlier, but similar as to in Figure 38, this product has an average sales progression which differs somewhat from the product with the average sales progression we evaluated so far. In fact, while the frequently evaluated product recovers from the warm-up period, this new product, with *average (-)* does not and has a bit lower sales compared to the earlier product. We show this figure because it can be perfectly compared with the result on the flexible shipment as that of Figure 40b.

We can compare them because both products have a similar sales level after two stages. This causes their flexible shipment arrival stage to be equal. However, the difference is that sales for the *average (-)* product do not progress in the same rate as for the earlier evaluated product. Therefore, we do not see any unmet demand for the product in Figure 42b, for both the S.O. and the B.M. This as opposed to the result in Figure 40b.

5.7.4 Conclusion

Already early on, we came to the conclusion that using a model to determine an optimal flexible shipment arrival stage is not successful, since this result is very sensitive to the used parameters. Furthermore, it has a very high computation time. Instead, we developed a heuristic based on the created scenario which is able to give good and fast estimates on this optimal stage.

For some products, in a low volume setting, this decision was split between two stages, mainly caused by the low number of sales in each stage. In a high volume setting, the heuristic does provide a consistent choice.

Using the heuristic and stochastic method, it is possible to apply a flexible shipment and maintain a similar sales level, which is not possible for all products according to our benchmark. However, while our stochastic method often provides a good decision, it could also be too conservative, resulting in a too early arrival of the flexible shipment.

6 Conclusion and discussion

At the start of the paper, we stated four main research questions to answer (Section 1.2). In the partial conclusions in the computational experiments section (Section 5), we already answered them for the better part. Here, we summarise those conclusions, comment on potential practical complications and discuss points up for further research.

The first two questions regarded the possibility of using uncertainty modelling to improve the sales value of an operations plan for a typical retailer, compared to a classical forecasting method. We largely performed those corresponding experiments in Sections 5.4 and 5.5. There, we concluded that we are able to use a stochastic programming approach by using scenario trees. The direct sales value, thus all items which sell while they are in stock, could be improved up to more than 10%. Especially for products encountering the problem of a shortage, thus where a decision on where to send an item should be made, this improvement was significant.

While our results are promising, we do have to denote some potential problems. We experimented using only one product at a time, where normally one would have to do this for all products. Also, the number of included stores was rather small compared to the reality. Especially if the problem becomes larger, because of these computation time increasing factors, we might need different methods to solve the mathematical model. Because we did use an optimality gap for our models and the results are promising, it seems that using an algorithm to solve the mathematical models would not hurt the results, but this is something for further research.

Also, the duration of the sales period is very relevant, especially with regards to the volume of items sold per stage. We showed that both in a low and high volume setting, our stochastic method produces an improvement in sales. However, especially in fashion retailer environments, products could move even slower per stage than in our low volume setting. Because the retailer can replenish everyday, each stage would then cover only one day of sales, instead of the week we used in our experiments. Although the resulting characteristics of the results show that it should also work in those settings, running entire simulations would be so time consuming it would not fit the scope of this paper.

The other two research questions comprised the inventory level at the distribution centre. For the first, we were interested in the impact on sales if not all items arrived at the distribution centre in time. In Section 5.6, we therefore went into depth regarding the behaviour of the supply plans in that case. Here, we showed that the stochastic method did not change its supply behaviour. This means that, if all items would arrive at such times that no stock out at the distribution centre occurs, a retailer is able to use such split shipments.

We explained benefits of using split shipments when describing the problem itself (Sections 2.3 and 2.4), which mainly come down to relieving and spreading pressure on the distribution centre. However, the paragraph above, we already denoted the biggest problem, *if* all items arrive *in time*. Especially if we choose this stage beforehand, we could choose a stage which would be too late.

Therefore, we took this one step further in Section 5.7. Here, we made the arrival time of such a split shipment flexible. As a recap, this entails that at the start of the sales period, the arrival stage of the split shipment is not known yet, but we decide on this arrival during this sales period. We showed that using such a flexible shipment made it able to maintain a similar sales value compared to when all items would have arrived at the start of the sales period.

Here, we have the same problem on the timing of the shipment, but we showed that for most products, we at least choose a stage which is on time, compared to using classical forecasting, where this idea is not applicable and would result in a suffering sales value for some products.

Even though this idea theoretically works and delivers positive results, the practical application side poses a different problem. For product a it might be best to ship before stage five, while this could be stage seven for product b. Since shipping from production sites could take months, all items might be sent in one shipment, after which all containers with products should be split into parts. This means it might be difficult to actually apply such an idea. However, solutions could be sought by for example grouping products with similar expected behaviour in sales (for example the same shirt with different size), such that it costs the least amount of time with grouping similar products when the optimal flexible shipment stage is determined. Further research is needed to find out up to which extent this idea of flexibility is applicable.

