Master's Thesis in Informatics
Study specialization: Optimization

Optimizing storage of fish feed

An experimental study on maximizing the available space at Skretting's storage facility.

Rune Reimertz

Department of Informatics
University of Bergen
2021

Abstract

In this Master's thesis I develop a mixed integer linear programming (MILP) optimization model. The model is designed to propose storage placements for upcoming production of fish feed. The placements should leave as much space as possible free for unknown production quantities due beyond the planning horizon. Hence, the model minimizes the storage space occupied by the feed, at the end of the planning horizon. I am supplied a data set containing all production and shipping orders over a time span of two months. To be able to apply this data to my model, I write a Python script which cleans and formats the data to the desired configuration. The model is multi-periodic, and a rolling horizon is utilized to deal with the continuous time span, subject to frequent change. The model is implemented in the algebraic modeling language of AMPL (A Mathematical Programming Language). Initial runs of the model show that I am able to help Skretting solve their storage problem. However, the model proves to give room for multiple optima, and to utilize this I further extend my model implementation with two additional objective functions. The secondary and tertiary objective functions consider other important aspects regarding the placement of feed. With the new objective functions implemented I experiment with continuous relaxations, to observe if the model implementation can be tightened with additional constraints. The relaxations result in the introduction of three constraints, which I prove to be satisfied at integer optimum by the existing constraints, and therefore redundant. After settling the model implementation I shift focus towards the data, and conduct experiments with three different time resolutions. The experiments help me tune the time resolution the model is to be run with for the best possible results.

Contents

1 Aquaculture, from past to present **1**
1.1 Aquaculture 1
1.2 History 2
1.3 Present situation 3

2 Feed production **5**
2.1 History 5
2.2 Modern feed production 6
2.2.1 Recipe and raw materials 6
2.2.2 The pellet 8
2.2.3 Manufacture 9

3 Getting to know Skretting AS **10**
3.1 History of Skretting AS 10
3.2 Feeding the future 12

4 Problem formulation **14**
4.1 Production and storing norms 15
4.2 Related work and published literature 17

5 Mathematical model **18**
5.1 Sets, parameters and variables 18
5.2 Mixed integer linear programming model 19
5.3 Further development of the model 22
5.3.1 Secondary objective function 22
5.3.2 Tertiary objective function 23
5.3.3 Tightening constraints 24

6 Design choices **26**
6.1 Rolling horizon time plan 26

6.2 Choice of primary objective function 28
6.3 Time resolution 29

7 Experimental studies 31
7.1 Goals 31
7.2 Data set 32
7.3 Constant parameters 34
7.4 Initial runs 34
7.5 Full runs 35
7.5.1 Computational burden 37
7.5.2 Objective function values 42

8 Discussion 50
8.1 Intention of use 50
8.1.1 Additional usage 51
8.2 Implementation 51
8.3 Results 52
8.4 Further development 54

9 Conclusion 56

References 57

Chapter 1

Aquaculture, from past to present

1.1 Aquaculture

Aquaculture is defined as the breeding, upbringing, and harvesting of fish, shellfish, algae, and other organisms in all types of water environments [19]. The concept is similar to agriculture, just with fish instead of plants and stock. Aquaculture is also known as fish farming [1].

To have enough sustainable animal protein for a global population estimated to reach 10 billion by 2050, it is important that the world focus on aquaculture [1]. This is because wild fish in the oceans have been overfished and exploited above a sustainable level for generations, and the oceans can not naturally provide these enormous amounts by themselves [1].

There are multiple ways to farm fish, but I will in this short paragraph explain a modern, popular way of doing so. In the first stage of farming, the fish is bred, hatched and brought through its early life stages. This is done on land, in smaller tanks and pools, which are monitored and controlled. Once the fish reaches a certain size and is sufficiently mature, it is moved to the farm in the sea where it spends the majority of its life swimming and eating. For the entirety of its life, the fish is fed with feed produced at the feed mills, such as Skretting. Once the fish reaches its desired harvest size, it is transported to the process facility where it is being processed, packed and shipped to its customers [1].

1.2 History

The process of capturing and keeping fish trapped has been around since roughly 4000 years B.C. It all began with people trapping fish in ponds, lagoons and small shallow lakes, so the fish would be available at all times. In the middle ages the pond farming techniques began developing, e.g. in Southern Europe where ponds and lagoons were modified to trap fish swept in by the tide. This practice was easily combined with the seasonal salt production.

A large breakthrough for the industry came in 1741 when a German by the name of Stephan Ludwig Jacobi built the first trout hatchery, where he was able to make the fish reproduce [8]. This meant that fish farmers no longer were solely dependent upon nature, and catching fish. However, it was not until roughly hundred years later, in the 19th century, when farmers were no longer satisfied with what nature supplied on its own, that they started to introduce fish fry from hatcheries and supplemental feed. This was partly because the increased need of food due to population growth, and the industrial revolution polluting and claiming lakes and rivers. As the industrial revolution roared in the western part of the world, fish farming caught the eye of scientific researchers. The species that underwent scientific research were white fishes, charrs, Atlantic salmon (*Salmo salar*), and rainbow trout (*Oncorhynchus mykiss*) [8]. The rainbow trout from America proved to be among, if not the best species adapted to fish farming. It could grow faster, live in higher densities, and was hardier [8].

Until the mid 20th century fish farming had a limited growth, as a result of unsuitable fish feed and diseases. The fish had typically been fed meat and grain, but in 1954, researchers at the Oregon Fish Commission developed a fish food in moist pellet form. The ingredients varied, but in 1956 it consisted of frozen tuna guts, herring meal, cottonseed oil meal, corn oil, folic acid, and niacin [13]. The process of making pellets made addition of nutrients, vitamins and antibiotics easy. The introduction of pellets decreased costs, need for storage space, and water pollution, compared to traditional feed [13]. Through the discovery of health and dietary needs of fish throughout the different stages of its development, a greater success followed.

Fish farmers from Japan created the first floating cages about the same time as pellets were introduced. The cages were anchored to the seafloor, had a large pocket shaped net, and were kept afloat by wood. This idea was adopted throughout the world soon after, and in 1959, Norwegian brothers Karstein and Olav Vik built the first cage to hold Atlantic salmon, in Norway [13]. The introduction of the new cages made the production of Atlantic salmon possible in Europe. Salmon was becoming scarce in the wild, and was considered a fish for the upper class. With the introduction of cheap salmon farmed in Europe, the beginning of a new and successful era for the fish farmers started. Norway has particularly good conditions for farmers to locate, as the fjords provide shelter from the harsh weather conditions in the North Sea, and the water temperatures remain relatively stable throughout the year. In 1970, Norwegian brothers Sivert and Ove Grøntvedt put 20,000 Atlantic salmon smolt into floating octagonal cages they had designed and built. The cages made it easier to feed the salmon, and gave protection against predators. This was considered the world's first successful salmon farm [13].

In the 1980s one saw a new type of fish farms appear in Denmark. They were constructed on land and used recirculating aquaculture system (RAS) technology. This system reuses water by filtering it between every time it circulates through the facility. Farming fish on land has the advantage of eliminating some of the largest concerns for the farmers, such as fish escaping, or suffering from diseases. One is also able to control oxygen levels, temperature and algae growth in the water. Since, RAS fish farms have become more frequent and are still developing as an alternative to farms in the sea [13].

1.3 Present situation

Today one can witness a number of different technologies for farming fish throughout the world. Some methods are e.g. floating cages of different dimensions, land based RAS, and most recently, large offshore farms [13].

Offshore fish farms are not protected by sheltered waters, like the farms in the Norwegian fjords. One major advantage of this is the dilution that the rushing water creates. The dilution makes it easier to avoid pollution and disease. The pollution is the fish waste that normally would build up on the sea bottom, which the waves and current of the deep waters now flush out [15]. The North Sea would traditionally not be considered for this type of farm, because of the harsh and stormy weather conditions, but new technology has proven to be promising. The Norwegian company SalMar is conducting offshore biological and technological pilot testing with their Ocean Farm 1 [22].

After oil and gas, fish is Norway's largest export (as of November 2020) [26], and in 2019, 36 million meals of Norwegian fish were served on average every day worldwide. 16 million of these meals were farmed Atlantic salmon, making Norway the largest exporter of farmed Atlantic salmon in the world [23]. Fish farming was in 2019 responsible for 45 percent of the total volume fish exported, and 71 percent of all earnings made in the Norwegian fish industry [23]. This proves how successful the Norwegian farming industry has become. Fish farmers are capable of exporting fewer tones, but still makes more money than traditional fishing.

In 2012, the amount of farmed fish succeeded farmed beef by a clear margin, for the first time [15]. One major advantage for farming fish is that they only need just over 1 kilo feed per 1 kilo body weight gained. Beef needs nearly 7 kilos, pork nearly 3 kilos and chicken nearly 2 kilos of feed per kilo body weight gained. However, this obviously depends on factors like the quality and content of the feed, and the local living environment of the animals/fish. Fish can turn more of their consumed feed into body weight, as they need fewer calories due to being cold blooded, and not having to fight as much gravity in their buoyant environment [15].

Chapter 2

Feed production

The human population is growing, and as a consequence the demand for food is also increasing. To satisfy this growing demand, fish farming has seen an increased expansion and growth [21]. Fish is a great source of nutrition as it contains protein, healthy fats such as omega-3, vitamins such as D and B12, and minerals such as Ca, Ka, Mg, Fe, Zn, I, P [12]. This, together with the low conversion factor between feed needed per kilo fish raised [15], makes fish an excellent option for sustainably feeding the population of the world. Together with the increased interest in aquaculture, there comes an increased interest and need for aquafeed [21]. For a fish farmer, feed is the single greatest expense, and can contribute up towards 80 percent of all costs involved in farming fish [21]. It is therefore in the farmers interest to obtain the highest amount of healthy, high quality, and fast growing fish, for their money spent on feed. As a result, large sums have been invested in finding the optimal fish feed recipes, raw materials and manufacture [14].

2.1 History

As mentioned in Section *1.2*, manufactured fish feed was introduced in the 1950's. It was a soft and moist pellet, which served as a replacement for dry grains and meat which had previously been used. This gave fish farmers more control of the nutrients and vitamins supplied to the fish. If needed, antibiotics could also easily be added. The first pellets were based on raw materials which resembled what the fish ate in its natural habitat.

Salmon, for example, eat fish and shrimp. The researchers therefore based the diet for farmed fish on raw materials such as fish meal and fish oil [21]. These are bi-products of the traditional fish industry, and were easy accessible.

With the new and more nutritional fish feed, there was simply less feed needed for growing fish to the same size as before. The pellets were also easier to store, and reduced the need for storage space. Traditional grains and meats had to a greater extent polluted the water in ponds used for farming, hence the pellets cut labor needed for cleaning. After two years using the pellets, the costs to produce a kilogram of fish was reportedly 41 percent less than previously [13].

2.2 Modern feed production

2.2.1 Recipe and raw materials

Fish feed producers are dependent upon having nutritional rich feed, as feed is the single most important factor for having healthy, and fast growing fish [21]. In the wild, a salmon may eat as much as 10 kilos fish per kilo body weight gained [24]. For farmed fish fed optimized feed, this has been reduced to just above 1 kilo feed, per kilo body weight gained. If fish farmers buy nutritionally good feed, they will need less feed to increase the weight of the fish. As a consequence, they will save resources spent on storing, feeding and transporting a greater amount of less optimized feed. Hence, to stay competitive, the fish feed manufacturers are interested in buying nutritionally rich raw materials for their feed [21]. This might be expensive, and the trick to stay competitive among other manufacturers is to find a balance where the cost of raw materials are weighed up against their nutritional values.

