



Master's degree thesis

LOG950 Logistics

Automated and optimized planning for traders of farmed fish - effect of uncertainty

Jana Perednia

Number of pages including this page: 48

Molde, 23.05.2022



Mandatory statement

Each student is responsible for complying with rules and regulations that relate to examinations and to academic work in general. The purpose of the mandatory statement is to make students aware of their responsibility and the consequences of cheating. Failure to complete the statement does not excuse students from their responsibility.

<p>Please complete the mandatory statement by placing a mark <u>in each box</u> for statements 1-6 below.</p>		
1.	<p>I/we hereby declare that my/our paper/assignment is my/our own work, and that I/we have not used other sources or received other help than mentioned in the paper/assignment.</p>	<input checked="" type="checkbox"/>
2.	<p>I/we hereby declare that this paper</p> <ol style="list-style-type: none"> 1. Has not been used in any other exam at another department/university/university college 2. Is not referring to the work of others without acknowledgement 3. Is not referring to my/our previous work without acknowledgement 4. Has acknowledged all sources of literature in the text and in the list of references 5. Is not a copy, duplicate or transcript of other work 	<p>Mark each box:</p> <ol style="list-style-type: none"> 1. <input checked="" type="checkbox"/> 2. <input checked="" type="checkbox"/> 3. <input checked="" type="checkbox"/> 4. <input checked="" type="checkbox"/> 5. <input checked="" type="checkbox"/>
3.	<p>I am/we are aware that any breach of the above will be considered as cheating, and may result in annulment of the examination and exclusion from all universities and university colleges in Norway for up to one year, according to the Act relating to Norwegian Universities and University Colleges, section 4-7 and 4-8 and Examination regulations section 14 and 15.</p>	<input checked="" type="checkbox"/>
4.	<p>I am/we are aware that all papers/assignments may be checked for plagiarism by a software assisted plagiarism check</p>	<input checked="" type="checkbox"/>
5.	<p>I am/we are aware that Molde University College will handle all cases of suspected cheating according to prevailing guidelines.</p>	<input checked="" type="checkbox"/>
6.	<p>I/we are aware of the University College's rules and regulation for using sources</p>	<input checked="" type="checkbox"/>

Personal protection

Personal Data Act

Research projects that processes personal data according to Personal Data Act, should be notified to Data Protection Services (NSD) for consideration.

Have the research project been considered by NSD?

yes no

- If yes:

Reference number:

- If no:

I/we hereby declare that the thesis does not contain personal data according to Personal Data Act.:

Act on Medical and Health Research

If the research project is effected by the regulations decided in Act on Medical and Health Research (the Health Research Act), it must be approved in advance by the Regional Committee for Medical and Health Research Ethic (REK) in your region.

Has the research project been considered by REK?

yes no

- If yes:

Reference number:

Publication agreement

ECTS credits: 30

Supervisor: Johan Oppen

Agreement on electronic publication of master thesis

Author(s) have copyright to the thesis, including the exclusive right to publish the document (The Copyright Act §2).

All theses fulfilling the requirements will be registered and published in Brage HiM, with the approval of the author(s).

Theses with a confidentiality agreement will not be published.

I/we hereby give Molde University College the right to, free of charge, make the thesis available for electronic publication:

yes no

Is there an agreement of confidentiality?

yes no

(A supplementary confidentiality agreement must be filled in)

- If yes:

Can the thesis be online published when the period of confidentiality is expired?

yes no

Date: 23.05.2022

Preface

The above topic was suggested as a part of the project FishTraOpt, which is led by Maritech Systems AS, with Lerøy Seafood AS, Møreforsking AS, and Molde University College as partners. The main goal of FishTraOpt is to develop and test a prototype solver for the planning problems faced by traders of farmed fish. As academic support, multiple thesis topics were suggested including the one above specifying on uncertainty effect.

I am grateful to my supervisor, Associate Professor, PhD Johan Oppen, for cooperation and fish domain clarifications. Also, I thank my family, friends, and colleagues for support and motivation to do my best.

Jana Perednia

Contents

1. <i>Introduction</i>	6
2. <i>Problem description</i>	7
2.1 Order specifications	7
2.2 Packing	8
2.3 Transportation.....	8
2.4 Planning horizon.....	8
2.5 Pricing policy.....	9
2.6 Supply prognose	9
3. <i>Mathematical Model</i>	11
4. <i>Literature review</i>	16
4.1 Domain overview	16
4.2 Methodology.....	17
5. <i>Method(s) and data</i>	21
5.1 Data transformation	21
5.2 Methods	22
6. <i>Findings</i>	23
6.1 Exploratory data analysis.....	23
6.2 Deterministic optimization	27
6.3 Sensitivity Analysis	28
6.4 Robust worst-case oriented reformulation.....	30
7. <i>Conclusions</i>	31
8. <i>Prospects</i>	32
<i>References</i>	33
<i>Appendix</i>	35
A. <i>Example of input data</i>	35
B. <i>Data transformation</i>	36
C. <i>Data preprocess for Model with 1 day of planning horizon (1)</i>	38
D. <i>Data preprocess for Model for the week planning horizon (2) and Model with demand gap minimization (3)</i>	39
E. <i>Implementation of Model with 1 day of planning horizon (1)</i>	41
F. <i>Implementation of Model for the week planning horizon (2)</i>	41

<i>G. Implementation of Model with demand gap minimization (3)</i>	42
<i>H. Sensitivity analysis for Model with demand gap minimization (3)</i>	43
<i>I. The figure of fitted distributions for supply forecast error</i>	44

1. Introduction

*To be uncertain is to be uncomfortable,
but to be certain is to be ridiculous.
Chinese proverb*

One of the most important segments of the Norwegian economy is aquaculture and especially its branch practiced in seawater habitats and lagoons called mariculture. Cultivation of marine organisms in enclosed sections of the open ocean raises multiple operational research problems as well as allocation of farm's perishable supply within the market, which is the main focus of this work. The conservative way of planning this allocation manually is not only inefficient but can also entail mistakes both ending up in additional costs of money and trustworthiness for the supplier.

To narrow it down the aim of the work is to investigate the effect of uncertain supply forecast on the allocation success as well as to decide whether this uncertainty is negligible in the prospect of a real-time planning process. The approach is to make use of the problem's data provided by Lerøy Seafood AS with exploratory data analysis, robust and deterministic optimization and some machine learning techniques.

2. Problem description

As mentioned above, the main task of the project is to construct the solver engine for dealing with planning problems, namely automatic schedule generation for allocation of caught fish onto customers' orders. Primarily, this schedule should consider such factors as customer orders' specifications, location of producing plants, and strict storage requirements due to the perishable nature of the goods. The previous year's thesis work within the project was focused on experiments with different objective functions and analyzing the allocation. Due to confidentiality and antiplagiarism reasons, one cannot reuse their planning instrument and will create a new one from scratch. On the one hand, it is more time-consuming, on the other it will avoid repeating possible mistakes and will not be tightened in any way with decisions made from close but different research. My toolbox will serve my purpose of work – research haul forecast uncertainty. More about approaches and instruments in section 5. To get an overall picture, consider the specific parts of the problem and its descriptions. Multiple mathematical models can be found in the so-called Chapter *Mathematical Model* below.

2.1 Order specifications

Demand is formed from both regular contract customers and through a spot market ending up with orders expressed as quantities of different fish types. Many orders require one or more certificates or approvals, or that the health status of the fish is according to certain standards. The different batches of fish at the different plants may or may not meet these requirements, so one must check that all requirements for approvals and health status for all orders are met. Most contract orders are flexible in what sizes (weight classes) the customer should have, and we want to utilize this flexibility to a) deliver sizes where the supply is high, to avoid deficits in orders, and b) deliver sizes where the current spot price is low so that other sizes with a higher spot price can be sold to other customers. Customer orders have different priorities, some customers always get what they order, and others may have to take what they can get. This is not a problem with high supply, but often the

demand is higher than the supply, and then some customers will have their orders cut or canceled.

2.2 Packing

The unit used is a number of boxes, a box contains ca 20 kg. A pallet contains 27 boxes, and most orders come in multiples of 27. A truckload is 33 pallets or 891 boxes. Some orders are sent directly from the plant to the customer, and these orders should not be split, i.e., they should not be served from two or more different plants. Other orders are sent via Oslo, and many of these can be split. If an order is split, each plant contributing should provide a multiple of 27 boxes (whole pallets). Exceptions exist for smaller orders. One plant has a setup where different size (weight) classes cannot be mixed on the same pallet, while other plants can produce pallets with more than one size.

2.3 Transportation

Plants, where the fish is slaughtered and packed, are located both in the southern and northern parts of Norway, and the time needed for truck transportation to Oslo is one and two days, respectively. All orders have a given day for departure from Oslo, and one or more days are available for packing, depending on how fresh the customer requires the fish to be. If, for example, a departure from Oslo for a given order is Thursday, and there is one day for packing, this means that the fish must be packed at one of the plants in the south on Wednesday to be able to meet the customer requirements. With two days for packing, the order could be packed in the north on Tuesday, or in the south on Tuesday or Wednesday.

2.4 Planning horizon

Planning for the coming week starts on Wednesday. However, replanning is performed if new important information pieces arise such as new orders or updated forecasts.

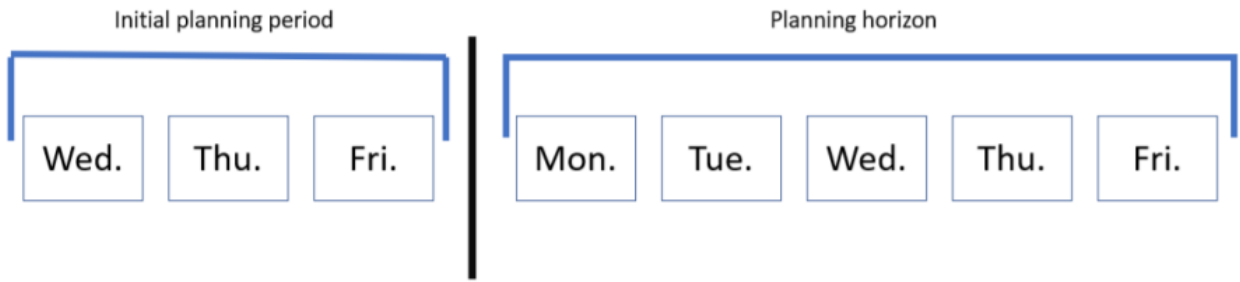


Figure 1 The initial planning period and the planning horizon. [1]

In addition, during the week replanning as displayed in [1] is also possible for the horizon excluding the current day, the plan for which is locked. As soon as the plan is ready, it is sent to the package coordinator.

2.5 Pricing policy

It is important to remember that for contract (regular) customer pricing is fixed, but the order might be flexible. For spot (irregular) customer price fluctuates depending on market indicators.