To conclude, we have shown that uncertainty models within a sales and operations process can be very useful. Using our method with scenario trees resulted in a more profitable operations planning. Therefore, while further research on the applicability is needed, uncertainty models definitely benefit the alignment of business segments of a retailer.

References

Baucke, R., Downward, A., & Zakeri, G. (2017). A deterministic algorithm for solving multistage stochastic programming problems. *Optimization Online*.

Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). *Robust Optimization* (Vol. 28). Princeton University Press.

Birge, J. R., & Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer Science & Business Media.

Bower, P. (2005). 12 most common threats to sales and operations planning process. *The Journal of Business Forecasting*, *24*(3), 4.

Dupačová, J., Consigli, G., & Wallace, S. W. (2000). Scenarios for multistage stochastic programs. *Annals of Operations Research*, *100*(1-4), 25–53.

Feng, Y., D'Amours, S., & Beauregard, R. (2008). The value of sales and operations planning in oriented strand board industry with make-to-order manufacturing system: Cross functional integration under deterministic demand and spot market recourse. *International Journal of Production Economics*, *115*(1), 189–209.

Growe-Kuska, N., Heitsch, H., & Romisch, W. (2003). Scenario reduction and scenario tree construction for power management problems. In *2003 IEEE Bologna Power Tech Conference Proceedings* (Vol. 3, pp. 7–pp).

Heitsch, H., & Römisch, W. (2003). Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, *24*(2-3), 187–206.

Høyland, K., & Wallace, S. W. (2001). Generating scenario trees for multistage decision problems. *Management Science*, *47*(2), 295–307.

Kazemi Zanjani, M., Nourelfath, M., & Ait-Kadi, D. (2010). A multi-stage stochastic programming approach for production planning with uncertainty in the quality of raw materials and demand. *International Journal of Production Research*, *48*(16), 4701–4723.

Kuhn, D., Wiesemann, W., & Georghiou, A. (2011). Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, *130*(1), 177–209.

Olhager, J., & Selldin, E. (2007). Manufacturing planning and control approaches: market alignment and performance. *International Journal of Production Research*, *45*(6), 1469–1484.

Rocha, P., & Kuhn, D. (2012). Multistage stochastic portfolio optimisation in deregulated electricity markets using linear decision rules. *European Journal of Operational Research*, *216*(2), 397–408.

Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, *28*(6-7), 971–983.

Sen, S. (2001). Stochastic programming: computational issues and challenges. *Encyclopedia of OR/MS*, 1–11.

Shapiro, A. (2001). Monte carlo simulation approach to stochastic programming. In *Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01ch37304)* (Vol. 1, pp. 428–431).

Shapiro, A., & Nemirovski, A. (2005). On complexity of stochastic programming problems. In *Continuous Optimization* (pp. 111–146). Springer.

Sodhi, M. M. S., & Tang, C. S. (2009). Modeling supply-chain planning under demand uncertainty using stochastic programming: A survey motivated by asset–liability management. *International Journal of Production Economics*, *121*(2), 728–738.

Sodhi, M. M. S., & Tang, C. S. (2011). Determining supply requirement in the sales-and-operations-planning (s&op) process under demand uncertainty: a stochastic programming formulation and a spreadsheet implementation. *Journal of the Operational Research Society*, *62*(3), 526–536.

Thomé, A. M. T., Scavarda, L. F., Fernandez, N. S., & Scavarda, A. J. (2012). Sales and operations planning: A research synthesis. *International Journal of Production Economics*, *138*(1), 1–13.

Tuomikangas, N., & Kaipia, R. (2014). A coordination framework for sales and operations planning (s&op): Synthesis from the literature. *International Journal of Production Economics*, *154*, 243–262.

Appendices

A Inventory Allocation Model - Notation

A.1 Sets

- $\omega \in \Omega = \{1, \ldots, W\} \rightarrow$ Scenarios
- $t \in \mathcal{T} = \{0, \ldots, T\} \rightarrow$ Stages
- $n \in \mathcal{N_S} = \{1, \ldots, N_S\} \rightarrow$ Locations (stores)
- $n \in \mathcal{N_D} = \{1, \ldots, N_D\} \rightarrow$ Locations (distribution centre)
- $\mathcal{N} = \mathcal{N_D} \cup \mathcal{N_S} \rightarrow$ Locations (all)

A.2 Parameters

- $a_{0,n} \rightarrow$ Initial inventory at location n
- $b_n \rightarrow$ Backorder costs for location n
- $d_{t,n}(\omega_t) \rightarrow$ Demand at location n in scenario w_t
- $h_{t,n} \rightarrow$ Holding costs in stage t for location n
- $m_{t,n} \rightarrow$ Minimum number of items to be present at the *start* of stage t
- $p(\omega_t) \rightarrow$ Occurrence probability of scenario ω_t
- $q_n \rightarrow$ Return rate of sold items for location n
- $\pi_{t,n} \rightarrow$ Sales price in stage t at location n