Even though recipes are, for the most part, well established formulations, the fish feed producer will most likely make small adjustments on a regular basis. The raw materials may differ in nutritional values, depending on which batches, producers, countries or regions they come from. Where the raw materials are bought from depends on what raw materials are available in the market, and at what price. To keep costs as low as possible, the fish feed producer might order raw materials at the lowest market price. However, if low price means low nutrition, this may potentially lead to the producer having to make variations in the formula to compensate for the lack of nutrition. As such, there needs to be bought more of the raw material in question, leading to increased expenses. The fish feed producer does, after all, have a responsibility to make sure the feed performs as promised. With the right formulations the fish gets the nutrition it needs to grow fast, and with the correct balance of fat and muscles. For example, if the feed does not contain enough protein compared to energy, the fish will have more than an optimal amount of fat in its muscle and visceral cavity. If the fish is to be processed, such as filleted, before being sold, this fat is wasted. As such, some of the feed which has been fed to the fish has ultimately been wasted [21]. In short, it might not always be beneficial towards cost, to order the cheapest raw material alternatives. This is a cost/profit-analysis which the fish feed producer needs to make.

As the increasing environmental problem is in focus of consumers, it is important that the feed is sustainable, and does not contain any unwanted substances. Feed contributes to the majority of the carbon foot print made by the fish farming industry [7]. This is mainly due to the growth, harvest and transport of raw materials used in the feed production, as well as the production and transport of the feed. To meet the ever-increasing climate concerns, the fish farmers are interested in having a carbon foot print as low as possible. In recent years, there has been a significant improvement in making fish feed more sustainable [17]. Where there in the 1990's could be as much as 80 percent fish meal and oils used in the feed, this can now be as low as 30 percent. This is thanks to the introduction of vegetable raw materials as substitutes [24]. In the wild, salmon gets it red/pink color from feeding on shrimp. Instead of using shrimp scales to give the salmon its natural colour, Astaxanthin, which is a colour pigment naturally occurring in certain algae, is added [25]. Other commonly used raw materials are soybeans, wheat, fababeans, insect meal, fish meal and fish oil [5]. One concern with using plant based materials is that they are subject to growing moulds if not stored properly. Moulds may produce mycotoxins, which can be harmful [20]. This has to constantly be supervised closely.

2.2.2 The pellet

Different fish in different parts of its life cycle eat feed with different physical attributes. This includes size, hardness, moisture and density among others. Across the globe one can witness waste from processed fish and animals, as well as plants, mashes, minces and pellets being fed to farmed fish. At large-scale fish farms in developed countries, pellets are a commonly preferred feed type [21]. This is also what the fish feed producer in question for this thesis is producing. I will therefore focus this explanation around the manufacture of extruded pellets.

Pellets are preferred by fish farmers for multiple reasons. Pellets tend to hold their form, and not dissolve too quickly when exposed to water. Hence, more feed can be eaten by the fish before the feed is dissolved and not consumable. Unlike feed in form like mash and mince, the pellets are dry. This is preferred as water adds weight and size, which lower the pellet's value for weight. By only having small amounts of moisture, one can save transport and storing costs, as well as reduce the global emissions. Pellets can be transported in both bag and bulk, which makes transport and storing convenient.

The density is important, as it controls the buoyancy of the pellet in the water. Some fish species prefer to feed at the surface, some under the water, and others at the bottom. Extruded pellets can easily be manipulated to meet the desired density from the fish farmers – this is often not the case for the other forms of feed mentioned [21].

The hardness contributes to how long the pellet stay in its desired form before eventually dissolving in the water. This is important as some species eat slow, and need a longer time to feed [21].

The size of the pellet is dependent upon the size and species of the fish. Larger fish will generally eat both small and large pellets, but with pellets adjusted to their size, their growth will be optimized to the fullest. With small pellets the large fish will waste energy gathering and eating more pellets than it would have to, if the pellet was sized correctly. It is more beneficial for the fish farmer if the fish save this energy and instead use it to grow [21].

2.2.3 Manufacture

Extrusion is a production process which is often used in modern large scale fish feed production [11]. In an extruded process the mix of raw materials is exposed to a high temperature process for a short time. This conserves the nutrients better, and makes the starch and protein more digestible, compared to pelleted feed [10][11]. In short, the full production process starts by grinding the raw materials, adding fluids and then mixing this dough. Mixing the raw materials well will distribute the nutrition evenly in each pellet, and lead to all fish eating the same even composition of nutrients when feeding. This is preferred as the fish will grow at the same pace. Hence, it can be fed the same size feed and be slaughtered at the same time. After mixing, the dough is passed on to the extruded process where it is exposed to high temperatures, pressed through slots of the desired diameter, and cut in desired lengths to finally end up in the pellet shape ordered. Next up is drying in an oven to get the desired hardness and coating the pellets with oil. The pellets are at last cooled, and then sent to be packed [11].

Dust and small particles of feed will generally tend to float instead of sink, and tends to be ignored by larger fish. This feed eventually drifts away, or dissolves, and is in that way wasted. The customers are not interested in buying feed that the fish does not eat. It is the manufacturers responsibility to ensure that there is as little dust and broken pellets as possible in the sold feed. During the production process the pellets are moving between different stages, which can lead to them breaking apart to dust and smaller particles. By using large sieves and screens the broken pellets are sorted out. However, one can not be entirely sure that the feed is delivered free of broken pellets, as the pellets may be crushed during transportation and storing.

Chapter 3

Getting to know Skretting AS

Skretting is a world leading fish feed producer, with head office in Stavanger, Norway. Skretting employs approximately 3,500 people, and produce feed in 20 countries, on 5 continents. Globally, the company delivers feed to more than 60 species from hatching until slaughter. The company claims to produce feed equivalent to 20 million meals of seafood per day, worldwide [3].

3.1 History of Skretting AS

The company was founded as "Stavanger Landbrugsforretning" by H. O. Oftedahl in 1899 in Stavanger, Norway. The recently established business produced and sold equipment, products and feed related to agriculture and stock. In 1913 Torgeir Skretting joined the company as a companion, and by 1916 he was in charge of operations all by himself. After running the company by himself for 12 years, Torgeir Skretting bought the company in 1925, and became the only owner. The company was renamed T. Skretting, and continued operations as a family business. By the end of the 1950's interest in aquaculture was increasing, and in 1960 the company started trials with dry feed made for farmed rainbow trout raised in wooden cages. The trials proved successful, and in 1963 dry feed for rainbow trout was presented to the market for the first time. In 1975 the first dry feed made for sea raised salmon was launched in the market. At this time, the company was only producing and experimenting with fish feed as a side operation, and the main focus was towards agriculture products, as well as cars, oil and other minor operations. However, times were tough, and it proved hard to be proficient on all these areas while still making a profit. As a result, the decision to prioritize fish feed production was made in 1979. The next years saw a 30% annual increase in fish feed production. This gained the attention of the Dutch company Trouw, which acquired 50% of the shares in the feed manufacturer [14].

In 1983 Skretting opened its new fish feed factory at Averøy. At the time of opening, the factory was the world's first factory singly devoted to producing fish feed, as well as Norway's first factory to have a production line devoted to small feeds meant for juvenile fish. There was also a fully enclosed production line within the factory which produced medical feeds. Medical feeds need to be produced by strict rules and regulations, and were therefore kept separate. The same year Skretting became the world's first manufacturer to start commercial production of extruded fish feed. Extruded feed revolutionized the quality of the feed, and made it easier to handle for both producer and fish farmer. The first extruded products sold contained an astonishing 23% fat, which was a landmark compared to the mere 6% in 1964. By being the first fish feed manufacturer to utilize this new technology, Skretting gained an advantage to their competitors. Throughout the 1990's further innovative thinking by employees resulted in a patented process which was able to include even more fat in the feed, and create so called high energy feed, with 40% fat content[14].

The Skretting Aquaculture Research Centre (ARC), was opened in 1989, and was designed to conduct research towards implementing innovative and sustainable solutions for the aquaculture industry [4]. Focus at Skretting ARC is to improve and optimize:

- fish health and well-being.
- the nutrition needed to optimize growth while maintaining food safety, quality and sustainability.
- the production process, where changing formulations gives room to develop new techniques and methods.

After having acquired the majority of Skretting over the past 19 years, Trouw acquired the last 25% of shares in 1990, and the company's history as a family owned business was over. After being both a agriculture and aquaculture company for the previous decades, the company stopped all other operations the same year, and was now fully devoted to producing only fish feed. Trouw, which owned Skretting under its daughter company BP Nutrition, formed Nutreco, which was to solemnly focus on feed production [14]. In 1990 NIR (Near-infrared spectroscopy) was introduced as a quick and accurate way to quality check the nutritional values for ingredients, intermediate products and finished feeds. 1996 saw the introduction of AminoBalance, which raised the protein value of the feeds, as well as the first ever AquaVision – a biennial global aquaculture conference in Stavanger. In 1997 the Feed Technology Plant (FTP) was opened in Stavanger. FTP uses small scale production equipment to explore new production methods/technology, new formulations and new raw materials. The small scale enabled an efficient optimization of equipment, production methods and production of small batches prototype feeds. A joint-venture in 1998 gave Skretting access to the Asian market, through Japan. All aquaculture parts of Nutreco were rebranded as Skretting, and Nutrace, which is a system for ensuring food safety and quality, was launched the very same year. The company joined the WWF aquaculture dialogue in 2004 and launched the SEA (Sustainable Economic Aquaculture) program in 2008. The African market was entered in 2013. In 2019 Nutreco and Skretting won "edie Sustainability Leaders Award" in the category of "Sustainability Product Innovation of the Year". In recent years the company has continued to grow, as well as invest in research, sustainability and good initiatives [4].

3.2 Feeding the future

The vision at Skretting is "Feeding the Future". To reach this vision, as well as the goals set by Skretting ARC listed on the previous page, the company is not only dependent upon doing good research, but also utilizing technology. The company has always been on the frontier of development, and technology has clearly had a role in this progress. In the mid 1960's, linear programming was used in optimizing feed recipes, and in 1994 a software product called WinMix was developed by Skretting [14]. By using technology to create an effective storage plan, Skretting is one step closer towards making sure that customers get the correct feed that they order, that the feed arrives on time, and at a price all parts involved can be satisfied with. These are all factors which contribute to the manufacturer's responsibility towards optimizing fish growth and profit.

Raising the efficiency of these operations will most likely result in increased competitiveness and profit. This increased profit can again be used to finance further research, development and technology – leading Skretting step-by-step into the future. Thanks to this thesis, which indirectly aims at aiding the fish feed producer towards having a storage as effective as possible, these goals might be closer to reach than ever before. At the storage location in question, there is need for a storage plan which takes into consideration the unknown amount of feed which is due ahead of time. This is to make sure there is always room for all feed produced. If this challenge is not solved to a sufficient extent, the producer will have to cancel orders, as there is no place to put the feed (see Section 4 for further information). This will most likely in the long term impact the results and reputation of the producer in a negative way. This thesis aims to show how technology can be used as an advantage to reach the goal of an efficient storage facility.

I wish to end this chapter with a paragraph from the book "Skretting i hundre 1899-1999" found on page 30 (see reference [14]), which I think sums the Section up well:

> The history has shown that development and ability to use new technology has resulted in new progress, and good results for coworkers, as well as for owners, customers and Norwegian aquaculture industry. Skretting is determined to lay one step ahead, also in the next century.

Chapter 4

Problem formulation

Skretting Stavanger wants a computer program which helps them optimize the storage of fish feed. The program should take the different types and amounts of feed being produced as input, and then propose the best possible storage placements for this feed. Best possible placement is considered as the placement which minimizes the amount of space being occupied. The company wants to make sure they always have enough space available for scheduled production, and they ensure this by making good plans for where the upcoming production of feed is to be stored. **In this thesis I create a minimization model which fulfills the manufacturer's needs, as well as their rules and norms for operations. Then, experiments with the model are conducted. The experimental results are useful for detailed tuning of the model, particularly the choice of redundant constraints to be included. Through an analysis of different resolutions in the time discretization, the experiments also provide useful insight about how to operate the model implementation in practice.**

As of now, a few experienced workers solve this problem manually by walking around the storage with lists of scheduled production and shipments. These skilled workers are able to solve this problem to a great extent by using their experience, and calculators. However, the fish feed manufacturer wishes to not solely depend on these workers for a satisfying result. They want to make sure good decisions can be made even if these particular workers are absent, and replaced by i.e. seasonal workers.