2.6 Supply prognose

At any moment in time, the company that trades farmed fish knows a forecast for the harvested fish in the coming week.

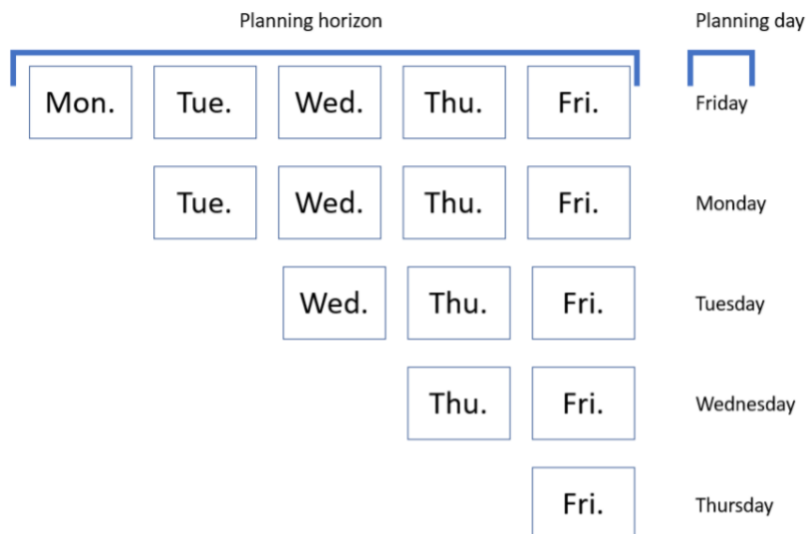


Figure 2 The remaining planning horizon for each planning day of the week [1]

The forecast includes parameters of harvested fish (size, quality, certification, health issues), and the quantity of each type. Generally, the more we observe, the better is the forecast, but still, it will not repeat the actual numbers of supply. There are two types of prognosis: long-term and short-term. The first one is made 2 weeks in advance or more by a biological planner. The short-term forecasts are composed gradually during packing. Usually, two-three prognosis iterations are performed for each day after a third and a half of what is expected to be packed in total during the day is processed. In our analysis, we will talk only about short-term prognosis due to a lack of information about biological forecasts from the company side.

3. Mathematical Model

In fact, each of the sections above is itself a separate optimization problem. It is possible to construct it all into one comprehensive problem, however, uncertainty research is reasonable to perform considering only the problem of allocating fish batches among customers since the only uncertain data reported is supply information.

We will deal with the assignment problem. The most basic example is the following *Model with 1 day of planning horizon (1)*

<p>Sets: C – set of customers G – set of fish groups (crossing sets of type, size, approvals)</p>
<p>Parameters: p_{gc} – price for one unit of fish group g for customer c, $p_{gc} \in \mathbb{R}^+$ $[l_{gc}, u_{gc}]$- bounds for amount of fish group g units assigned for customer c, $l_{gc}, u_{gc} \in \mathbb{Z}^+$ \hat{s}_g - supply of fish group g, $\hat{s}_g \in \mathbb{Z}^+$</p>
<p>Variables: X_{gc}- assigned value of units of fish group g for customer c, $X_{gc} \in \mathbb{Z}^+$</p>

$$\max_{X_{gc}} \sum_{g \in G, c \in C} p_{gc} X_{gc} \quad (1.1)$$

$$l_{gc} \leq X_{gc} \leq u_{gc}, \forall g \in G, \forall c \in C \quad (1.2)$$

$$0 \leq \sum_{c \in C} X_{gc} \leq \hat{s}_g, \forall g \in G \quad (1.3)$$

$$X_{gc} \in \mathbb{Z}^+, \forall g \in G, \forall c \in C \quad (1.4)$$

Objective (1.1) is to maximize the total price of sold fish. The order is considered fulfilled if it is possible to get X_{gc} within its bounds, i.e. satisfaction of (1.2). It is not possible to sell more fish than we caught, so the total number of sold fish for each group should not exceed our supply (1.3). As a result, model variables X_{gc} express the quantity bought (assigned) of each fish group g from a set of all groups G by (to) each customer c from customer set C . Its integrality is ensured by (1.4). In addition, as formulated above, \hat{s} is a

vector of fixed values representing supply, but in reality, it is only a forecasted approximation of the real supply.

Important to note that we are dealing with a mixed-integer programming problem (MIP) since our variables may have only integer values or to be specific mixed-integer linear programming problem (MILP) since constraints' and objective's functions are linear. However, we usually have inventory from past days' harvest and to be sustainable it's needed to optimize an assignment of these leftovers. To address this, please consider the following model with an assumption of having zero holding costs:

Model for the week planning horizon (2)

<p>Sets:</p> <p>G – set of fish groups</p> <p>H – set of harvesting dates (combined with region South / North)</p> <p>S – set of shipping dates (from Oslo)</p> <p>C – set of customers</p>
<p>Parameters:</p> <p>p_{ghsc} – price for one unit of fish group g harvested at day h with shipping date s for customer c, $p_{ghsc} \in \mathbb{R}^+$</p> <p>$\sigma_h$ - vector of penalties of length H weighting fish leftovers, $\sigma_h \in \mathbb{R}^+$</p> <p>$[l_{gsc}, u_{gsc}]$- bounds for value of fish group g units assigned with shipping date s for customer c, $l_{gsc}, u_{gsc} \in \mathbb{Z}^+$</p> <p>$f_{ghsc} = \begin{cases} 1, & \text{fish of group } g \text{ harvested at day } h \text{ sent at day } s \text{ is accepted by the customer } c \\ 0, & \text{otherwise} \end{cases}$</p> <p>$\hat{s}_{gh}$- supply forecast of fish group g when harvesting at date h, $\hat{s}_{gh} \in \mathbb{Z}^+$</p>
<p>Variables:</p> <p>X_{ghsc}- assigned value of fish units of group g collected at harvesting date h with shipping date s for customer c, $X_{ghsc} \in \mathbb{Z}^+$</p> <p>$\Delta_{gh}$- leftover units of fish group g collected at harvesting date h after assignment for planning horizon, $\Delta_{gh} \in \mathbb{Z}^+$</p>

$$\max_{X_{ghsc}} \left(\sum_{g,h,s,c} p_{gsc} X_{ghsc} \right) - \left(\sum_h \sigma_h \sum_g \Delta_{gh} \right) \quad (2.1)$$

$$l_{gpc} \leq \sum_h X_{ghsc} \leq u_{gsc}, \forall g \in G, s \in S, c \in C \quad (2.2)$$

$$M * f_{ghsc} \geq X_{ghsc} \geq 0, \forall g \in G, h \in H, s \in S, c \in C \quad (2.3)$$

$$\sum_{s,c} X_{ghsc} \leq \hat{s}_{gh}, \forall g \in G, h \in H \quad (2.4)$$

$$\Delta_{gh} = \hat{s}_{gh} - \sum_{s,c} X_{ghsc}, \forall g \in G, h \in H \quad (2.5)$$

$$X_{ghsc} \in \mathbb{Z}^+, \forall g \in G, h \in H, s \in S, c \in C \quad (2.6)$$

$$\Delta_{gh} \in \mathbb{Z}^+, \forall g \in G, h \in H \quad (2.7)$$

Our goal (2.1) here is to maximize the total price of sold fish and minimize the leftovers. The σ vector was introduced to emphasize minimizing the oldest leftovers the more since we still have a chance to sell the most recent ones during the next planning horizon. As described above in section Planning horizon, usually we start planning on Wednesday for all next week (Monday-Friday), so the vector σ will look somewhat like this:

penalty\harvest	Wed h=0	Thu h=1	Fri h=2	Sat h=3	Sun h=4	Mon h=5	Tu h=6	Wed h=7	Thu h=8	Fri h=9
$\sigma =$	10	9	8	7	6	5	4	3	2	1

For each order we have the `num_days_for_packing` input parameter telling the maximum number of days accepted by the customer for storing fish. Based on that and shipping date requirements, we may calculate freshness parameter f_{ghsc} in advance representing if the period of storing a particular bunch of fish is acceptable for the customer and ensure fulfilling at least the lowest demand l_{gsc} with fresh enough fish in constraint (2.2). On the other hand, we must make sure we assign fish only when f_{ghpc} is non-zero (approved by customer), so we add “big-M constraint” (2.3). Apparently, we cannot assign more fish than we have in supply, as illustrated in (2.4). Constraint (2.5) defines leftover inventory for each fish group and harvesting date by substituting from the supply the sum of assigned fish among all customers and packing dates. Integrality of assignment variables X_{ghsc} and delta variables Δ_{gh} is ensured by (2.6) and (2.7) respectively. This is also a MILP problem. However, as mentioned in section , our automated planning tool should be prepared for situations when demand exceeds supply, so “hard” constraint (2.2) may make the problem infeasible. The solution is to make it “soft” by moving it into objective function by minimizing the gap between lower bound of demand and actually assigned units of fish.

Please, see the next model:

Model with demand gap minimization (3)

<p>Sets:</p> <p>G – set of fish groups</p> <p>H – set of harvesting dates (combined with region South / North)</p> <p>S – set of shipping dates</p> <p>C – set of customers</p>
<p>Parameters:</p> <p>σ_h - vector of penalties of length H weighting fish leftovers, $\sigma_h \in \mathbb{R}^+$</p> <p>$[l_{gsc}, u_{gsc}]$- bounds for value of fish group g units assigned with shipping date s for customer c, $l_{gsc}, u_{gsc} \in \mathbb{Z}^+$</p> <p>$f_{ghsc} = \begin{cases} 1, \text{ fish of group } g \text{ harvested at day } h \text{ sent at day } s \text{ is accepted by the customer } c \\ 0, \text{ otherwise} \end{cases}$</p> <p>$\hat{s}_{gh}$- supply forecast of fish group g when harvesting at date h, $\hat{s}_{gh} \in \mathbb{Z}^+$</p>
<p>Variables:</p> <p>X_{ghsc}- assigned value of fish units of group g collected at harvesting date h with shipping date s for customer c, $X_{ghsc} \in \mathbb{Z}^+$</p> <p>$\Delta_{gh}$- leftover units of fish group g collected at harvesting date h after assignment for planning horizon, $\Delta_{gh} \in \mathbb{Z}^+$</p> <p>$d_{gsc}$- units of fish of unsatisfied demand group g with shipping date s for customer c, $d_{gsc} \in \mathbb{Z}^+$</p>

$$\min_{X_{ghsc}} \left(\sum_{g,s,c} d_{gsc} \right) + \left(\sum_h \sigma_h \sum_g \Delta_{gh} \right) \quad (3.1)$$

$$l_{gpc} \leq \sum_h X_{ghsc} \leq u_{gsc}, \forall g \in G, s \in S, c \in C \quad (3.2)$$

$$M * f_{ghsc} \geq X_{ghsc} \geq 0, \forall g \in G, h \in H, s \in S, c \in C \quad (3.3)$$

$$\sum_{s,c} X_{ghsc} \leq \hat{s}_{gh}, \forall g \in G, h \in H \quad (3.4)$$

$$\Delta_{gh} = \hat{s}_{gh} - \sum_{s,c} X_{ghsc}, \forall g \in G, h \in H \quad (3.5)$$

$$d_{gsc} = l_{gsc} - \sum_h X_{ghsc}, \forall g \in G, s \in S, c \in C \quad (3.6)$$

$$X_{ghsc} \in \mathbb{Z}^+, \forall g \in G, h \in H, s \in S, c \in C \quad (3.7)$$

$$\Delta_{gh} \in \mathbb{Z}^+, \forall g \in G, h \in H \quad (3.8)$$

$$d_{gsc} \in \mathbb{Z}^+, \forall g \in G, s \in S, c \in C \quad (3.9)$$

This model differs from the previous one by defining a new slack variable d_{gsc} via constraint (3.6) which answers “how much we are not satisfying the demand”. Integrality is controlled by (3.9). The objective (3.1.) is to fulfill the demand to the extent it is possible and sell maximum fish in general. The current model is also classified as MILP problem. In the section Appendix one may find the code of implementations and results discussions in the section *Deterministic optimization*.