A.3 Variables

- $I_{t,n}(\omega_t) \rightarrow$ Inventory at the *end* of stage t at location n in scenario ω_t
- $r_{t,n}(\omega_t) \rightarrow$ Number of returned items from sales in stage $t-1$ at location n in scenario ω_t
- $u_{t,n}(\omega_t) \rightarrow$ Unmet demand in stage t at location n in scenario ω_t
- $v_{t,n}(\omega_t) \rightarrow$ Backordered demand in stage t at location n in scenario ω_t
- $y_{t,n}(\omega_t) \rightarrow$ Supplies arriving *before* the *start* of stage t at location n in scenario ω_t
- $z_{t,n}(\omega_t) \rightarrow$ Sales in stage t at location n in scenario ω_t

B Flexible (Split) Shipment Model - Additional Notation

B.1 Sets

- $d \in \mathcal{D} = \{1, \ldots, D\} \rightarrow$ Flexible split shipments
- $s \in \mathcal{S} = \{1, \ldots, S\} \rightarrow$ Transportation methods

B.2 Parameters

- $A_d \rightarrow$ Batch size of shipment d
- $c^s \rightarrow$ Transportation costs of method s
- $k_d \rightarrow$ Decision moment for shipment d, which lies between stages k_d and $k_d + 1$
- $L^s \rightarrow$ Lead time of transportation method s

B.3 Variables

- $x_d^s(\omega_t) \rightarrow$ Denotes whether transportation method s is chosen for shipment d

C Computational Experiments - Parameter Settings

C.1 Data Inclusion

- Number of branches in the scenario tree

Table 5: Example of number of branches and consequent scenarios in each stage in the scenario tree for the general results.

Stage	t	t+1	t+2	t+3	t+4
Branches	3	3	3	2	2
Scenarios	3	9	27	54	108

- Length of the sales period → We adapted the data to use a sales period of 13 stages.
- Data included at the root of the scenario tree → In general, we want to use all historical data which followed a similar path as to the product we are evaluating. Since we do not have that many data and we ran out of available data for some products, we widened the acceptable range by using all historical sales information with a similar sales progression at that stage, not necessarily also the stages before. We used a base interval of 7.5% at both sides, which we widened linearly with 1.5% with the absolute difference to *stage* 6.5. This accounts for the *eye*-pattern of the best and worst sales progressions as in Figure 15, such that we do not miss out on much data during the creation of the scenario tree.
- Data included in branches → After each branch, we would normally only keep the items with a similar sales progression, such that we only end up with few historical sales points for scenarios near the leaves of the scenario tree. Therefore, we expand the scenario tree to include sales data which followed a different sales progression, but are at the same level at that stage, similar as we do it for the root. Here, we use an interval of 5% at both sides.

C.2 Relevant Parameters

- Sales price

Table 6: Sales prices used for all results, including the salvage price in stage 14.

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Sales price	30	30	30	30	30	30	30	30	25	25	20	10	5	-5

- Return rate

Table 7: Return rates of each type of store, which lies in an interval dependent on how often a store sells in general compared to other stores of the same type.

Type	Online - DC	Online - partner	Offline
Minimum return rate	10%	5%	40%
Maximum return rate	10%	15%	60%

- Optimality gap

Table 8: Optimality gap used in the inventory allocation models, which is linearly scaled from 2% to 0% dependent on the stage.

Stage	1	2	3	4	5	6	7
Optimality gap	1.85%	1.69%	1.54%	1.38%	1.23%	1.08%	0.92%
		8	9	10	11	12	13
		0.77%	0.62%	0.46%	0.31%	0.15%	0%

- Holding costs

Table 9: Holding costs used with t as the first stage in the scenario tree, which is linearly scaled between 2 and 6 (or 6 and 2).

Stage	t	t+1	t+2	t+3	t+4
DC	6	5	4	3	2
Other	2	3	4	5	6

- Store presentation minimum requirement

Table 10: Store presentation minimum number of items to be available in each stage, used for all results.

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13
Minimum	1	1	1	-	-	-	-	-	-	-	-	-	-

- Backordering costs

Table 11: Backordering costs used for all results, which was only applied to the model and was set at 55% of the sales price in that stage.

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13
Sales price	30	30	30	30	30	30	30	30	25	25	20	10	5
Backorder costs	16.5	16.5	16.5	16.5	16.5	16.5	16.5	16.5	13.75	13.75	11	5.5	2.75