4.1 Production and storing norms

Skretting has a dual production line in Stavanger which is able to produce up to 3,200 tonnes of fish feed per week, if run at full capacity. The produced feed is packed in large 750 kilogram big bags, and then transported to its slot and stored, by forklift. The majority of feed is shipped to customers by vessels, but some is also shipped by truck. On average there are three sea shipments per week, and while the feed is waiting to be shipped, it needs to be stored. For storing the feed, the company has a large storage facility with a capacity of roughly 4,600 tonnes. The entire storage facility is divided into separate sections, which again are divided in separate slots. The slots in a section all have the same storage capacity, but slots in neighbouring sections have different storage capacities. The sections are simply just different buildings and locations with varying free floor space.

The fish feed manufacturer uses a great variety of recipes when producing feed. The recipes differ in both ingredients and composition among other things (see Section *2.2.1*). Over time, the production of each recipe is divided into *batches*. The combination of a feed recipe and a batch is henceforth referred to as a (product) *type*. Recipes varies as to which fish specie the feed is meant for, together with factors like the size of the fish, if it is sick, and so on. Production batches are used to separate between feed of the same recipe, because they may have different expiry dates, and nutritional values. The nutritional values may vary between batches, as the ingredients and raw materials used in the production may have a different quality, based on its origin and batch. The origin changes due to import often being based on delivery dates and prices, which vary over time. The batches are given unique numbers which identify them, called *batch numbers*. When production of an order is finished, the batch number changes. This is because there is a new feed with a different recipe to be produced. If one would like to produce the same recipe multiple times over a time period, one would have the same amount of distinct batch numbers. Say for example feed of recipe A is produced, then feed of recipe B is produced, and then some more feed of recipe A again. Even though we produce feed A with the same recipe and ingredients twice, the nutritional values and detailed mixture may vary. We will therefore have three distinct batch numbers. If quality testing reveals that one batch of feed does not fulfill the requirements for nutrition or the composition/mixture is wrong, one can easily source the already stored feed by looking at the batch number.

The dual production line at Skretting Stavanger never produces the same sort of feed simultaneously. The entire production of an order is run on one line. Otherwise there would be two different batch numbers, as the raw materials are mixed in different tanks for the two production lines.

The fish feed manufacturer does not want to mix different types of feed. If some type of feed is stored at a slot, that slot is considered occupied, no matter how small the amount stored is. So if 5 tonnes are stored at a slot which can hold 80 tonnes, 80 tonnes of storage is considered to be occupied. Other types can not be stored in front, or on top of that. Consequently each slot can store at most one type at a time. Once that type is removed, it is free for any other type of feed to be stored there.

Skretting tries to run production on a first in, first out (FIFO) principle. This means that the feed which is being produced first, is most likely going to be shipped first. However, if several customers order the same type of feed, where some of the orders are not to be shipped right away, they might still be produced together. The orders not being shipped at first will have to be stored for a longer time, and production is in that case not FIFO. The reason for not producing FIFO is then to produce as large batches as possible, to be the most efficient. Changing production between types will most often require some work, and time. To ensure production does not switch between small volume orders too frequently, an order should be above 30 tonnes to be produced.

Production and shipping plans are finalized four days in advance. If the customers would like to change an order when it is less than four days to shipment, they need to call the order center at Skretting. The capacities of the storage slots are fixed, and do not change.

The dual production line in Stavanger does not produce all feed that is stored and shipped from there. Some of the feed is internal transfer from the two other plants in Norway. This is because the production line in Stavanger is not capable of producing all types of feed that they sell, so they are dependent upon having it shipped to Stavanger first. It might also be if one plant has too much feed stored, it is sent elsewhere to be sold, instead of letting it expire on due date. For simplicity, I will at times only mention shipped/produced feed, but this may also include feed removed/added to the storage as internal transfer.

4.2 Related work and published literature

To have a starting point, and to figure out what had previously been done regarding the storage of fish feed, I searched for existing literature. I used online search engines with combinations of keywords such as; "optimization", "storage", "fish", "feed", "technology", "program", "optimal", "of", "storing", "big", "bags", "materials". I know that my problem most likely is not distinct, and that other storages might have the same problem. Hence, I used the keywords to form searches targeting the storage of other materials as well. The searches did not give me any information of importance. The closest results found were mostly about under what conditions fish feed should be stored to retain their nutrients [6][18]. Other results described how to best fill big bags, which types of bags to choose, and under what conditions the bags should be stored to keep their attributes and properly protect their contents [21][2]. There were also some results about how to store big bags in containers and ships, to avoid unwanted incidents such as bags tipping over and moving around [9].

Chapter 5

Mathematical model

As the search for existing literature regarding related models gave little to no information, I have to start from scratch, and develop a model which does what Skretting is asking for, myself.

5.1 Sets, parameters and variables

Let I denote the set of slots at Skretting's storage facility. The types of feed that Skretting produce are denoted by the set J. Let T denote the number of periods in the planning horizon, and the set of time periods, also referred to as the time frame, is represented by the integer set $\{1, 2, \ldots, T\}$.

For each slot $i \in I$, denote by c_i the maximum tonnage of feed that, at any time, can be stored at i. We also refer to c_i as the *capacity* of slot i. Let u_{jt} denote the amount, for each feed $j \in J$, which is to be shipped in each time period $t = 1, \ldots, T$. For each feed $j \in J$, and time period $t = 1, \ldots, T$, let a_{jt} denote the amount of feed j which is to be added to the storage, in time period t. The added feed is either produced at the factory in Stavanger, or it arrives by sea shipment from another Skretting facility, as internal transfer.

For all $i \in I$, $j \in J$ and $t = 1, \ldots, T$, let x_{ijt} be a binary variable, where $x_{ijt} = 1$ if slot i is occupied by type j in period t, and $x_{ijt} = 0$ otherwise. Let s_{ijt} be a variable representing the amount of feed of type j which is stored at slot i in the beginning of time period t. The variable l_{ijt} represents the amount of feed of type j which is to be stored at slot i in time period t. This includes feed produced in Stavanger, as well as internal transfer between facilities. The variable h_{ijt} represents the amount of feed of type j, which the storage workers are to collect from slot i in time period t. The feed is collected to be shipped. Let d_{jt} be a variable which represents the amount of feed of type j to be transported directly from production, to the loading area in time period t. This rarely happens, but some times the ship or truck is waiting for feed that is currently being produced. In that case the feed does not enter the storage, but moves straight to loading.

5.2 Mixed integer linear programming model

The following mathematical model seeks to minimize the storage space occupied by fish feed on Skretting's premises in the last time period, T, of the time frame. The objective function sums over all storage slots i, which sums over all feed types j, and determines which placement of feed minimizes the storage space c_i in time period T. The mathematical notation is as follows,

$$\min \sum_{i \in I} \sum_{j \in J} c_i x_{ij(T+1)}.$$

Feed type j can be stored in slot i in period t only if the slot is occupied by j in period t. In that case, the amount s_{ijt} that is stored can be no more than c_i. Otherwise, s_{ijt} must be zero. Because allocation of slot i to type j in period t is equivalent to assigning the value 1 to x_{ijt}, the constraint

$$s_{ijt} \leq c_i x_{ijt} \quad i \in I,\ j \in J,\ t = 1, \ldots, (T+1)$$

applies. The amount u_{jt} of feed type j, shipped in time period t, has to equal the sum of the amount d_{jt} of direct transportation from production, and the total amount $\sum_{i \in I} h_{ijt}$ collected from storage slots. Hence,

$$u_{jt} = \sum_{i \in I} h_{ijt} + d_{jt} \quad j \in J,\ t = 1, \ldots, T.$$

In the same manner, the amount a_{jt} of feed, type j, which in time period t is added to the storage, has to equal the amount l_{ijt} which in the same time period t is placed at slot i. The amount d_{jt} which is directly transported from ship to storage as internal transfer has to be considered as well. It follows that,

$$a_{jt} = \sum_{i \in I} l_{ijt} + d_{jt} \quad j \in J,\ t = 1, \ldots, T.$$

The amount s_{ijt} of feed j stored at a slot i at time period t has to equal the amount $s_{ij(t-1)}$ that was stored there, minus the amount $h_{ij(t-1)}$ that was collected, and plus the amount $l_{ij(t-1)}$ that was put there in time period $(t-1)$. This gives the constraint

$$s_{ijt} = s_{ij(t-1)} - h_{ij(t-1)} + l_{ij(t-1)} \quad i \in I,\ j \in J,\ t = 2, \ldots, (T+1).$$

There can at any given time period t only be stored one type of feed j at a slot i. The constraint

$$\sum_{j \in J} x_{ijt} \leq 1 \quad i \in I,\ t = 1, \ldots, (T+1)$$

thus applies. At the start of a time frame $\{1, 2, \ldots, T\}$, the amount of feed j at a slot i has to equal what was initially placed there. Hence,

$$s_{ij1} = \bar{s}_{ij} \quad i \in I,\ j \in J.$$

At the start of a time frame $\{1, 2, \ldots, T\}$, the slots i occupied x_{ij1} by feed j has to equal the slots initially occupied. Hence,

$$x_{ij1} = \bar{x}_{ij} \quad i \in I,\ j \in J.$$

In summary the mixed integer programming model reads:

Sets:

- J – Types of feed
- I – Storage slots
- T – Time frame

Parameters:

- c_i – Storage capacity at slot i
- u_{jt} – Amount of feed of type j which is to be shipped in period t
- a_{jt} – Amount of feed of type j which is to be added (production and internal transfer) in period t

Variables:

- x_{ijt} – Equals 1 if slot i is occupied by type j, in the beginning of period t, equals 0 otherwise.
- s_{ijt} – Amount of feed of type j stored at slot i, in the beginning of period t.
- l_{ijt} – Amount of feed of type j, which is to be stored at slot i, in period t
- h_{ijt} – Amount of feed of type j, which is to be collected at slot i, in period t
- d_{jt} – Amount of feed of type j, which is to be transported directly from the production line to loading, in period t

Objective function:

$$\min \sum_{i \in I} \sum_{j \in J} c_i x_{ij(T+1)} \tag{1}$$

Constraints:

$$\begin{aligned}
s_{ijt} &\leq c_i x_{ijt} && i \in I,\ j \in J,\ t = 1, \ldots, (T+1) && (2)\\
u_{jt} &= \sum_{i \in I} h_{ijt} + d_{jt} && j \in J,\ t = 1, \ldots, T && (3)\\
a_{jt} &= \sum_{i \in I} l_{ijt} + d_{jt} && j \in J,\ t = 1, \ldots, T && (4)\\
s_{ijt} &= s_{ij(t-1)} - h_{ij(t-1)} + l_{ij(t-1)} && i \in I,\ j \in J,\ t = 2, \ldots, (T+1) && (5)\\
\sum_{j \in J} x_{ijt} &\leq 1 && i \in I,\ t = 1, \ldots, (T+1) && (6)\\
s_{ij1} &= \bar{s}_{ij} && i \in I,\ j \in J && (7)\\
x_{ij1} &= \bar{x}_{ij} && i \in I,\ j \in J && (8)\\
x_{ijt} &\in \{0, 1\} && i \in I,\ j \in J,\ t = 1, \ldots, (T+1) && (9)\\
s_{ijt}, l_{ijt}, h_{ijt}, d_{jt} &\geq 0 && i \in I,\ j \in J,\ t = 1, \ldots, (T+1) && (10)
\end{aligned}$$

5.3 Further development of the model

The model solved as above gives room for multiple optima. Multiple optima means that there are different placements of feed which all reach the same minimal objective value. This leaves room to consider other concerns regarding the placement of feed, which the fish feed producer might have. The minimal objective value reached by the model above is hereby referred to as the *primary objective value.*

5.3.1 Secondary objective function

In the early time periods in which the capacity is not minimized, the model is free to place feed at all feasible slots, as long as it does not affect the primary objective value negatively. The placements do not affect the primary objective value as long as the placed feed is shipped before the final time period T. This results in the model placing feed over multiple slots, instead of using as few as possible. The model might for example split an order of 80 tonnes on two slots capable of storing 50 and 100 tonnes, instead of placing it on one slot capable of storing 100 tonnes. To make sure the model does not do this, a secondary objective function is introduced. The new objective function minimizes the number of storage slots occupied in all time periods t, and is given by

$$\min \sum_{t=1}^{T} \sum_{i \in I} \sum_{j \in J} x_{ijt}.$$

This forces the model to always utilize as few storage slots as possible, and thereby no more storage slots than necessary are used. The result from the secondary objective function says how many slots are needed in the time frame $\{1, 2, \ldots, T\}$, and is referred to as the *secondary objective value.*

The reason for stopping the model from splitting the orders is simply that it is not realistic to do so. For the model to work, the workers need to follow the placements given by the model, however, splitting an order over several slots for no reason is not desired. The secondary objective function makes the placements more realistic.