4. Literature review

4.1 Domain overview

Let us start the section with an overview of [1] - past thesis work within the same project FishTraOpt. Besides its main focus was handling multiple objectives in optimization, it includes a profound introduction to the problem's environment and its specifications. First, let me emphasize the crucial role of mariculture in the Norwegian economy. During the last decade, industry developed dramatically, creating a lot of workplaces, and raising export significantly, see **Figure 3** Sales of slaughtered food fish in Norway, from 1986-2018. Amount in 1000 tons, value in billion 2019-NOK. Statistics Norway, in [1]. So even a small percentage efficiency improvement of farm operations might make a sufficient win in absolute money value.

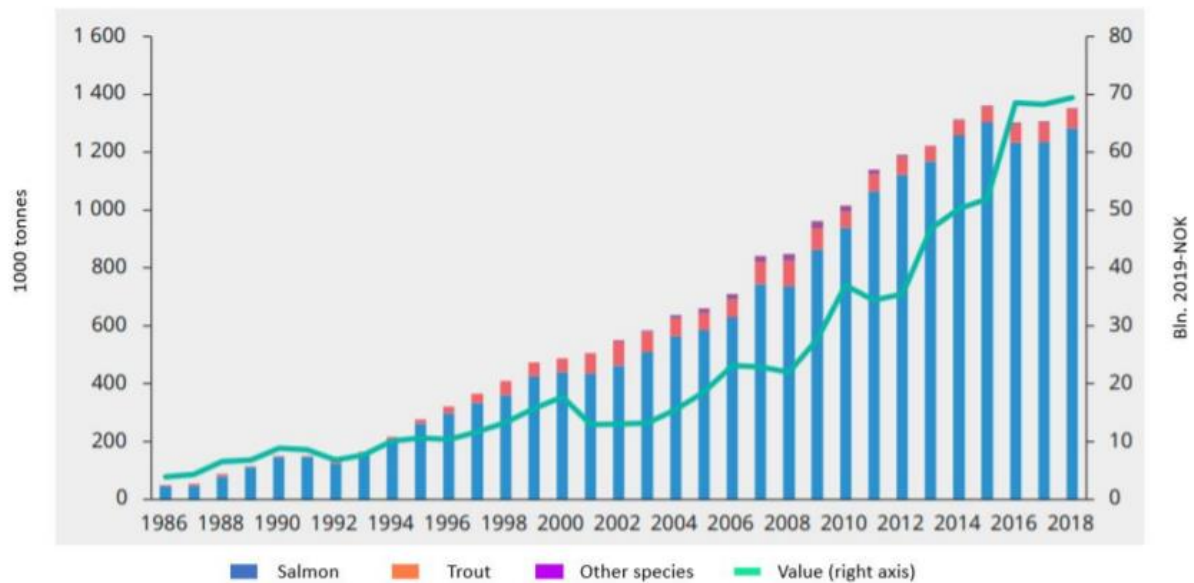


Figure 3 Sales of slaughtered food fish in Norway, from 1986-2018. Amount in 1000 tons, value in billion 2019-NOK. Statistics Norway, in [1]

In addition, let me show you an illustration of the physical flow of the supply chain described in the previous section, see **Figure 4**. Since fish freshness is critical, we face

perishable supply chains. Obviously, this freshness requirement makes the chain more complicated and therefore more susceptible to uncertainties.

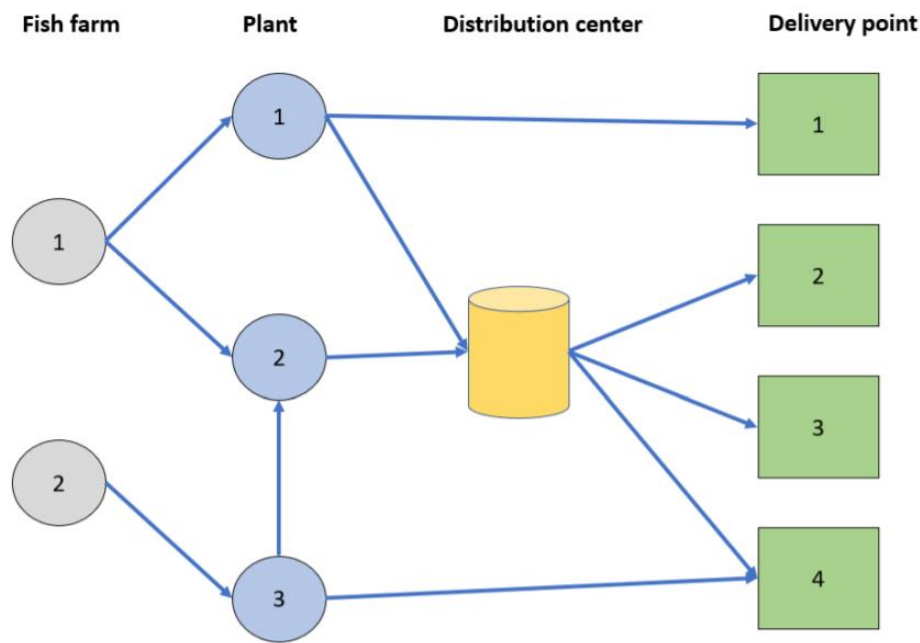


Figure 4
Simplified
illustration of
the physical
flow [1]

4.2 Methodology

As mentioned in the introductory section, Robust optimization (RO) is one of the approaches to examine when dealing with uncertainty. In fundamental work representing the approach of [2] we find the definition of a term *uncertain LP problem*, which is a collection of LP programs of a common structure

$$\{\min_x \{c^T x : Ax \leq b\} : (c, A, B) \in U\}$$

with the data (c, A, b) varying in a given uncertainty set U . In our case, as stated the model (2) includes approximated vector \hat{s} of the real supply s . To classify our problem as uncertain we need to define uncertainty set U , namely the subset of right-hand side B possible vectors since the rest objective vector c and constraint matrix A are assumed to be known exactly. Enumerating all potential values of fish supply, it will be $[0, u] \times |G|$, which is the value from 0 to some maximum upper supply bound u for each of the groups ($|G|$ - the power of groups set), so it results in solving $(u + 1)^{|G|}$ problems and it seems to require a lot of computational resource for such approach. In addition, we should note that RO treats all constraints as “hard“, i.e. no violation tolerance is accepted, but from our problem description, we know that for example order fulfillment might be not so strict.

Another widely used approach for treating the problem with uncertain data is Stochastic Optimization (SO) or Stochastic Programming (SP). To the point, [2] and [3] emphasize that within SO uncertain numerical data is assumed to be random. In the basic SO version, we assume to know the data distribution, which seems to be more applicable to our problem. Indeed, for every fish group (type, size, approvals) there are some bounds for possible supply and some distribution may be derived from historical data. A typical representative of SO is the chance-constrained problem defined by [2] as

$$\min_{x,t} \{t: \text{Prob}_{(c,A,b) \sim P} \{c^T x \leq t \ \& \ Ax \leq b\} \geq 1 - \epsilon\},$$

where $\epsilon \ll 1$ is a given tolerance and P is the distribution of the data (c, A, b) . If probabilistic-nature guarantees given by chance constraints are recognized, then the decision-maker is to decide on an appropriate safety margin ϵ and therefore the restriction of the feasible region. As [4] notes, chance constraints are making the problem non-convex. In particular, [5] describes two approaches for solving a chance-constrained problem. A straightforward one is to solve exactly requiring the computation of gradients, second derivatives, and information in the covariance matrices of the uncertainty factors. On the other hand, by bounding the constraint it might be possible to find a good convex approximation of the chance constraint. Convexity makes a problem a way easier to solve since every local optimum is a global one [6]. However, the convexity issue for a chance-constrained problem is widely known for a considerable difficulty [7]. Besides, it might be reasonable to combine RO with SO, as emphasized in [2] on page 15, “Stochastic and Robust Optimization are complementary approaches for handling data uncertainty in Optimization, each having its own advantages and drawbacks”.

Since our deterministic models are MILP problems, we are empowered by many existing impressive algorithms to solve it. Often an analysis of the effect on the optimal solution of changes in the input data, can be more important in practice than finding the optimal solution, claims the father of the Simplex Method George Dantzig in [8]. The described above type of analysis refers to the term Sensitivity Analysis. In other words, this is an investigation of the solution to a dual formulation – vector π of shadow prices. The shadow price of constraint i is defined in [8] to be the rate of the change in the objective function as

a result of a change in the value of b_i , the righthand side of constraint i . As a result, if the problem below has a non-degenerate optimal solution x^0

$$\{\max_x \{c^T x : Ax = b, x \geq 0\}\}$$

then dual solution or shadow prices are expressed as following

$$\pi^0 = \frac{\partial(c^T x_b^0)}{\partial b_i}, i = \overline{1, m}$$

Respectively, if $\pi^0 > 0$ then increasing the amount of resource b_i leads to an increase in profits and the more efficient the greater the resource, else if $\pi^0 < 0$ a decrease in resource b_i leads to an increase in maximum profit [9]. In other words, according to Strong Duality Theorem, if primal and dual optimal solutions exist primal objective equals dual one [8]:

$$z^0 = c^T x^0 = b^T \pi^0$$

Because our only uncertainty is in the right-hand side it is practical to measure the maximum effect of supply perturbations on the objective value gained via Sensitivity Analysis. MILP problems are usually solved via Linear Programming wrapped into a Branch-and-Bound algorithm, so there are actually no well-defined shadow prices for the original MILP [10]. Nevertheless, it is always possible to work with MILP's relaxation providing clear dual variables, but then one should note that it is possible for a variable to have a largely reduced cost in the relaxed model even though you can easily change that variable in the original MIP without any change in the objective value [10].