Let V denote the optimal objective function value in model (1)-(12). Together with the new objective function, a new constraint is introduced.

The new constraint is (1) less than or equal to V. This new constraint makes sure the secondary objective function does not place feed in a manner which leads to less available storage space in the final time period than V. This is after all what is most important to Skretting. Hence,

$$\sum_{i \in I} \sum_{j \in J} c_i x_{ij(T+1)} \leq V. \tag{11}$$

The constraints (2)-(10) given above also apply to the model.

5.3.2 Tertiary objective function

As the primary and secondary objective functions are optimized, there is still room for multiple optima. This is a result of the storage having several slots with the same capacity. To help choose between the possible slots we now introduce a new parameter,

- p_i – Ranking of storage slot i.

Parameters p_i $(i \in I)$ are integers where a small numerical value indicates high rank. The ranking of slots are regarded as a function of how close they are located to the loading dock, as well as how accessible they are. A slot which is close to the dock is preferred because it makes loading more efficient. It is more efficient because it requires less driving, and less driving leads to the workers being able to sustain a good loading pace without the need for additional workers.

After the two first objective functions are solved, the model knows the amount s_{ijt} of feed j it is to place at slot i. The tertiary objective function considers these slots and selects the ones with the required capacity which are ranked highest in p_i. The new objective function is given by

$$\min \sum_{t=1}^{T+1} \sum_{i \in I} \sum_{j \in J} p_i s_{ijt}.$$

If the model, for example, determines that two slots of 80 tonnes capacity are needed, and there are five slots available, the tertiary objective function will choose the two slots which are ranked highest. If only one of the slots is filled to its capacity, the model will fill the slot with highest rank.

Again, we need a constraint which ensures the decisions made in the secondary objective function are respected, as that is considered more important. The new constraint makes sure the model does not occupy more slots i than necessary, given by the secondary objective value B. The constraint

$$\sum_{t=1}^{T}\sum_{i\in I}\sum_{j\in J} x_{ijt} \leq B \tag{12}$$

follows. The constraints (2)-(11) given above still apply.

5.3.3 Tightening constraints

After implementing the model in the language of AMPL, I run some initial tests of the model implementation on several data sets with a time constraint. I notice that there is a gap between the optimal solution found by the CPLEX® continuous relaxation, and the solution found by the integer solver. To minimize the gap I introduce three additional valid constraints which I believe can tighten the solution space. The new constraints are the following: From a storage slot i, it is not possible to collect a larger amount of feed j, than the amount s_{ijt} which is stored there. The constraint is written as

$$h_{ijt} \leq s_{ijt} \quad i \in I,\ j \in J,\ t = 1, \dots, T. \tag{13}$$

When storing feed type j at a slot i in time period t, the amount l_{ijt} put on storage there has to be no more than the capacity c_i of the slot, minus the amount s_{ijt} already stored there of the same type, from the period before. The constraint becomes

$$l_{ijt} \leq c_i - s_{ijt} \quad i \in I,\ j \in J,\ t = 1, \dots, T. \tag{14}$$

When storing feed type j at a slot i in time period t, the amount l_{ijt} which is stored there has to be no more than the capacity c_i of the slot, if the slot j is occupied in the next time period, that is, if $x_{ij(t+1)} = 1$. Otherwise it has to be zero. The constraint becomes

$$l_{ijt} \leq c_i x_{ij(t+1)} \quad i \in I,\ j \in J,\ t = 1, \dots, T. \tag{15}$$

By inspecting the constraints, I can show redundancy of constraints (13)-(15), by proving that they are satisfied at integer optimum by constraints (1)-(10). To prove this I use the fact that the variables h_{ijt} and

l_{ijt} cannot be positive simultaneously. Skretting does not collect feed j at slot i in time period t, while storing the same feed at the same slot in the same time period. This practice is respected by the model. In instances where an optimal solution exists, there exists one optimal solution where at least one of the variables h_{ijt} and l_{ijt} is zero. To see this, consider an optimal solution where $h_{ijt} > 0$ and $l_{ijt} > 0$. An alternative solution with the same objective function value, is constructed by reducing h_{ijt} and l_{ijt} by $\delta = \min\{h_{ijt}, l_{ijt}\}$, and by increasing d_{ijt} by δ.

For (13) I first assume $h_{ij(t-1)} > 0$, hence constraint (5) reads $s_{ijt} = s_{ij(t-1)} - h_{ij(t-1)}$. Since constraint (10), $s_{ijt} \geq 0$, states that s cannot be less than 0, that implies $h_{ij(t-1)}$ always have to be less than or equal to $s_{ij(t-1)}$, and thereby fulfilling (13). If $h_{ij(t-1)} = 0$, constraint (5) reads $s_{ijt} = s_{ij(t-1)} + l_{ij(t-1)}$, which is also feasible since (2) prevents $l_{ij(t-1)}$ from becoming too large.

For showing that constraint (14) is redundant, I use constraints (1) and (5). In (1), $s_{ijt} \leq c_i x_{ijt}$, I replace s_{ijt} with $s_{ij(t-1)} - h_{ij(t-1)} + l_{ij(t-1)}$ from (5), which gives $s_{ij(t-1)} - h_{ij(t-1)} + l_{ij(t-1)} \leq c_i x_{ijt} \leq c_i$. If $l_{ij(t-1)} > 0$, then $h_{ij(t-1)} = 0$. I am left with the constraint $l_{ij(t-1)} \leq c_i - s_{ij(t-1)}$, after rewriting the inequality. A summary of how (1) and (5) imply (14) follows below:

$$\begin{aligned} s_{ijt} &\leq c_i x_{ijt} \\ s_{ij(t-1)} - h_{ij(t-1)} + l_{ij(t-1)} &\leq c_i x_{ijt} \leq c_i \\ l_{ij(t-1)} &\leq c_i - s_{ij(t-1)}. \end{aligned}$$

Otherwise, if $l_{ij(t-1)} = 0$, (14) follows directly from (2). For constraint (15) if $l_{ij(t-1)} > 0$, $h_{ij(t-1)} = 0$ and (5) imply $s_{ijt} = s_{ij(t-1)} + l_{ij(t-1)}$. Since (1) states $s_{ijt} \leq c_i x_{ijt}$, this implies $s_{ij(t-1)} + l_{ij(t-1)} \leq c_i x_{ijt}$. Constraint (15) then follows from $s_{ij(t-1)} \geq 0$. On the other hand if $h_{ij(t-1)} > 0$ and $l_{ij(t-1)} = 0$, (5) is valid by (2), and the fact that $c_i \geq 0$ and $x_{ijt} \in \{0, 1\}$.

Chapter 6

Design choices

6.1 Rolling horizon time plan

For this problem I recommend that Skretting uses a rolling horizon time plan. In a rolling horizon time plan the model works with subsets of the time span, referred to by me as *time frames*, instead of the entire time span at once. When the model is finished working with the current time frame, it rolls on and works with the next time frame. The time frames are overlapping, so one time period is added, and one is deleted in each end of the time frame as the model rolls on (see *Figure 6.1*). The decisions the model propose after working with the current time frame are split into two subsets. One subset for the decisions which are guaranteed committed, and one subset for the decisions which the model suggests, but which will be reevaluated in the upcoming time frames. Rolling horizon time plans are often used if the parameters the model work with become increasingly uncertain after a while, if it is computationally expensive to work with the large amount of data that a long time span gives, or if one lacks data for the future [16].

If we look at Skretting's reality, the reasons for selecting a rolling horizon are definitely present:

- There is an uncertainty in production a_{jt}, and shipping plans u_{jt} at Skretting. As stated by the company, plans are not finalized until four days before they are due. So for every day passing, the production a_{jt}, and shipping plans u_{jt} gets updated by one day. Even though the fish feed producer states that the four-day plan is finalized, there might be changes to the finalized plan every once in a while.

By continuously running the model at each time period t, I ensure that any changes to the finalized plan are accounted for. This is in addition to the new information that comes with rolling past one time period t and expanding the horizon. Both factors might change the decisions supported by the model for the placement l_{ijt}, and in that case a new revised plan for the upcoming four days is generated.

- I can not be certain that the production ever stops. If the demand is high enough, Skretting runs their production every hour of the week. If so, the time span may stretch for months at a time, and then they always have to have the optimal plan in mind. They can not plan until a certain day, and then run the model again when that day arrives. If they do this they will already be behind on the planning, and might not be able to fit all the feed. This is because the model does not care about the feed produced after the time frame it has solved ends. This is further explained in the next section.

Hence, we apply a rolling horizon time plan, and make plans for one subset at a time.

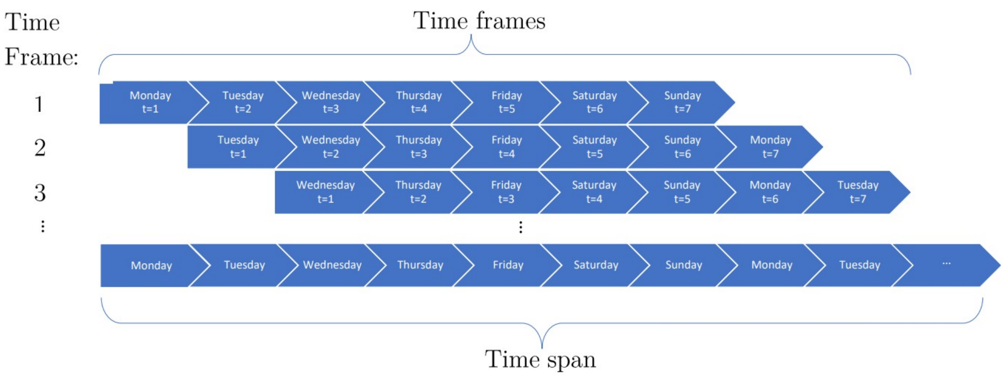


Figure 6.1: How the rolling horizon selects time frames in the example below.

We will now look at an example that resembles the fish feed producer's problem of placing feed at the optimal slots, as to save space. We start by assigning one time period t for each day, and we declare one time frame $\{1, \ldots, T\}$ as a seven-day working week. Operation starts on a Monday morning, $t = 1$, and the first time frame $t = 1, \ldots, T$ is until Sunday night, $T = 7$. The model is run, and this result tells where the feed produced until the end of the time frame should be stored, to have the most available storage space at the end of that time frame. However, the only supported decisions that are guaranteed committed, are the ones for the first time period $t = 1$. The decisions supported from Tuesday up until Sunday $t = 2, \dots, T$ only guarantee that everything will fit at the storage, and belong to the set of proposed decisions. When Monday is over, the horizon rolls one time period t on, and the next time frame is declared. The model is now able to plan one more day of production a_{jt}, and orders u_{jt}. The new time frame is Tuesday morning, $t = 1$, until Monday night, $T = 7$. The model is run again, and makes calculations as of what will be the optimal placement of feed by the end of Monday. From the set of proposed placements we had suggestions as of what were the optimal placements, however these are now ignored since there is more information available to base the decisions upon. The model commits the decision supported for Tuesday, as that is the current time period $t = 1$. The horizon rolls on, and the model continues to make predictions for the rest of the time frames.

Decisions supported by the model are partitioned into two sets – decisions that are committed by the model, and those that are not. The variables representing decisions in the first category correspond to the first time period t in the time frame $t = 1, \dots, T$. That is, all decisions corresponding to the first time period are fixed once the model is run, whereas the others will be fixed after future model runs. Decisions that are not committed needs to be a part of the model, as they represent the current solution, in later time periods t. The variables for the later time periods t are not fixated, but ensure that the decisions made does not cause an infeasible solution. An infeasible solution would lead to Skretting not being able to fit all the feed at the storage. This is the reason for the subscript t, for the variables x_{ijt}, s_{ijt}, l_{ijt}, h_{ijt} and d_{jt}. The model sets values for the variables in the upcoming time periods t, to see what will be a good solution. The decisions are based on the information available at the current time.

6.2 Choice of primary objective function

The model is run in the beginning of each time frame, and the primary objective function seeks to minimize the amount of occupied storage space at the end of the current time frame. Skretting's goal is to ensure there is always room to store all feed at the storage. This is possible only if the constraints, which state that the slots can not store more feed than there is room for, are fulfilled. So as long as a feasible solution to (1)-(12) is found, it is demonstrated that all the produced feed can be stored in some slots. With the use of an objective function which minimizes the occupied space at the end of the planning horizon, we ensure that the fish feed producer has the best possible prerequisite for storing the feed produced later, and therefore fulfilling the constraint. The upcoming production is unknown, so by maximizing the availability of idle storage space by the end of the horizon, we maximize the likelihood that there is enough space.

Other options for solving the problem that I have considered are; minimize the maximal space used, and minimize the average space used, both during a time frame. These objective functions do not consider the crucial fact that there is more feed being produced later, which we have to do our absolute best to make sure there is room for. They are therefore deemed as not suited.

I also see minimizing the maximal/average space used as irrelevant, as constraints in the model prevents the model from storing more feed than the capacity of the storage. If the maximal or average happens to be high, that does not bother Skretting. As long as the model is able to fit everything, they are satisfied, and the storage plan is feasible.

6.3 Time resolution

We can not predict the exact time when a specific feed is loaded, and can therefore not predict when its slot is made free. A ship can take anywhere between 1 hour and a day to load, depending on the ship, crew, amount of feed and where the feed is placed at the storage. Loading usually starts in the very beginning of the work day, but that also depends on when the ships arrive at port. By fine tuning the size of the time periods one is able to more precisely specify when shipments are loaded. In the example in Section *6.1*, one full day of production (*24 hours*) was used. This is not necessarily the best choice. By turning the size down to half a day (*12 hours*), or a quarter of a day (*6 hours*) for example, it is easier to predict when the feed has been loaded and the slots once again are free to use. There are two ways of approaching the problem as of when slots are considered free for further use. This is either as an aggressive, or as a passive approach.

In the aggressive approach the model does not distinguish between slots that are free in the beginning, and slots that are made free for use within a time period. The model proposes solutions where feed is to be placed in slots which in reality might not have been emptied yet. This will cause a conflict if the feed proposed to a slot is produced before the feed in that storage slot is shipped. By narrowing down the time periods, one is more accurately able to specify in which time period a slot is emptied. This helps fight the conflict, but does not remove it. The model still does not specifically know when a slot is free for use.

To totally overcome the recently mentioned conflict, one must choose to only mark slots as free, after one is certain that they have been emptied. That will be in the closest time period after they are due. This will lead to a more conservative model, as storage space might be marked as used when it has been emptied. However, the model will not propose conflicting solutions, such as the aggressive model might. A model running narrow time periods will prove to reduce the time a slot is falsely marked as occupied. If the model is run once per day, the slots freed for use are only updated once per day. This means that a slot can be free for the majority of the previous day, before the model is run, and recognizes the free slot. If a ship is loaded in the morning, the storage will have slots which have been made free, yet the model has to wait until the next day to use it.

The main reason for considering the aggressive placements, is that if the slots happen to be empty by the time the feed is produced, there are more slots to use. The objective values will as follows be better, than compared to the conservative model.

Longer time periods have the advantage of leading to fewer time periods, and therefore decreased size of the model. With decreased size follows reduced run time of the solver, which speeds up the process of finding the optimal solution. Finding a well suited period size is done by doing experimental studies. After doing several runs of the model implementation where I play with the time resolution, and take note of the effects it has on performance, I will compare the results.

Chapter 7

Experimental studies

To find out how the model and the data are to be set up for the best possible performance, I conduct experiments with the model in a rolling horizon.

7.1 Goals

In the experiments my goals are to:

- Determine if constraints (13)-(15) are to be included in the model or not.
- Experiment with the size of the time periods and observe their performance, to decide which size might be best.

The performance for the different sized time periods is determined by:

- How close the solution is to the optimum.
- How similar the objective values are.
- How fast the minimum gap requirement, or optimum is reached

7.2 Data set

To have as realistic experiments as possible, Skretting has supplied a data set containing all orders between January 2$^{\text{nd}}$ 2020 and February 28$^{\text{th}}$ 2020. The data set needs to be cleaned and formatted to be applicable to the model implementation. I write a Python script which is able to transform all data sets of the same format into the desired form. The script splits production and shipping orders into separate tables where the time period is placed on the index, and the feed type in the header. A row contains information about the amount all types of feed is produced/shipped in, in a time period. A column contains information regarding the amount of produced/shipped feed in all time periods, for one type of feed. Hence, to find the produced/shipped amount of a specific feed, in a given time period, one can look at the cell at the intersection of its column and row. In the Python script I am able to specify the time resolution to format by (see Section *6.3*), and the tables generated are saved as .txt-files on my disk.

In the supplied data set, each of the orders contain information about:

- The feed sort.
- If the feed is to be produced, sold or transferred internally between the factories.
- The amount of feed.
- The start time for the order.

The start time specifies when the order is due to be produced or shipped. The amount of time it takes to handle an order is not specified, but it is needed in case an order is not finished within the time period it is started, and stretches over multiple time periods. To find the time it takes to handle each order, I first sort the orders which are due to be shipped out, by start time. I then set the end time of each order as the start time of the following order. By having the start and end time of each order, I am able to check if they span across multiple time periods. If that happens to be the case, I distribute the correct percentage-wise amount of feed produced or shipped to the storage in those time periods accordingly. Let's say that we have 6-hour time periods, and the current time period started at 09:00. There is at midday started an order of 100 tonnes, and it is due to end at midnight. The correct percentage wise distribution of the order will then be to distribute 25% (tonnes) in the current time period, 50% (tonnes) in the time period which lasts from 15:00 until 21:00, and 25% (tonnes) in the time period that starts at 21:00 and ends at 03:00.

The same is done for the orders being produced, even though it is not as simple. There are two production lines, and it is not specified in the supplied data which production line a feed is produced by. By using the same strategy as above when setting the end time, some of the feed might be added faster or slower, in theory. The sum of the feed being added to the storage will however be accurate, which is the most important.

To have different instances to compare against each other, I create three different instance collections. This is according to the discussion in Section *6.3*. As mentioned in Section *4.1*, Skretting would like to run the model implementation with a 4-day time frame. An instance is therefore considered as the time periods that cover all production or shipping orders in a given 4-day time frame. Instances and time frames are therefore used similarly. The model implementation solves each instance by itself, before rolling over to the following instance, as described in Section *6.1*. When using the different collections to run the implementation in a rolling horizon, the implementation rolls over as frequent as the size of the time resolution in the collection, for a total amount of times as there are instances. To cover all available time within the full 57-day time span for the experimental studies, the collections look like:

- The first collection has 213 instances with 16 time periods, each of 6 hour length.
- The second collection has 107 instances with 8 time periods, each of 12 hour length.
- The third collection has 54 instances with 4 time periods, each of 24 hour length.

The reason that I decide to experiment with these three sizes is because I see them as a good fit for a 4-day time frame, and that they are easily compared against each other as both 6 and 12 divide 24. When making comparisons I look at every 24-hour instance against every other 12-hour instance and every fourth 6-hour instance. So when I look at the 2nd 24-hour instance I will have to compare it against the 4th 12-hour instance and the 8th 6-hour instance. That way I can be sure that the comparisons

I am making are made on the same 4-day time frame, which includes the same orders. Even though there are other numbers which divide 24 evenly, I think that having to run the model implementation more often than every 6 hours will be too tiresome. When comparing the same 4-day time frames against each other, I have 54 full time frames, and therefore 54 comparable instances.

7.3 Constant parameters

In the supplied data there are 119 different sorts of feed which are handled. Hence, I operate with this as a constant factor. If there are to be introduced or cancelled any sorts after February 28th, this will have to be changed. There are 88 storage slots at Skretting's premises in Stavanger. The slots range from a capacity of 4,500 kilos up to 153,000 kilos. Ranking of the slots (see Section *5.3.2* on the tertiary objective function) varies from 1 as the best, up until 50 for the worst. Some slots have the same ranking, as they are located side-by-side and are considered equally good.

7.4 Initial runs

The goal of the first experiment is to see whether the new constraints from Section *5.3.3* have a tightening effect, or not. I run the original model (1)-(12), and the new model (1)-(15) as continuous relaxations on 60, 12-hour instances, obtained from the supplied data. A continuous relaxation is obtained by discarding the integer constraint for the binary variables, and instead allowing solutions between 0 and 1 for the same variables. The reason that I run continuous relaxations on both models is to compare them and observe if the new constraints give me higher optimal objective values. If they do, they have a tightening effect, otherwise, the new constraints might be considered unproductive. Even if the constraints should prove to be completely unproductive for all continuous relaxations I run, I have to be careful to draw a conclusion. There might exist instances where the constraints are productive, even if I do not find any.

When running the relaxations for the 60 instances I observe that the optimal answer to the primary objective function is identical in all instances. The secondary objective function in the new model is marginally better than in the original model, in most of the instances. This result further affects the optimal answer to the tertiary objective function, which differ between being higher or lower. The fact that the optimal answer to the secondary objective function is *better* after *adding* constraints when minimizing may seem paradoxical, but has a plausible explanation. By further inspection I see that the first time frame is the only time frame where the optimal values are identical in all three objective functions for both models. However, by studying the actual placements of feed, that is, the optimal values of s_{ijt} and x_{ijt} in both models, I see that the placements differ. This tells me that the model even after three minimizing objective functions has room for multiple optima. In the instances I have at hand, the new model happens to give placements in the first instance which, when applying a rolling horizon, proves to be slightly better for the secondary objective function in later time frames. In the tertiary objective function I select slots based on their rank, but several of the slots with the same capacity share the same rank. This is because they are positioned next to each other, and are considered equal. This leads to several optima.

After observing that the optimal primary objective value is identical in all 60 instances, that the original model is simpler and therefore easier to solve, as well as having proved that the new constraints are redundant, I decide to continue with the original model and ditch the new constraints. I can not know if the new model consequently will perform better in the secondary objective function, with different data sets. I prefer a model which is as fast, less complex and as general as possible. If I continue with the new model, one can argue that the model might be subject to over fitting.

7.5 Full runs

After deciding which constraints to include in the final model implementation, it is time to conduct full runs of the implementation with the 3 collections of time frames. Considering the complexity of the model, the size of the constant parameters, and size of the data, I decide to set a time limit of 15 minutes per time frame. The time is distributed in ascending order between the three objective functions. When the implementation is run with a specific time frame, the primary objective function is given 15 minutes to find a solution within a 1 percent distance from optimality (the lower bound). The lower bound found by the implementation is the lowest value found by the branch-and-bound algorithm when the implementation terminates – either because the time runs out, or the gap is considered sufficiently good enough. The gap is considered good enough if the objective function value is less than 1% from the lower bound. The gap is calculated by the formula:

$$gap = \frac{objective\,function\,value - lower\,bound}{objective\,function\,value}.$$

Hence, if the model finds a solution within a 1% gap, the remaining time is given to the secondary objective function, and the same applies to this objective function, before the tertiary objective function eventually is distributed any time. If an objective function use all supplied time while trying to find a sufficiently good solution, the solution found by this function is passed on to the less important objective function(s), and the lower prioritized objective function(s) does not receive any time to try and improve the solution.

I consider 15 minutes a good amount of time for the model implementation to find as good of a solution as possible. It gives a balance between being quick, while still supplying a decent amount of time to find a sufficiently good solution. If 6 hours should prove to be the best option for the time resolution, I think it will be less than optimal to run the model implementation for any more than 15 minutes every 6 hours. If the implementation is run during a break, or once one arrives at work, the 15 minutes seems perfect to use while perhaps drinking a coffee or while getting ready for the shift.