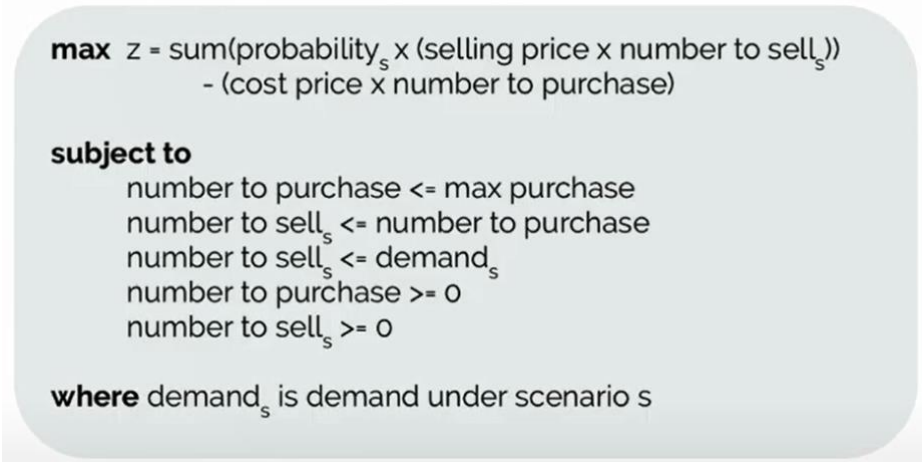
Practical examples of handling or ignoring uncertainty within one problem together with the theoretical background are given in [2]. There we can find a comparison of robust and 'classical optimization' decisions and see how crucial sometimes might be to take uncertainty into account. For instance, in medical production example, it is to decide an amount of needed raw materials for producing the most profitable medicines. However, percentages of chemicals in the raw materials used for making components of the different drugs vary. In an experiment of generating chemical concentration variation of 2-5% within uniform distribution it was discovered that in at least 25% of the experiments, it is lost at least 15% of the profits associated with production (and often 20% or more) if classical deterministic optimization is performed. Nevertheless, robust worst-case-oriented reformulation of the problem, namely adding (substituting) maximum variation values into

(from) the constraint matrix, allows getting a solution whose degradation is at most 6% over the nominal.

$$\begin{aligned} & \text{minimize } c^T x \\ & \text{subject to } (A + \Delta)x \leq b \end{aligned}$$

Figure 5, Robust worst-case-oriented reformulation of our problem to address the uncertainty [2]

Fortunately, we operate with discrete values and my supply forecast may vary within a very limited number of values. A demonstrative example of handling the uncertainty of inventory and demand is inspired by [11]. Basically, the author has a set of possible inventory (demand) values with their probabilities each representing a given scenario (uncertainty set) and it is to optimize an expected profit.


$$\begin{aligned} \mathbf{max} \quad & z = \text{sum}(\text{probability}_s \times (\text{selling price} \times \text{number to sell}_s)) \\ & - (\text{cost price} \times \text{number to purchase}) \end{aligned}$$

subject to

$$\begin{aligned} & \text{number to purchase} \leq \text{max purchase} \\ & \text{number to sell}_s \leq \text{number to purchase} \\ & \text{number to sell}_s \leq \text{demand}_s \\ & \text{number to purchase} \geq 0 \\ & \text{number to sell}_s \geq 0 \end{aligned}$$

where demand_s is demand under scenario s

Figure 6 Deterministic equivalent formulation for the stochastic program [11]

However, introducing sets of possible values for each stochastic variable means an increase in problem sizes up to the product of set sizes.

5. Method(s) and data

As was mentioned above, to answer the main question of current research, i.e. “Is uncertainty negligible in prospective of the real-time planning process? What are the consequences of including it into the model?”, we need to experiment with different methods involving uncertainty.

5.1 Data transformation

Note that unlike in past thesis work on this project, real data from Lerøy Seafood AS will be used instead of fictional generated instances. Since the data received is in the XML format (see section Example of input data), it is needed to transform data into a tabular format which is more user-friendly in terms of statistical analysis and launching optimization. Please, see UML-diagram of available data.

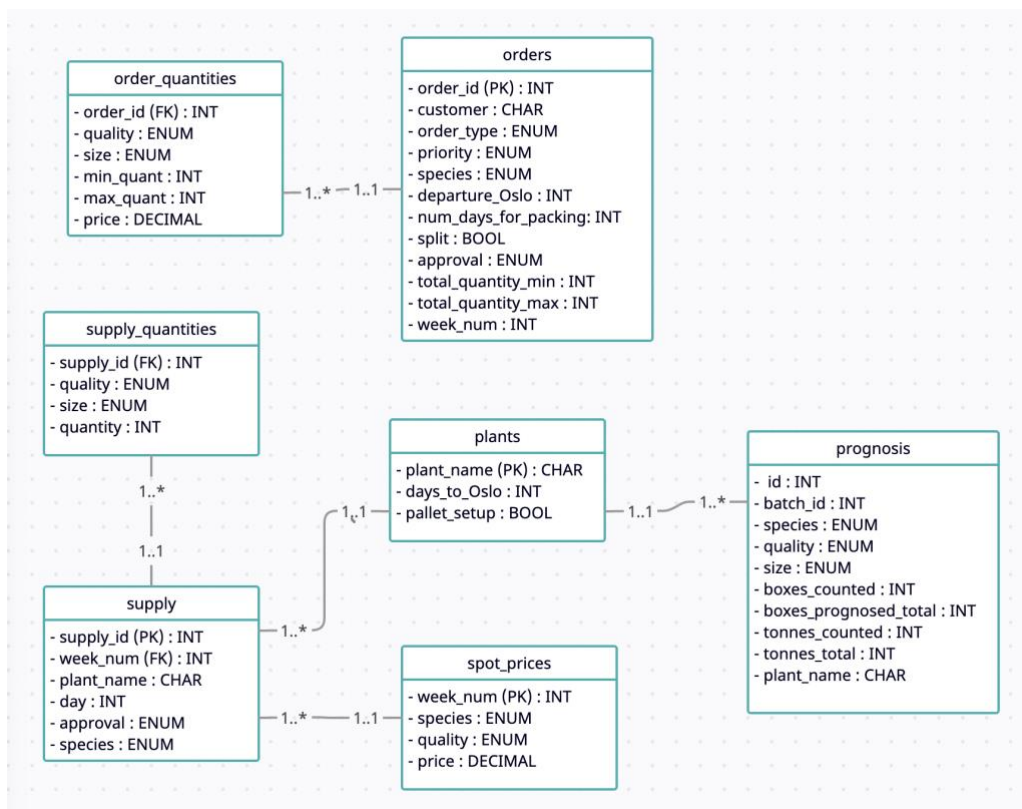


Figure 7 UML-diagram of Lerøy Seafood AS research data

One can find the code for transformation in the section Data transformation.

Please note, that the prognosis data received did not include the rest of the tables, so it was not possible to properly test out the effect of prognosis incorrectness on demand satisfaction.

5.2 Methods

First, exploratory data analysis will be performed to acknowledge previously unnoticed properties of data and problem. Afterwards the margin parameters for worst-case robust optimization will be chosen. Eventually the mathematical model of the problem will be optimized using safe version of uncertain supply vector.

6. Findings

This chapter presents the main findings, insights, and discussions when working with the problem. Section 6.1 reveals a description of dataset characteristics that might be useful for the overall understanding of the problem and at further stages. Implementational details of the optimization models considered above can be found in Section 6.2 and their sensitivity discussion was put into Section 6.3.

6.1 Exploratory data analysis

Let us start by exploring the patterns of supply throughout the week for different fish groups. **Figure 8** illustrates that despite the various supply scales, the week starts with high supply and finishes with low for all groups.

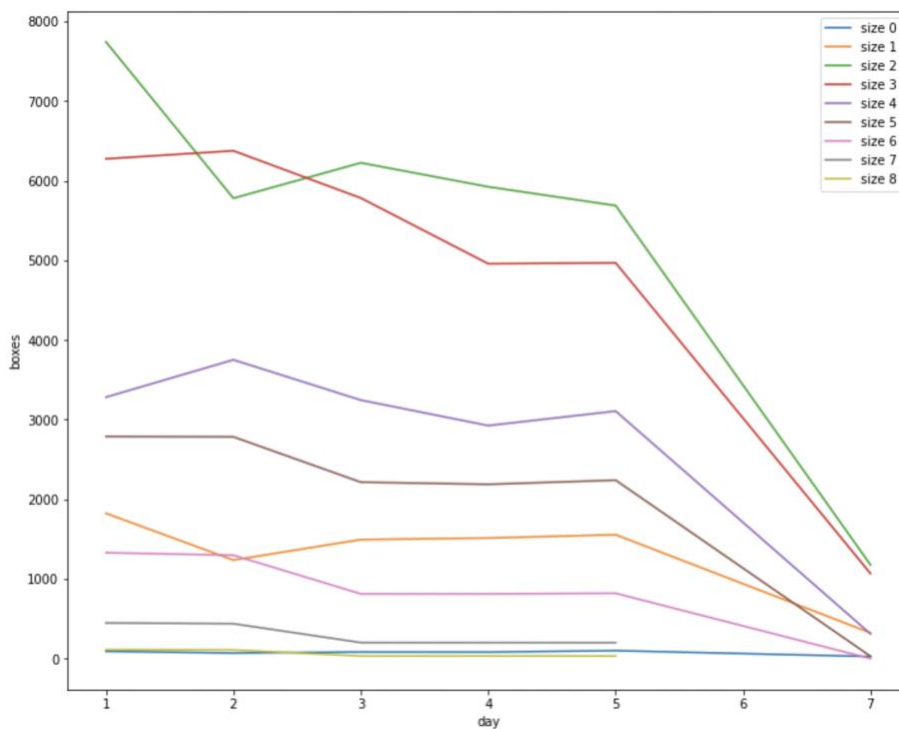


Figure 8 Supply for different salmon sizes for a week

Comparing the last figure with *Figure 9*, we observe that the beginning and end of the week are low in terms of demand, but in the middle the behavior might differ.

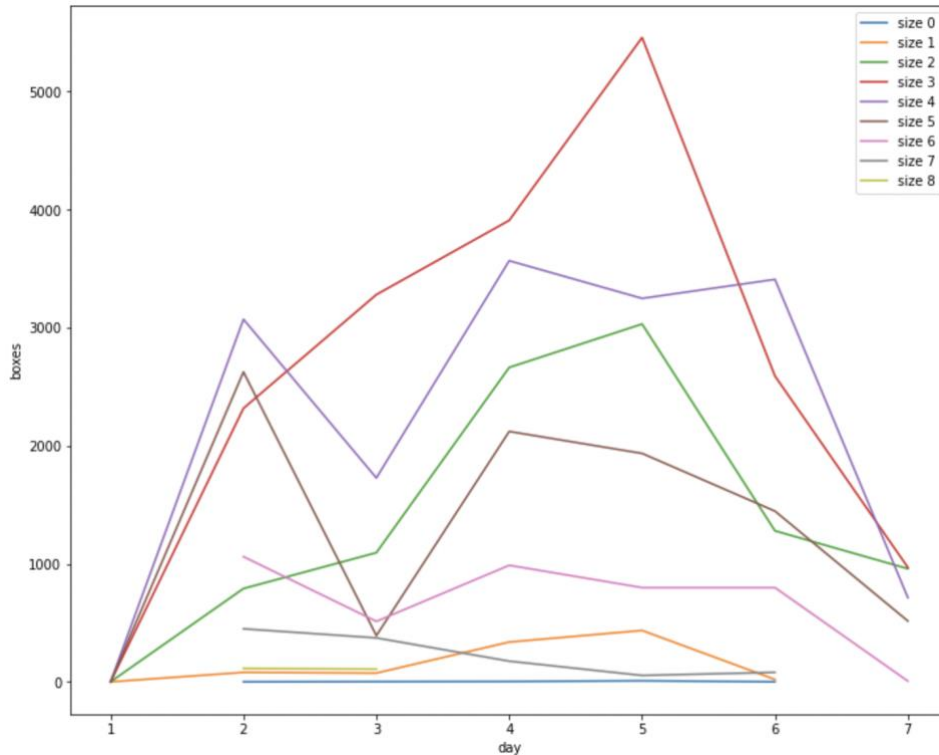


Figure 9 Demand for different salmon sizes for a week

From the two above figures, we may already identify the probable cases of unsatisfied demand and too high supply. For instance, the green line representing size 2 reaches almost 6000 boxes for 5 out of 7 days of the week on the supply plot, but the same line on the demand plot barely touches the mark of 3000 boxes only on one day, so it must be not possible to allocate all supply of this size. Please, see more comparisons below.

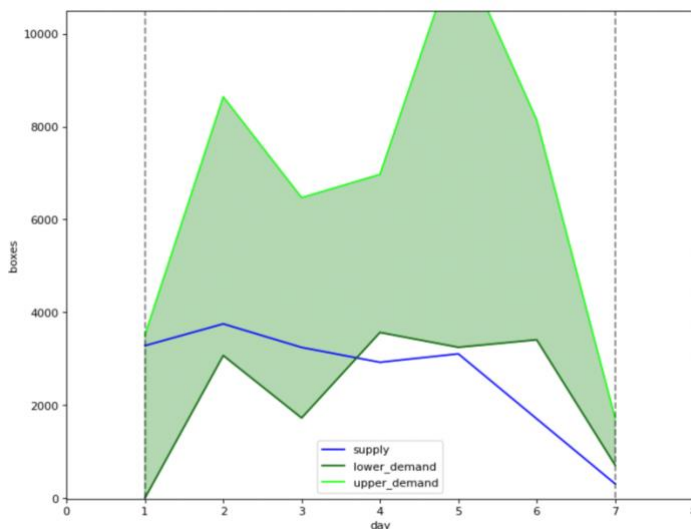


Figure 10 Example of hardly satisfied demand

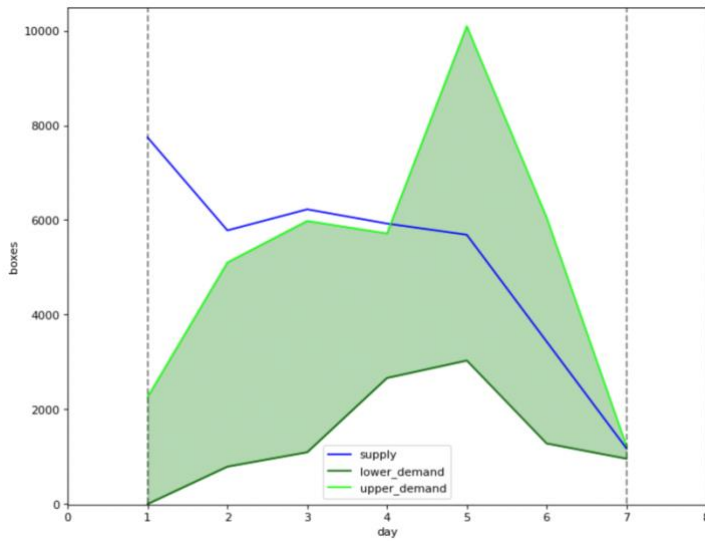


Figure 11 Example of unallocated supply

It is important to mention that either the prognosis data has poor quality or the forecasts themselves have questionable accuracy. For example, for one fish batch (plant, day, species, quality, size) after 15.8% of the total daily volume was packed it was estimated the batch to be 193 boxes by the end of the day. After 39.3% was packed the estimation changed to 689 boxes and eventually it appeared to be 1353 boxes! Thus, every iteration multiplied the previous guess by 2 or 3 times, which is probably acceptable for small numbers but not for hundreds. *Figure 12* represents the histogram of raw forecast errors percentage.

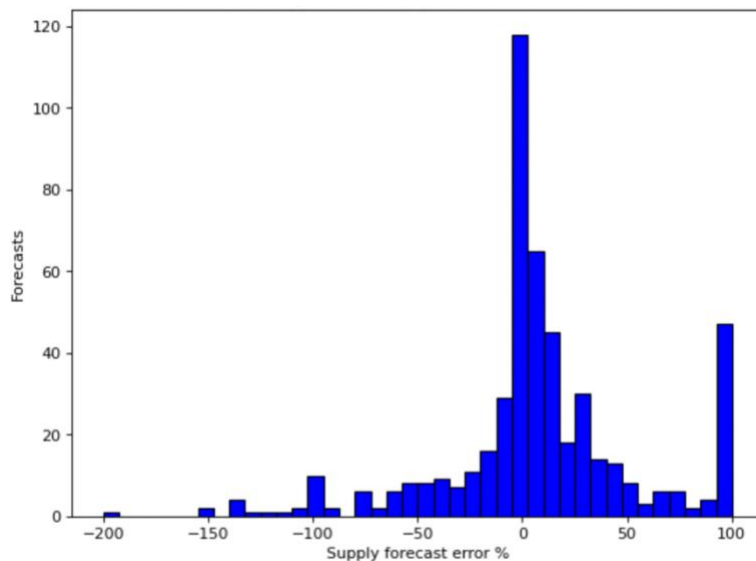


Figure 12 Histogram of supply forecast error %

All the fish groups are plotted together due to the assumption that the forecast is generated in the same way for all groups and taking the percentage relative to the average supply is

aimed to bring errors to a comparable scale. Let us try to identify the distribution of the forecast error percentage. Thanks to the library *Fitter* [12] [13] in Python it is very easy to do. In order to be more precise in exploring the closest distribution, it is reasonable first to remove outliers from the dataset. It is always tricky to decide the cutoff points and there are no general rules for defining them partially because of the various possible origin of outliers [14]. Since our data has only one dimension, we make a desperate decision to take the simplest and probably most popular Z-score method [15], where z is defined as

$$z = \frac{x - \mu}{\sigma}$$

Usually, if Z-score for an observation x is greater than 3 we consider x as an outlier. After removing outliers, the process of scanning the most common distributions was launched. Please, see *The figure of fitted distributions for supply forecast error percentage* produced with *Fitter* in the Appendix. As a result, the best-suited distribution for our data sample turned out to be the Cauchy distribution $\sim \text{Cauchy}(x_0, \gamma)$, where location parameter $x_0 = 4.02$, scale parameter $\gamma = 13.42$. Cauchy distribution may look similar to a normal distribution, but it has heavier tails, undefined mean and standard deviation so that an estimate of mean on a large data set doesn't add any accuracy in comparison with an estimation derived from a single sample [16].

For every fish batch we have multiple prognosis taken as the packing fulfils. *Figure 13* illustrates dependency between forecast error and percentage of packed volume. One may observe that over the packing process maximum positive error decreases, i.e. we underestimate less. On the other hand, overestimation doesn't change on average (red line) as more volume is packed.

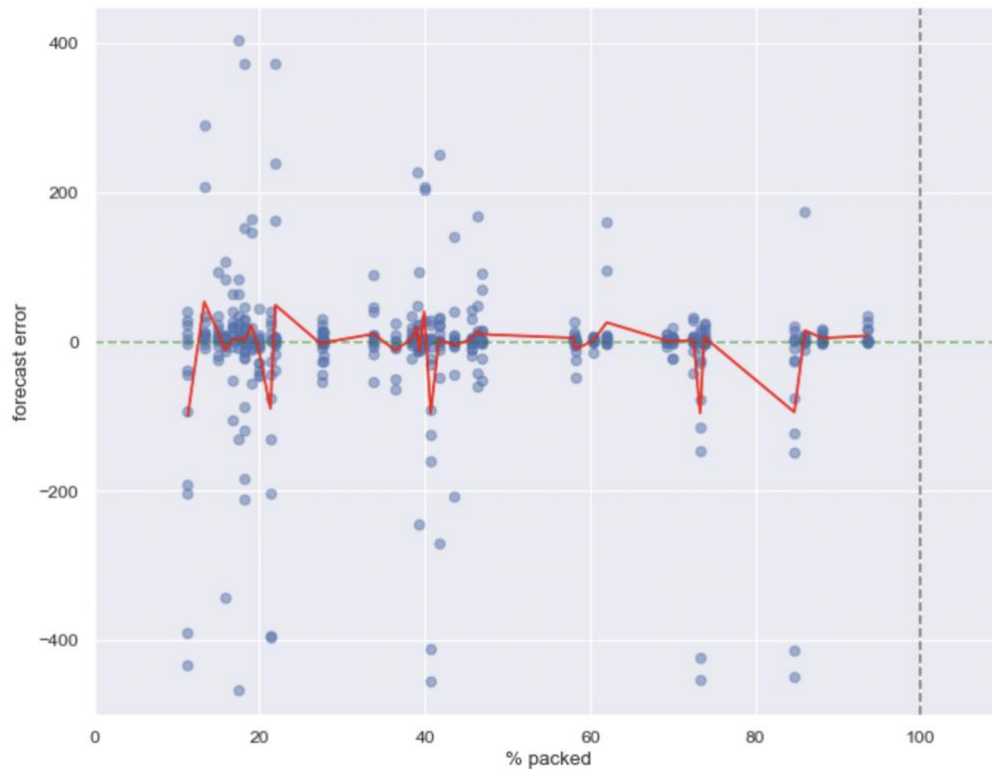


Figure 13 Forecast error with average depending on percentage packed

6.2 Deterministic optimization

Although the focus of this work is an effect of uncertainty, one will find below a profound discussion about the best deterministic model suiting our goal. As assessed in the chapter *Literature review* all highlighted ways of approaching uncertain data start from editing a deterministic model.