For the full runs the model implementation encounter some difficulties, for all three collections. The issue is that the model implementation is not able to find an integer solution within the 15 minutes available, for a few of the time frames. As a result, the implementation is not able to ship out, or place any feed at the storage. When the implementation then tries to roll over to the next time frame, there is either too little, or too much feed at storage of some sort(s), and the implementation terminates with an error message. By instead stopping the rolling horizon, and resetting the storage in this time frame, the rolling horizon and implementation can continue. The storage is reset by erasing everything the model implementation has placed in all slots, and then adding all orders which are to be shipped in later time periods, but which are not scheduled to be produced or transferred internally, to the storage. For the three collections, the difficulties occur at:

- 6-hour collection: 43rd and 210th time frame
- 12-hour collection: 32nd and 86th time frame
- 24-hour collection: 43rd time frame

Hence, it seems like the time frames above are to much for the model implementation to handle within the given 15 minute time bound, but once the implementation gets some help, it is able to finish the rolling horizon. This might be the result of poor decisions made by the implementation from earlier time frames, or simply that the storage is full, and does not have room for any more feed. I do try to solve the issue by increasing the 15 minute time bound, and the model implementation is in fact able to place the feed. This means that there is room for more feed, but the implementation can not find a feasible solution within the given time. This is also confirmed by Figures 7.4, 7.5 and 7.6, where there are recorded time frames where the objective values are greater, than at the time frames where the implementation can not find a feasible integer solution. Hence, there is more available storage space to be used.

After running the model implementation on all three collections of data sets, there are large amounts of data to be analyzed, as to find out which time resolution is to be preferred. From experience I know that studying plots are a great way to compare trends, and results. Using this knowledge, I register relevant output from the implementation in a spreadsheet, and create suitable plots which I now will examine. The plots only contain the 54 comparable instances, and not all instances solved (see Section *7.2* for further information).

7.5.1 Computational burden

To evaluate the computational burden, I create plots which show how much time each objective function is distributed by the model implementation. In the plots, the primary objective function starts from 0 seconds, and continues until it has reached its desired 1% gap, or until the time runs out. If the gap is reached, the secondary objective function immediately takes over, and again, the same counts for this objective function, before the remaining time is eventually passed to the tertiary objective function. This is represented by bar charts, where for every instance, the time each objective function use is stacked on top of the previous, for a total of 15 minutes (900 seconds). If there is an instance where an objective function is not represented in the bar, then that objective function was not distributed any time by the model implementation. If there is space above a bar, and all three objective functions are present in the bar, that instance has been solved to within the desired gap for all three objective functions.

6-hour collection

The 6-hour collection is the most detailed, and details are desired because they make it easier to specify when an order is due. However, increased details also mean increased instance size and computational burden. The 6-hour collection is therefore expected to have the highest computational burden among the three collections(see Section *6.3*).

Out of the 54 comparable time frames, we can from Figure 7.1 observe that in the first 22 time frames the implementation spends most of the 15 available minutes to optimize the primary objective function. The implementation is only able to find a sufficient solution for all three objective functions in instance 2 and 9. Furthermore, it is able to find a sufficient solution for the two first objective functions in instance 16 and 3. The primary objective function is solved to within the desired gap in instance 1, 10, 17 and 18. As mentioned earlier, this can be determined by observing that the secondary objective function takes over for the primary objective function. Between time frame 22 and 33 the model implementation is able to find sufficient solutions for all three objective functions, as the full 15 minutes are not spent. From time frame 33 and until the end of the data the majority of the primary objective functions are solved to within the 1% gap. Some of the time frames also see a sufficiently good gap for the secondary objective value, and in 4 time frames the implementation finds a sufficient gap for all three objective functions. The entire time span has the following stats:

- Amount of time frames where all three objective functions are solved to within the gap: 16
- Amount of time frames where only the two first objective functions are solved to within the gap: 4
- Amount of time frames where only the first objective function is solved to within the gap: 11

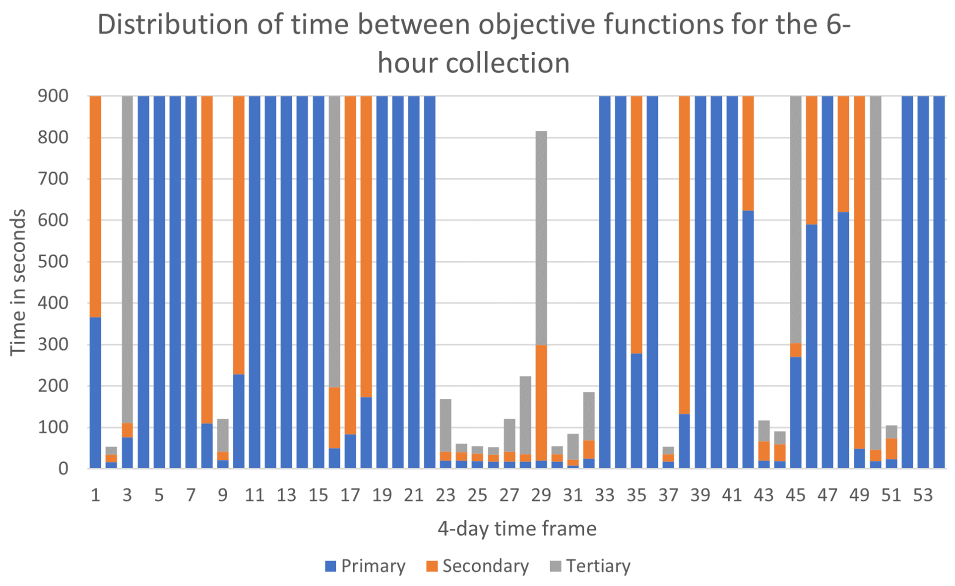


Figure 7.1

12-hour collection

In this collection each 4-day time frame is divided into 8, 12-hour periods and should therefore have a lower computational burden than the 6-hour collection. This is a result of the problem instance size being smaller. Figure 7.2 shows how the time is distributed by the model implementation for this collection.

We can observe that there seems to be a connection between which time frames are easy to solve, and which are hard, in this Figure and Figure 7.1. These may be instances where the storage might be close to full/empty, and it is therefore hard/easy to find sufficiently good solutions for the model. However, at first glance, there looks to be more time frames which are solved to within the desired gap, for Figure 7.2 in general. From time frame 1 until 22 there are 5 time frames with solutions below the 1% gap for all three objective functions, 3 time frames where only the two first objective functions have sufficiently good solutions and 5 where only the primary gap is fulfilled. For this collection, the span of solved objective functions in the middle is expanded by one, to stretch from the 23rd to the 33rd time frame, compared to the previous collection. From time frame 33 until the end of data, the trend continues for the most part. There is one more time frame solved below the acceptable gap, but some of the other time frames are able to distribute more of the given time to the secondary or tertiary objective functions. This means that they are easier to solve, and that the model implementation is closer to the desired gap than it was in the previous collection. There are however a few time frames where the implementation is further away from desired solutions for all three objective functions. The stats from this collection are summarized below:

- Amount of time frames where all three objective functions are solved to within the gap: 22
- Amount of time frames where only the two first objective functions are solved to within the gap: 7
- Amount of time frames where only the first objective function is solved to within the gap: 9

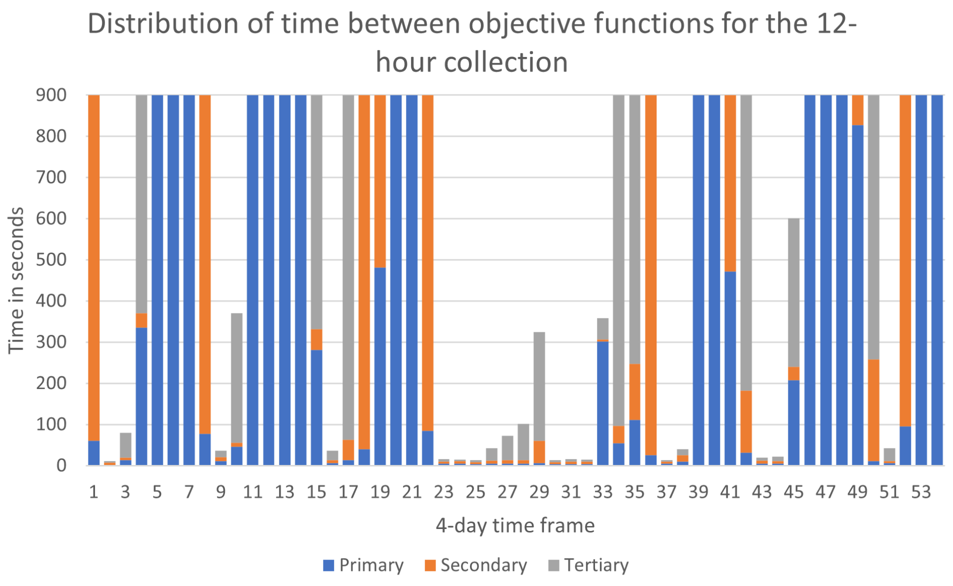


Figure 7.2

24-hour collection

The 54 time frames in the 24-hour collection is predicted to have the smallest computational burden, as it only has 4 time periods per time frame. Figure 7.3 shows how the distributed time was spent by the model implementation.

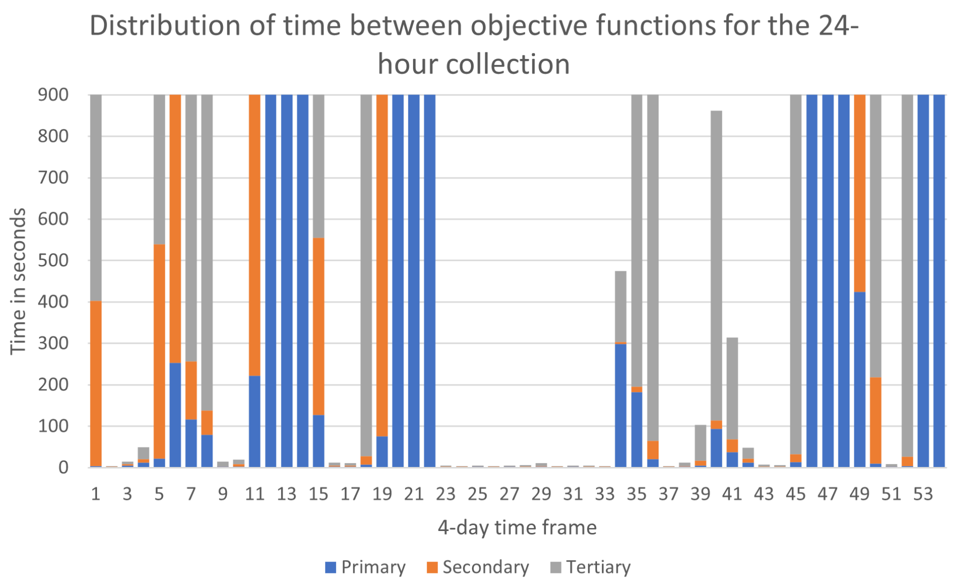


Figure 7.3

The most noticeable fact in this Figure is that there are fewer time frames which need the full 15 minutes. Compared to Figure 7.2 there are two more time frames solved to within the desired gap in the first 23 instances. However, some of the other instances which are not fully solved, also see an increase in time left for the secondary and tertiary objective functions. The 22nd instance is however not able to find a sufficient solution for the primary objective function anymore, as it was able to in the previously analyzed collection. In the middle part of the time span there is an additional instance solved to within the desired gap, compared to the previous collection analyzed. This is the 34th instance. For the remaining instances the saga continues, and the 24-hour collection proves to be better at finding solutions within the desired gap, across the range of objective functions. The observations are summarized below:

- Amount of time frames where all three objective functions are solved to within the gap: 28
- Amount of time frames where only the two first objective functions are solved to within the gap: 11

- Amount of time frames where only the first objective function is solved to within the gap: 4

7.5.2 Objective function values

Now that we are familiar with how the model implementation distributes the time between the objective functions for all instances in the three collections, it is time to look at how the objective function values compare against each other. In the instances where the storage is reset, there appears a dot in the same colour as the collection which is reset.