The first part of a computation study is solving described above in Chapter *Mathematical Model* deterministic models with commercial MIP and LP solver Gurobi [16]. It was chosen as the TOP-3 Simplex LP solver by the results of independent Hans Mittelmann's benchmarks [17]. The solver is equipped with a set of presolve techniques which allow solving even large and sophisticated problems relatively quickly. The code of models' implementations might be found in *Appendix*. The software and hardware used to implement and run the models are listed below:

Processor	Apple M1 Pro
Number of processors	1
Cores per processor	10

Memory (RAM)	32 GB
Operating system	macOS Monterey
Gurobi version	9.5.1
Python version	3.9.12

Table 1 Software and hardware used for experiments

Thus, the optimization launch of the Model with 1 day of planning horizon (1) with real data turned out to be infeasible. Specifically, the lower bound constraints (1.2) stayed unsatisfied, and therefore it was decided to remove them so that each assignment variable has a default zero lower bound. Then on average, 95.5% of supply was allocated but only 70.6% of demand was satisfied when the model was tested sequentially for each day of the week so it was not possible to use the inventory, and possibly some supply batches could be used multiple times.

Second, testing the Model for the week planning horizon (2) for the whole week resulted in 97.3% allocated supply and only 43.3% fulfilled demand after removing lower bound constraints (2.2) otherwise the model is infeasible the same as the previous one.

Finally, the launch of the Model with demand gap minimization (3) exactly as formulated for the whole week was feasible and showed 64.2% covering supply but 96.3% satisfied demand.

As concluded in [1], the most important goal for us – guarantying satisfaction of the lowest demand bound and then only maximize the overall sold units of fish. Nevertheless, it is always possible to tune the coefficients between objective parts to reach a more “correct” blend.

Instance	Variables	Constraints	Running time (sec)
Model (1)	70	17	0.01
Model (2)	89680	74600	0.13
Model (3)	82040	97120	0.11

Table 2 Instances characteristics

6.3 Sensitivity Analysis

Next, let us try exploiting Sensitivity Analysis for the Model with demand gap minimization (3) as the best in terms of demand satisfaction. As mentioned before, this type of analysis helps to investigate the rate of the change in the objective function as a

result of a change in the constraint resource, in our case supply vector. Since we may receive clear shadow prices only for the relaxed model, we withdraw integrality constraints (3.7) – (3.9), optimize again, and output values of dual variables for supply constraints. On the *Figure 14* below we may see an overall statistics of shadow prices values among all supply constraints. The vast majority is zero meaning that any changes in supply value for that fish group g and harvesting at date h don't influence the objective at all. Probably, the explanation is that an upper bound of constraints (3.2) is lower than the existing supply and it is not possible to sell more fish from the corresponding groups with corresponding harvesting dates. In addition, we have a small number of constraints with negative shadow prices implying that with increasing supply value for that particular fish groups and harvesting dates we would be able to increase the objective value proportionally to the value of shadow prices, i.e. decrease.

So as a maximum per one constraint we could have lost 5 times added supply amount from objective function value. However, as we see from *Figure 13* more often we overestimate the supply, so we should be ready for real supply being smaller and respectively objective value being larger.

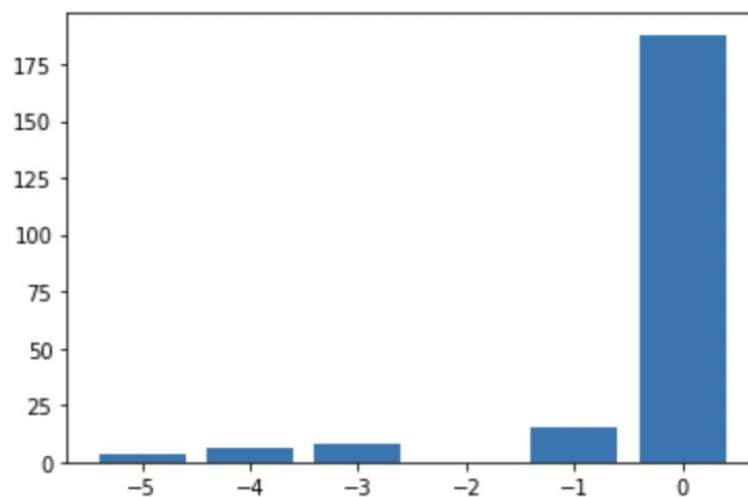


Figure 14 Bar plot of shadow prices for Model (3)

6.4 Robust worst-case oriented reformulation

Robust Optimization is a powerful tool for managing uncertainty, as discussed in *Methodology*. However, specifying a full uncertainty set seems to be ineffective for our discrete but large feasible set. Thus, we may consider the worst-case-oriented reformulation as an attempt to approach automated planning taking into account the effect of uncertainty. Let us define the worst-case goal as to minimize the risk of selling (assigning) fish that we actually don't have, i.e. forecast was overestimated. Then we should decide what the "safe" supply will be. For example, [19] says that often variance or standard deviation is used as a safe margin. But as examined in *Exploratory data analysis* our forecast error percentage is from the Cauchy distribution which is known by its undefined variance and consequently standard deviation. As a result, we are not able to estimate the variance from the data sample. *Table 3* specifies derived probabilities that supply is overestimated for a certain percentage and more, i.e. supply forecast error percentage is less or equal to a specific negative percentage value.

P(e% ≤ -150%)	P(e% ≤ -100%)	P(e% ≤ -50%)	P(e% ≤ -25%)	P(e% ≤ -10%)	P(e% ≤ -1%)
0.03	0.04	0.08	0.14	0.24	0.39

Table 3 Relative violation of supply constraints

(-200%, -1%]	(-150%, -1%]	(-100%, -1%]	(-50%, -1%]	(-25%, -1%]	(-10%, -1%]
94.6%	92.8%	87.2%	79.5%	64.1%	38.5%

Table 4 Covering negative percentage intervals

The "too pessimistic" choice would be the minimal error percentage. *Table 4* implies the distribution of samples among the negative half of the axis. When deciding the relevant safe margin we should realize that picked negative margin will improve overestimated forecasts while already underestimated forecasts will be reduced even more. After we determine the margin percentage we simply apply it to our uncertain forecast vector and feed it into the Model with demand gap minimization (3). For example, if we choose 79.6% risk minimization it means neglecting overestimation of 50% and less, i.e. minimization of forecast error percentage of -50% and more. Please, see below the calculations of the safe supply vector \hat{s} .

$$\hat{s} = \frac{\hat{s} * 100}{100 - margin_prc}$$

7. Conclusions

The aim of the thesis was to create an automated tool for fish assignment and to study the effect of uncertain supply data. First of all, the problem background was enclosed in detail. With these considerations as a starting point, a throughout literature review was done highlighting all the existing approaches connected to the topic and didactic case studies. In addition, some obstacles and remarks were discussed in terms of applicability to our specific problem.

Three mathematical models were proposed, described, implemented, and compared. The best *Model with demand gap minimization (3)* is based on given sets of customers, harvesting dates, sending dates, and fish groups. The formulation focuses on intelligent inventory usage and caring about demand satisfaction. Afterward, the real-life instances were solved to optimality with one of the best commercial solvers Gurobi within one second. The results were validated and interpreted. Sensitivity analysis for the solution was produced describing the effect of changing the supply vector.

The profound exploratory analysis uncovered data insights telling us more about problem structure and properties. The analysis showed how poor the short-term forecast is, so the main effect of uncertainty is simply struggling to allocate hundreds of non-existing supply boxes. This knowledge helped us in formulating worst-case oriented robust formulation of the *Model with demand gap minimization (3)*.

Eventually, the automated tool was written in Python. It reads input data, transforms it, and feeds it into the optimization solver. Thus in a couple of seconds, we get the assignment of fish between customers' orders. It is possible to run in deterministic and robust mode. The first one takes the supply as a true one, the second one puts the margin on the supply vector. And no, uncertainty degree derived from the current forecast cannot be neglected.

8. Prospects

Increasing dataset size, completeness and quality would bring even more value and precision to analysis. The study of uncertainty effect makes sense to perform with a more or less probable forecast, but for now improving forecast precision is worth doing. Thereby enhancing prognosis performance will relieve error distribution tails and consequently may change forecast error distribution from Cauchy to normal.

It would be wise to invest time trying to formulate and solve chance-constraint formulation, which probably is easier to do when operating with normal and not Cauchy distribution.

References

- [1] E. Molland, S. Knudseth, "Allocating Farmed Fish to Customer Orders Using Multi-Objective Optimization," 2021.
- [2] A. Ben-Tal, L. El Ghaoui, A. Nemirovski, Robust Optimization, Princeton and Oxford, 2009.
- [3] B. Liu, Theory and Practice of Uncertain Programming, Springer, 2009.
- [4] J. Duchi, "Optimization with uncertain data," Stanford, 2018.
- [5] Wim van Ackooij, Riadh Zorgati, René Henrion, Andris Möller, "Chance Constrained Programming and Its Applications to Energy Management," in *Stochastic Optimization - Seeing the Optimal for the Uncertain*, Intechopen, 2011.
- [6] Jean-Baptiste Hiriart-Urruty, Claude Lemarechal, Convex Analysis and Minimization Algorithms Fundamentals I: Fundamentals, Springer, 1993.
- [7] S. Peng, "Chance Constrained Problem and Its Applications," 2019.
- [8] George B. Dantzig, Mukund N. Thapa, Linear Programming, 1: Introduction, Springer, 1997.
- [9] V. V. Alseovich, V. V. Krahotko, Optimization methods: exercises and assignments, Minsk: Belarusian State University, 2005.
- [10] G. Butera, "How do I retrieve the (dual) Pi values for a MIP problem?," Gurobi, 2021. [Online]. Available: <https://support.gurobi.com/hc/en-us/articles/360034305272-How-do-I-retrieve-the-dual-Pi-values-for-a-MIP-problem->
- [11] N. Listiyani, "PyCon SG 2019," in *When Uncertainty Matters: Stochastic Programming for Inventory Model with Python*, <https://www.youtube.com/watch?v=5IhNiKt1IhE>.
- [12] T. Cokelaer, "Fitter documentation," 2019. [Online]. Available: <https://fitter.readthedocs.io/en/latest/>.
- [13] R. Raoniar, "Finding the Best Distribution that Fits Your Data using Python's Fitter Library," Medium, 2021. [Online]. Available: <https://medium.com/the-researchers-guide/finding-the-best-distribution-that-fits-your-data-using-pythons-fitter-library-319a5a0972e9>.
- [14] J. Grus, Data Science from Scratch, O'Reilly Media, Inc., 2019.

- [15] Y. Kharin, N. Zuev, E. Zhuk, Probability theory, mathematical and applied statistics, Minsk, 2011.
- [16] L. Gurobi Optimization, "Gurobi Optimizer," [Online]. Available: <https://www.gurobi.com/products/gurobi-optimizer/>.
- [17] H. Mittelmann, "BENCHMARKS FOR OPTIMIZATION SOFTWARE," 2022. [Online]. Available: <http://plato.asu.edu/bench.html>.
- [18] V. Menshikh, "Phys.: Conf. Ser. 973 012040," in *Models of resource allocation optimization when solving the control problems in organizational systems*, 2018.
- [19] J. Perednia, "Master Thesis Proposal," Molde, 2021.

Appendix

A. Example of input data

```
<SeaFoodAssignment>
<inst_name>Salmon Week 16 Friday</inst_name>
<week>16</week>
<plant>
  <name>Best_ever_plant</name>
  <days_to_Oslo>1</days_to_Oslo>
  <pallet_setup>0</pallet_setup>
</plant>
<spot_price>
  <species>0</species>
  <quality>0</quality>
  <size>0</size>
  <price>5.5</price>
</spot_price>
<order>
  <order_no>0</order_no>
  <customer>Best_ever_customer</customer>
  <order_type>Spot</order_type>
  <priority>1</priority>
  <species>0</species>
  <departure_Oslo>Tuesday</departure_Oslo>
  <num_days_for_packing>1</num_days_for_packing>
  <split>1</split>
  <health>ILA</health>
  <health>Listeria</health>
  <quantity>
    <quality>0</quality>
    <size>0</size>
    <min_quant>30</min_quant>
    <max_quant>80</max_quant>
    <price>2.5</price>
  </quantity>
  <total_quantity>
    <sup>
      <min_quant>30</min_quant>
      <max_quant>80</max_quant>
    </sup>
  </total_quantity>
```

```

</order>
<purchase>
  <number>0</number>
  <day>Monday</day>
  <location>Location_1</location>
  <plant>Best_ever_plant</plant>
  <producer>Best_ever_plant</producer>
  <species>0</species>
  <approval>Approval_1</approval>
  <approval>Approval_2</approval>
  <supply>
    <quality>0</quality>
    <size>0</size>
    <quantity>39</quantity>
  </supply>
</purchase>
</SeaFoodAssignment>

```

B. Data transformation

XML parsing

```

import pandas as pd
import numpy as np
import xml.etree.ElementTree as et

xtree = et.parse("data/salmon_wk16_Friday.xml")
xroot = xtree.getroot()

plants_cols = ["name", "days_to_Oslo", "pallet_setup"]
plants_rows = []
spot_prices_cols = ["species", "quality", "size", "price", "week"]
spot_prices_rows = []
orders_cols = ["orderID", "customer", "order_type", "priority", "species", "departure_Oslo", "num_days_for_packing", "split", "health", "total_quantity_min", "total_quantity_max", "week"]
orders_rows = []
o_quant_cols = ["orderID", "quality", "size", "min_quant", "max_quant", "price"]
o_quant_rows = []
supply_cols = ["supplyID", "plant_name", "approval", "day", "species"]
supply_rows = []
s_quant_cols = ["supplyID", "quality", "size", "quantity"]
s_quant_rows = []
purchase_ID = 0
week_num = 0

for node in xroot:
    if node.tag == 'week':
        week_num = int(node.text)

    if node.tag == 'plant':
        s_name = node.find("name").text if node is not None else None
        s_days_to_oslo = int(node.find("days_to_Oslo").text) if node is not None else None
        s_pallet_setup = bool(node.find("pallet_setup").text) if node is not None else None

```



```

    plants_rows.append({"name": s_name, "days_to_Oslo": s_days_to_oslo, "pallet_setup": s_pallet_setup})

    if node.tag == 'spot_price':
        species = node.find("species").text if node is not None else None
        quality = int(node.find("quality").text) if node is not None else None
        size = int(node.find("size").text) if node is not None else None
        price = float(node.find("price").text) if node is not None else None
        spot_prices_rows.append({"species": species, "quality": quality, "size": size, "price": price, "week": week_num})

    if node.tag == 'order':
        orderID = int(node.find("order_no").text) if node is not None else None
        customer = node.find("customer").text if node is not None else None
        order_type = node.find("order_type").text if node is not None else None
        priority = int(node.find("priority").text) if node is not None else None
        species = node.find("species").text if node is not None else None
        departure_Oslo = node.find("departure_Oslo").text if node is not None else None
        num_days_for_packing = int(node.find("num_days_for_packing").text) if node is not None else None
        split = bool(node.find("split").text) if node is not None else None
        total_quantity = node.find("total_quantity")
        total_quantity_child = total_quantity.find("sup")
        if total_quantity_child is None:
            total_quantity_child = total_quantity.find("prod")
        total_quantity_min = int(total_quantity_child.find("min_quant").text) if node is not None else None
        total_quantity_max = int(total_quantity_child.find("max_quant").text) if node is not None else None
        health = ""

        for elem in node.iter():
            if elem.tag == "health":
                health += elem.text + ","
            if elem.tag == "quantity":
                quality = int(elem.find("quality").text) if node is not None else None
                size = int(elem.find("size").text) if node is not None else None
                min_quant = int(elem.find("min_quant").text) if node is not None else None
                max_quant = int(elem.find("max_quant").text) if node is not None else None
                price = float(elem.find("price").text) if node is not None else None
                o_quant_rows.append({"orderID": orderID, "quality": quality, "size": size, "min_quant": min_quant, "max_quant": max_quant, "price": price})

        orders_rows.append({"orderID": orderID, "customer": customer, "order_type": order_type, "priority": priority, "species": species, "departure_Oslo": departure_Oslo, "num_days_for_packing": num_days_for_packing, "split": split, "health": health, "total_quantity_min": total_quantity_min, "total_quantity_max": total_quantity_max, "week": week_num})

    if node.tag == 'purchase':
        supplyID = purchase_ID
        plant_name = node.find("plant").text if node is not None else None
        day = node.find("day").text if node is not None else None
        species = node.find("species").text if node is not None else None
        approval = ""
        for elem in node.iter():
            if elem.tag == "approval":
                approval += elem.text + ","

```

```

        if elem.tag == "supply":
            quality = int(elem.find("quality").text) if node is not None else None
            size = int(elem.find("size").text) if node is not None else None
            quantity = int(elem.find("quantity").text) if node is not None else None
            s_quant_rows.append({"supplyID": supplyID, "quality": quality, "size": s
ize, "quantity": quantity})
            supply_rows.append({"supplyID": supplyID, "plant_name": plant_name, "day": day,
"species": species,
                               "approval": approval})
            purchase_ID += 1

print(str(week_num) + "th week in processing")
switcher = {
    "Monday": 1,
    "Tuesday": 2,
    "Wednesday": 3,
    "Thursday": 4,
    "Friday": 5,
    "Saturday": 6,
    "Sunday": 7,
}

```

Creating dataframes

```

plants = pd.DataFrame(plants_rows, columns = plants_cols)
spot_prices = pd.DataFrame(spot_prices_rows, columns = spot_prices_cols)
orders = pd.DataFrame(orders_rows, columns = orders_cols)
orders['departure_Oslo'] = list(map(lambda x: switcher.get(x), orders['departure_Oslo']
)
orders = orders.astype({"species": int})
order_quantities = pd.DataFrame(o_quant_rows, columns = o_quant_cols)
supply = pd.DataFrame(supply_rows, columns = supply_cols)
supply['day'] = list(map(lambda x: switcher.get(x), supply['day']))
supply = supply.astype({"species": int})
supply_quantities = pd.DataFrame(s_quant_rows, columns = s_quant_cols)

demand_data = orders[['orderID', 'customer', 'species', 'departure_Oslo', 'num_days_for_
packing']].merge(
    order_quantities[['orderID', 'quality', 'size', 'min_quant', 'max_quant', 'price']])

supply_data = supply[['supplyID', 'plant_name', 'day', 'species']].merge(
    plants[['name', 'days_to_Oslo']], left_on='plant_name', right_on='name')
supply_data.rename(columns={'day': 'harvest_day'}, inplace=True)
del supply_data['name']
supply_data = supply_data.merge(supply_quantities)

```

C. Data preprocess for Model with 1 day of planning horizon (1)

```

s1 = supply_data[['earliest_departure_fromOSL', 'harvest_day', 'species', 'quality',
'size', 'quantity']]
s1 = s1.groupby(['earliest_departure_fromOSL', 'species', 'quality', 'size'],
group_keys=False).apply( #harvest_day
                        lambda g: g.quantity.sum()
                        ).reset_index().rename(
                            columns={0: 'total_supply'})
d1 = demand_data[['departure_Oslo', 'customer', 'species', 'quality', 'size',
'min_quant', 'max_quant',
'num_days_for_packing',
'price']]

```

```

d1 = d1.groupby(['departure_Oslo', 'customer', 'species', 'quality', 'size', 'price'],
               group_keys=False).agg({'min_quant': 'sum',
                                     'max_quant': 'sum',

'num_days_for_packing': 'min'}).reset_index().rename(columns={
                                     'min_quant': 'lower_demand',
                                     'max_quant':
'upper_demand'}, inplace=False)
d_and_s = d1[d1['departure_Oslo'] > 1].merge(s1, how='left',
                                           left_on=['departure_Oslo', 'species',
'quality', 'size'],
                                           right_on=['earliest_departure_fromOSL',
'species', 'quality', 'size'])
d_INV1 = d1[d1['departure_Oslo'] == 2][['customer', 'species', 'quality', 'size',
'lower_demand', 'upper_demand',
'price']]
s_INV1 = s1[s1['earliest_departure_fromOSL'] == 2][['species', 'quality', 'size',
'total_supply']]

group_df = d_INV1[['species', 'quality',
'size']].drop_duplicates().sort_values(by=['species', 'quality', 'size'])
group_df['group'] = np.arange(len(group_df))
d_INV1 = d_INV1.merge(group_df).sort_values(by=['group', 'customer'])
s_INV1 = s_INV1.merge(group_df)

combinations = list(d_INV1[['group',
'customer']].drop_duplicates().itertuples(index=False, name=None))
prices = d_INV1['price'].to_list()
lb = d_INV1['lower_demand'].to_list()
ub = d_INV1['upper_demand'].to_list()
m_combinations, m_scores, m_lb, m_ub = gp.multidict({combinations[i]: [
prices[i], lb[i], ub[i]] for i in
range(len(combinations))})
supply = s_INV1[['total_supply', 'group']].sort_values(by=['group'])['total_supply']
G = sorted(set(group_df['group']))
C = sorted(set(d_INV1['customer']))