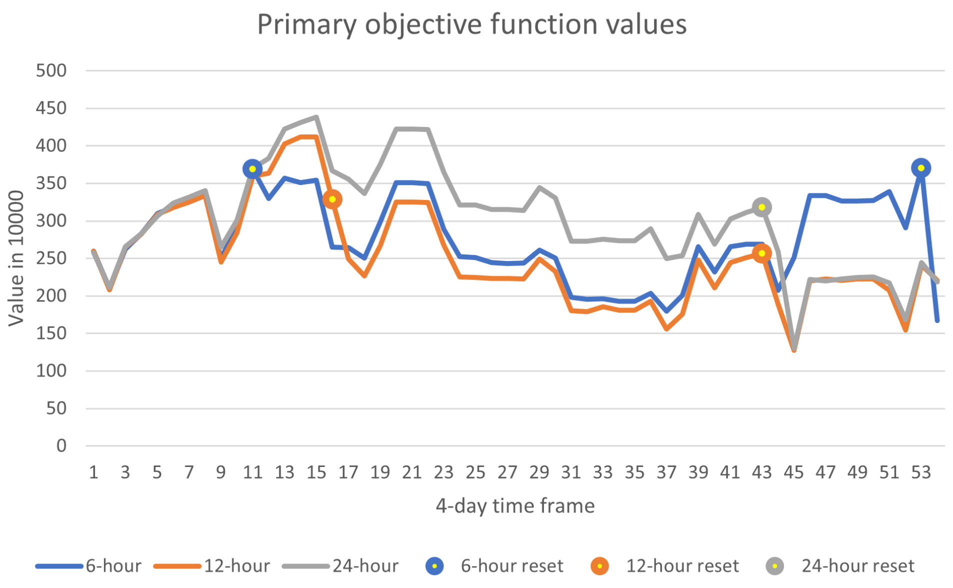


Figure 7.4: The primary objective values for the three time resolutions.

Primary objective function

Figure 7.4 shows how the primary objective function values differ between the three collections. One can observe that all three collections follow each other closely for the first 11 instances. From instance 11 and onward the 24-hour collection consequently has a higher objective function value. For the two other collections, the 6-hour collection starts out by being lower in value than the 12-hour collection, but at instance 17 the two smallest collections switch between having the lowest objective function value. This continues until instance 43 where the 24-hour collection suddenly dives down and joins the 12-hour collection as the lowest objective function value. At almost the same time the objective function value for the 6-hour collection increases noticeably. This order continues until the 52$^{\text{nd}}$ instance when the 6-hour collection plunges down and finishes with the lowest objective function value. These sudden changes of direction can be related to resetting the storage at the instances where the model implementation is not able to find an integer solution. For the 6-hour collection the storage is reset just before instance 12 and 54. In the latter instance, the plot can be seen deviating quite significantly from its previous path. For the 12-hour collection the storage is reset just before instance 17 and 44. In the first case on can argue that the objective function should have stayed on the same path as it was following together with the 24-hour collection, and not dive down as deep as it does. Both the 24-hour collection and the 12-hour collection is reset right before instance 44. They both see a significant decrease in their objective function values, and are most likely reset to have the exact same inventory at storage.

In Figure 7.7, one can see how the objective function values found by the model implementation for the three collections relate to their respective lower bounds found. As previously mentioned, the lower bound found by the model implementation is the lowest value found by the branch-and-bound algorithm when the implementation terminates, either because the time runs out, or because the gap is considered sufficiently good enough. In there is no time remaining for the secondary or tertiary objective functions, the model implementation solves the continuous relaxation and uses that as the lower bound.

The reason that it is interesting to see whether the objective function values found match the lower bound closely, is to see whether the values are close to a potential optimal solution. This says something about the quality of the solution. The closer the gap, the higher the quality of the solution. The primary objective function is the most important, so it uses either the full 15 minutes, or it reaches the 1% gap. In all three instance collections it is easy to see that the solutions found by the model implementation are close to the lower bounds. Where the gap is at its largest, we can compare with Figures 7.1, 7.2 and 7.3, and observe that the implementation spent all available time prioritizing to find a good primary objective function value in these instances.

As the computational burden decrease with fewer time periods in the instances, it makes sense that the 24-hour collection has more instances solved close to optimality than the two other time resolutions. The average gaps for the 54 comparable data points for the primary objective function are given below:

- the 6-hour collection: 2,59%
- the 12-hour collection: 1,83%
- the 24-hour collection: 1,10%

Secondary objective function

For the secondary objective function we can in Figure 7.5 see that the secondary objective function values do not follow each other as closely as the primary objective function values did in Figure 7.4.

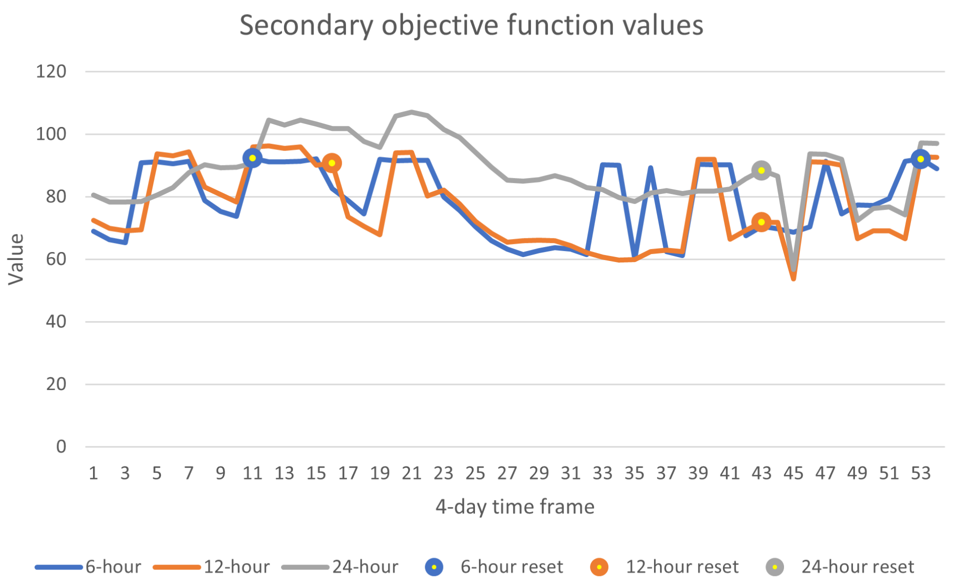


Figure 7.5: The secondary objective values for the three time resolutions.

All three instance collections switch multiple times between having the lowest objective function value. For the 6-hour instances one can observe that re-setting the storage does not have an impact as great for the secondary objective function as it did for the primary objective function.

On the contrary, it does have an easy noticeable effect on the 12-hour and 24-hour collections.

Figure 7.8 reveals that there are a great amount of time frames where the model implementation spends all available time solving the primary objective function. The sum of gaps seem to be a lot bigger in these plots, than in Figure 7.7. This is confirmed by Figures 7.1, 7.2 and 7.3 where we can observe that the secondary objective functions are distributed less time to find a sufficient solution than the primary objective functions. Where there are large gaps the implementation does not have any time to try to find a good solution, and instead only uses the solution handed down the hierarchy from the primary objective function. The average gaps for the 54 comparable instances, where the model implementation uses the three collections to solve the secondary objective function are:

- the 6-hour collection: 16,66%
- the 12-hour collection: 12,37%
- the 24-hour collection: 7,38%

One can observe that the average gaps are not as low as for the primary objective function, but that is as expected.

Tertiary objective function

The three tertiary objective functions from Figure 7.6 follow the same general trend, but the three collections switch between having the lowest objective function value. It seems like the 6-hour collection has the lowest value for the majority of time frames. For the 6-hour collection the storage is reset just before time frame 12 and 54. Just as in the secondary objective function, the tertiary objective function value is greatly affected after the reset before time frame 54. The changes made at the storage for the 12-hour collection in time frame 17 and 44, also affect the tertiary objective value in which it becomes lower. For the 24-hour collection the reset just before time frame 44 also decrease the objective function value.

When looking at Figure 7.9 one can observe that the model implementation did most likely not spend much time solving the tertiary objective function, and instead used the solution which was passed down from the

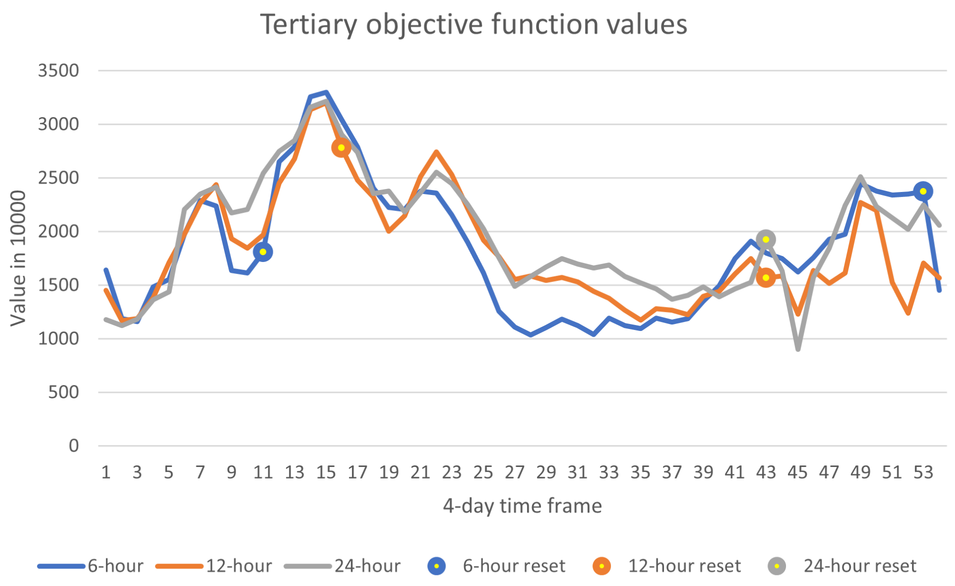
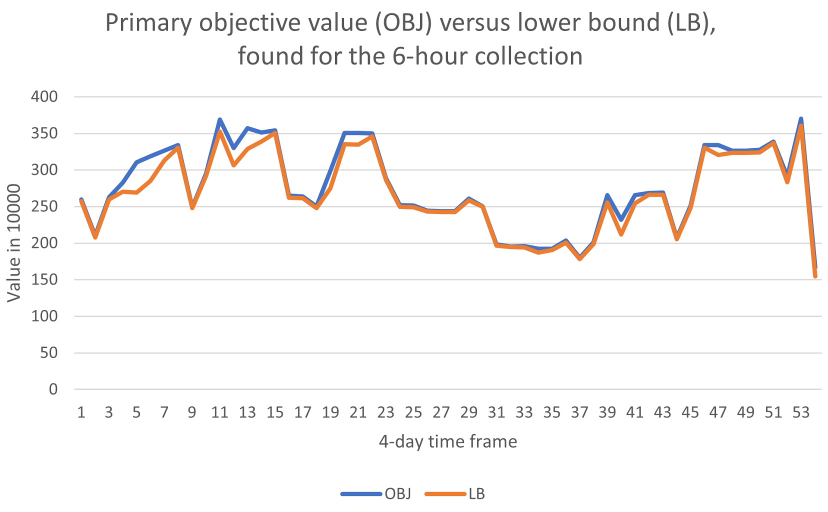


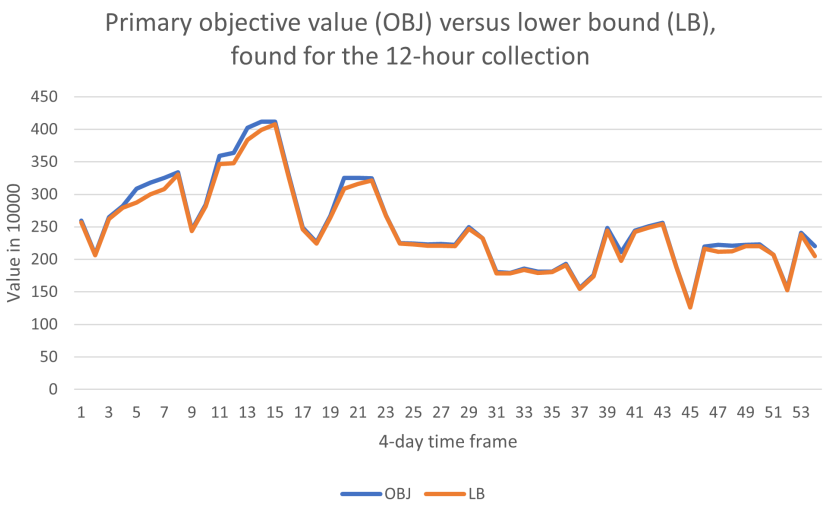
Figure 7.6: The tertiary objective values for the three time resolutions.