```

D. Data preprocess for Model for the week planning horizon (2) and Model with demand gap minimization (3)

```

s1 = supply_data.groupby(['harvest_day', 'days_to_Oslo', 'species',
'quality', 'size'], group_keys=False).apply(
lambda g: g.quantity.sum()
).reset_index().rename(
columns={0: 'total_supply'})

d1 = demand_data.groupby(['departure_Oslo', 'customer', 'species', 'quality', 'size', 'p
rice'],
group_keys=False).agg({'min_quant': 'sum',
'max_quant': 'sum',
'num_days_for_packing': 'min'}).reset_index(
).rename(columns={
'lower_demand': 'min_quant',
'upper_demand': 'max_quant'
}, inplace=False)

d_and_s = d1[d1['departure_Oslo'] > 1].merge(s1, how='left',
left_on=['species', 'quality', 'size'],
right_on=['species', 'quality', 'size'])

```

```
d_and_s = d_and_s[d_and_s['num_days_for_packing'] >= d_and_s['days_to_Oslo']] # define f
reshness parameter
d_and_s = d_and_s[(d_and_s['departure_Oslo'] - d_and_s['harvest_day'] <= d_and_s['num_da
ys_for_packing']) &
                (d_and_s['departure_Oslo'] - d_and_s['harvest_day'] >= d_and_s['days_to
_Oslo'])]
```

Data preparation for using solver

```
d_INV2 = d_and_s[['species', 'quality', 'size', 'harvest_day', 'days_to_Oslo', 'departur
e_Oslo', 'customer',
                'lower_demand', 'upper_demand', 'price', 'num_days_for_packing']].drop
_duplicates()
s_INV2 = d_and_s[['species', 'quality', 'size', 'harvest_day', 'days_to_Oslo', 'total_su
pply']].drop_duplicates()

group_df = d_INV2[['species', 'quality', 'size']].drop_duplicates().sort_values(by=['spe
cies', 'quality', 'size'])
group_df['group'] = np.arange(len(group_df))
d_INV2 = d_INV2.merge(group_df)
s_INV2 = s_INV2.merge(group_df)

harvest_region_df = d_INV2[['harvest_day', 'days_to_Oslo']].drop_duplicates().sort_value
s(by=['harvest_day',
     'days_to_Oslo'])
harvest_region_df['harvest_region'] = np.arange(len(harvest_region_df))
d_INV2 = d_INV2.merge(harvest_region_df).sort_values(by=['group', 'harvest_region', 'dep
arture_Oslo', 'customer'])
s_INV2 = s_INV2.merge(harvest_region_df).sort_values(by=['group', 'harvest_region'])

G = sorted(set(group_df['group']))
H = sorted(set(d_INV2['harvest_region']))
S = sorted(set(d_INV2['departure_Oslo']))
C = sorted(set(d_INV2['customer']))

combinations = [(g, h, s, c) for g in G for h in H for s in S for c in C]
comb_df = pd.DataFrame(combinations, columns=['group', 'harvest_region', 'departure_Oslo
', 'customer'])
comb_df = comb_df.merge(d_INV2[['group', 'harvest_region', 'departure_Oslo', 'customer',
                              'price', 'lower_demand', 'upper_demand', 'num_days_for_packi
ng']]).drop_duplicates(),
                    how='left').sort_values(by=['group',
                    'harvest_region', 'departure
_Oslo', 'customer'])
comb_df['freshness'] = comb_df['price'].notnull()
comb_df = comb_df.fillna(0)

prices = comb_df['price'].to_list()
freshness = comb_df['freshness'].to_list()
m_combinations, m_scores, m_freshness = gp.multidict({combinations[i]:
                    [prices[i], freshness[i]] for i in
range(len(combinations))})

g_s_c = [(g, s, c) for g in G for s in S for c in C]
v_bounds = pd.DataFrame(g_s_c, columns=['group', 'departure_Oslo', 'customer']).merge(d
_INV2[['group',
     'departure_Oslo', 'customer', 'lower_demand', 'upper_demand']].drop_duplicates(),
                    how='left').fillna(0).sort_values(by=['group', 'departur
e_Oslo', 'customer'])
lb = v_bounds['lower_demand'].to_list()
```

```

ub = v_bounds['upper_demand'].to_list()
m_g_s_c, m_lb, m_ub = gp.multidict({g_s_c[i]: [lb[i], ub[i]] for i in range(len(g_s_c))})

g_h = [(g, h) for g in G for h in H]
v_supply = pd.DataFrame(g_h,
                        columns=['group', 'harvest_region']).merge(s_INV2[['total_supply',
                        'group', 'harvest_region']].drop_duplicates(), how='left')
                        ).fillna(0)['total_supply']

penalties_df = pd.DataFrame(g_h,
                           columns=['group', 'harvest_region']).merge(harvest_region_df[['harvest_
region', 'harvest_day']])
penalties_df['penalty'] = max(penalties_df['harvest_day']) + 1 - penalties_df['harvest_d
ay']
v_penalty = penalties_df['penalty']
m_g_h, m_supply, m_penalties = gp.multidict({g_h[i]: [v_supply[i], v_penalty[i]] for i i
n range(len(g_h))})

M = 10000

```

E. Implementation of Model with 1 day of planning horizon (1)

```

# Declare and initialize model
m = gp.Model('INV1')
# Create decision variables for the INV1 model (1.4, 1.2u)
x = m.addVars(combinations, ub=m_ub, vtype=GRB.INTEGER, name="assign")
# Create resource constraints (1.3)
m.addConstrs((x.sum(g, '*') <= supply[g] for g in G), name='resource')
# Objective: maximize total matching score of assignments (1.1)
m.setObjective(x.prod(m_scores), GRB.MAXIMIZE)
# Run optimization engine
m.optimize()

Postprocessing results
sol = []
for g, c in combinations:
    sol.append(int(x[g, c].x))
d_INV1['assign'] = sol

print(str(sum(sol)*100 / sum(supply)) + "% of supply or " + str(sum(sol)) + " units allo
cated")
not_satisfied = d_INV1[d_INV1['lower_demand'] > d_INV1['assign']]
val_not_satisfied = sum(not_satisfied['lower_demand']) - sum(not_satisfied['assign'])
print(str(100 - 100 * val_not_satisfied/ sum(d_INV1['lower_demand'])) + "% of demand sat
isfied")

```

F. Implementation of Model for the week planning horizon (2)

```

# Declare and initialize model
m = gp.Model('INV2')
# Create decision variables for the INV2 model (2.6)
x = m.addVars(combinations, vtype=GRB.INTEGER, name="assign")
# Create delta variables for the INV2 model (2.7)
delta = m.addVars(g_h, vtype=GRB.INTEGER, name="delta")
# Create lower demand (zero) constraints (2.2L)
m.addConstrs((x.sum(g, '*', s, c) >= 0 for g in G for s in S for c in C), name='lower de
mand (zero)')

```

```

# Create upper demand constraints (2.2u)
m.addConstrs((x.sum(g, '*', s, c) <= m_ub[g, s, c] for g in G for s in S for c in C), name='upper demand')
# Create freshness constraints (2.3)
m.addConstrs(x[g, h, s, c] <= M * m_freshness[g, h, s, c] for g in G for h in H for s in S for c in C)
# Create supply constraints (2.4)
m.addConstrs((x.sum(g, h, '*', '*') <= m_supply[g, h] for g in G for h in H), name='supply')
# Create leftover constraints (2.5)
m.addConstrs((x.sum(g, h, '*', '*') + delta[g, h] == m_supply[g, h] for g in G for h in H), name='leftover')
# Objective: maximize total matching score of assignments
# Leftovers are heavily penalized (2.1)
m.setObjective(x.prod(m_scores) - delta.prod(m_penalties), GRB.MAXIMIZE)
# Run optimization engine
m.optimize()

```

Postprocess in the same as below

G. Implementation of *Model with demand gap minimization (3)*

```

# Declare and initialize model
m = gp.Model('INV3')
# Create decision variables for the INV2 model (3.7)
x = m.addVars(combinations, vtype=GRB.INTEGER, name="assign")
# Create delta variables for the INV2 model (3.8)
delta = m.addVars(g_h, vtype=GRB.INTEGER, name="delta")
# Create demand_gap variables for the INV3 model (3.9)
demand_gap = m.addVars(g_s_c, vtype=GRB.INTEGER, name="demand_gap")
# Create lower demand (zero) constraints (3.2l)
m.addConstrs((x.sum(g, '*', s, c) >= 0 for g in G for s in S for c in C), name='lower demand (zero)')
# Create upper demand constraints (3.2u)
m.addConstrs((x.sum(g, '*', s, c) <= m_ub[g, s, c] for g in G for s in S for c in C), name='upper demand')
# Create freshness constraints (3.3)
m.addConstrs(x[g, h, s, c] <= M * m_freshness[g, h, s, c] for g in G for h in H for s in S for c in C)
# Create supply constraints (3.4)
m.addConstrs((x.sum(g, h, '*', '*') <= m_supply[g, h] for g in G for h in H), name='supply')
# Create leftover constraints (3.5)
m.addConstrs((x.sum(g, h, '*', '*') + delta[g, h] == m_supply[g, h] for g in G for h in H), name='leftover')
# Create gap to lower demand constraints (3.6)
m.addConstrs((x.sum(g, '*', s, c) + demand_gap[g, s, c] == m_lb[g, s, c] for g in G for s in S for c in C),
name='demand gap')
# Objective: maximize total matching score of assignments
# Leftovers are heavily penalized (3.1)
m.setObjective(demand_gap.sum() + delta.prod(m_penalties), GRB.MINIMIZE)
# Run optimization engine
m.optimize()

```

Postprocessing results

```

sol = []
sol_delta = []
sol_gap = []

```

```

for g, h, s, c in combinations:
    sol.append(x[g, h, s, c].x)

for g, h in g_h:
    sol_delta.append(delta[g, h].x)

comb_df['assign'] = sol

print(str(sum(sol)*100 / sum(v_supply)) + "% of supply or " + str(sum(sol)) + " units
allocated")

not_satisfied = comb_df
not_satisfied = not_satisfied.groupby(['group', 'departure_Oslo', 'customer',
'lower_demand']).agg({'assign':'sum'}).reset_index()
not_satisfied = not_satisfied[not_satisfied['lower_demand'] > not_satisfied['assign']]
val_not_satisfied = sum(not_satisfied['lower_demand']) - sum(not_satisfied['assign'])
print(str(100 - 100 * val_not_satisfied/ sum(v_bounds['lower_demand'])) + "% of demand
satisfied")
print(str(val_not_satisfied) + " demand units out of " +
str(sum(v_bounds['lower_demand'])) + " are not allocated")

```

H. Sensitivity analysis for Model with demand gap minimization (3)

```

negative_prices = []

for g, h in g_h:
    negative_prices.append(m.getAttr("Pi", [m.getConstrByName('supply[' + str(g) + ', ' +
str(h) + '']')])[0])

coord = sorted(list(set(negative_prices)))
counts_val = []
for n_p in coord:
    counts_val.append(negative_prices.count(n_p))
plt.bar(coord, counts_val)

```

I. The figure of fitted distributions for supply forecast error percentage