secondary objective function. Again, this is confirmed by examining Figures 7.1, 7.2 and 7.3. Following the trend from the primary and secondary objective functions we can observe that the fine time resolution struggles to follow the lower bound to a greater extent than the coarse time resolution, as they are harder to solve. The mentioned observations are confirmed by examining the average gaps for the 54 comparable instances, where the model implementation uses the three collections to solve the tertiary objective function:

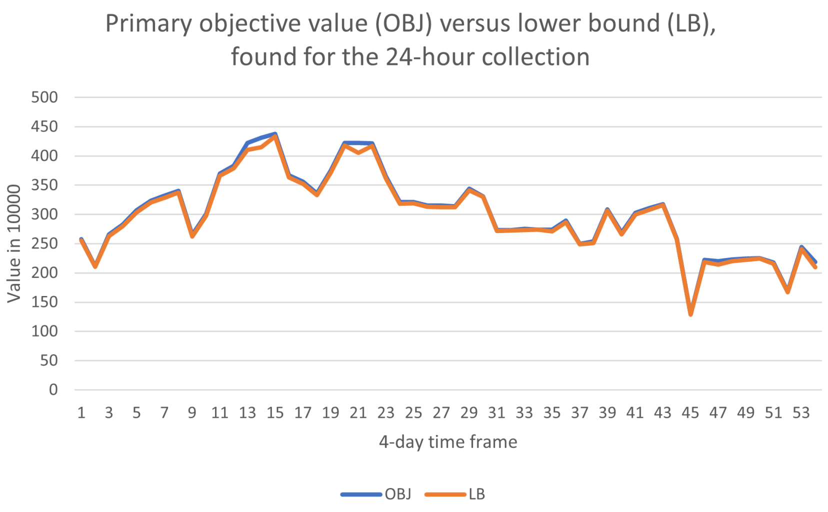
- the 6-hour collection: 19,10%
- the 12-hour collection: 17,24%
- the 24-hour collection: 14,09%



(a)

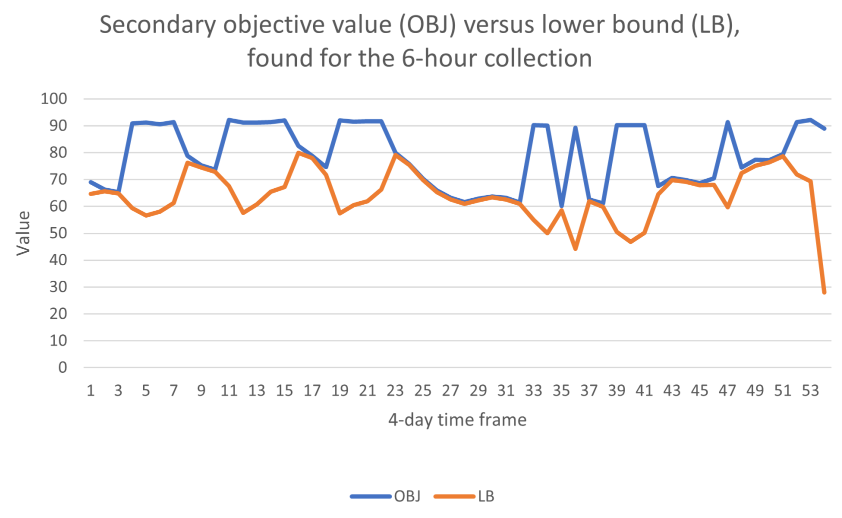


(b)

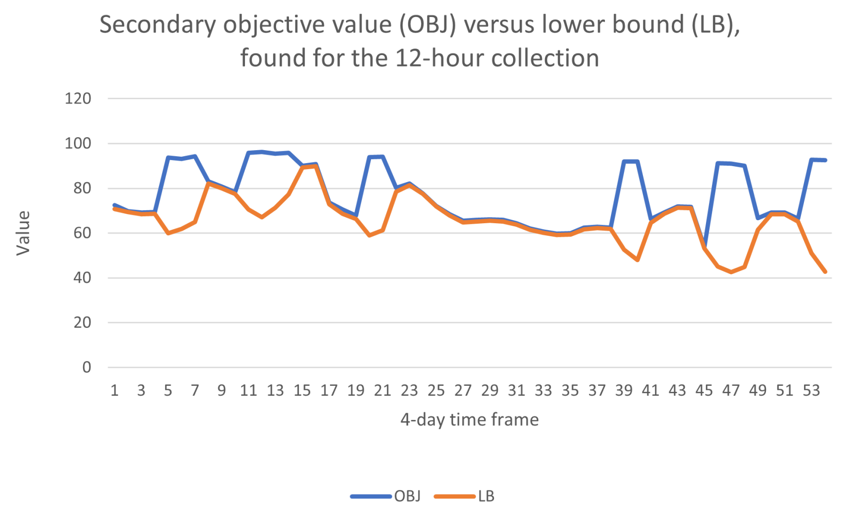


(c)

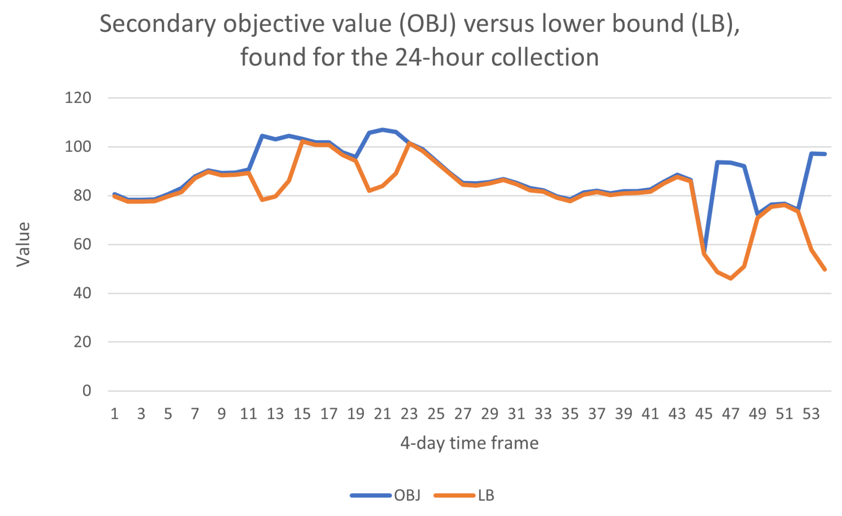
Figure 7.7



(a)

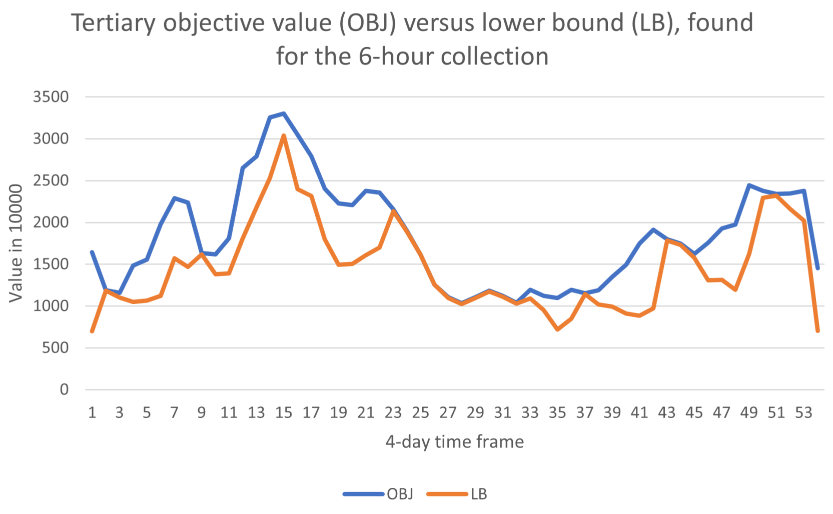


(b)

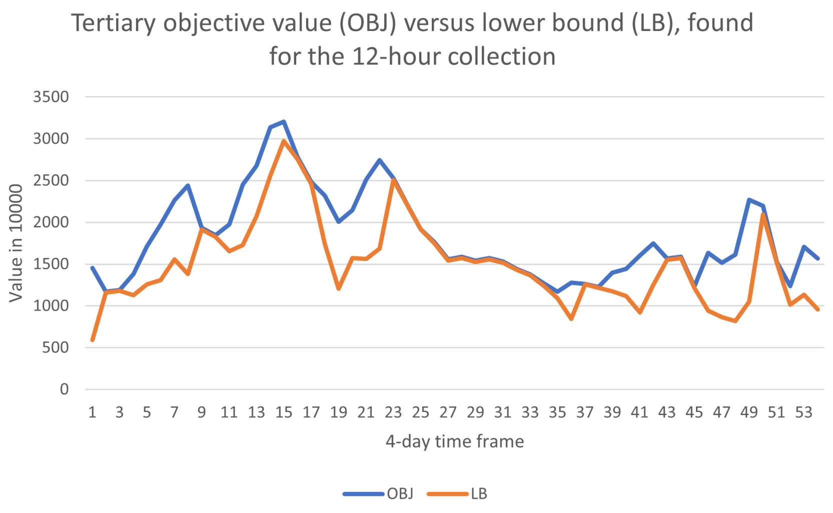


(c)

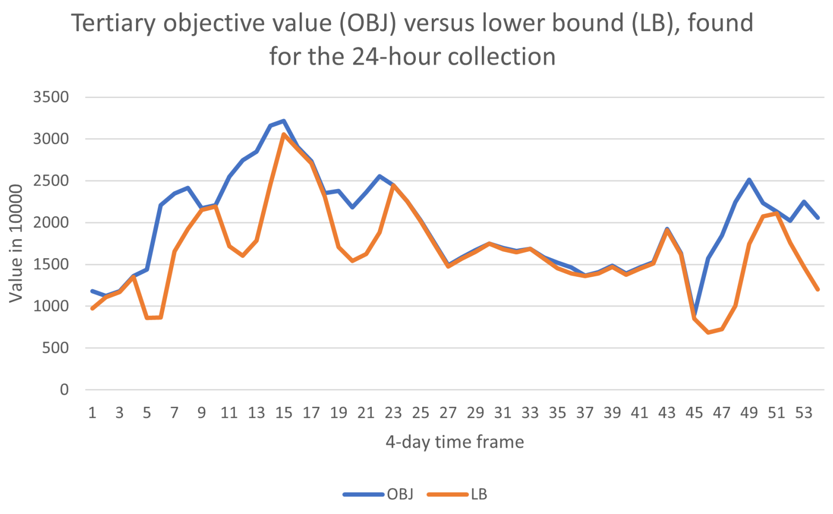
Figure 7.8



(a)



(b)



(c)

Figure 7.9

Chapter 8

Discussion

8.1 Intention of use

For this Master's thesis in optimization, I have developed a model implementation which tries to find the best possible storage placements of 750 kilogram big bags of fish feed, at Skretting's storage facility in Stavanger. The best possible placements are regarded as the placements which leave the most amount of storage space available, at the end of a 4-day production plan. The production quantity which comes after this is unknown, hence it is important to keep as much storage space as possible free for upcoming production. Multiple sorts of feed are produced by Skretting, and feed of different sorts, or batches, can not be stored together. If there are multiple placements of feed which all reach the same minimum amount of storage space used, the implementation selects which slots to fill first based on which slots are regarded best. The model implementation is meant to make it easier for Skretting's employees to decide where it is best to store the feed. As of now, this is done manually by experienced workers. With the help of this model implementation, good decisions can be made even when the more experienced workers are absent. This is especially important during the summer, when regular staff may be on vacation and substitutes are in their place. The workers will also be saved from the labour of walking around the storage trying to solve the puzzle of making everything fit. This labour can instead be used elsewhere.

8.1.1 Additional usage

By adapting the constant parameters and data, the model implementation can be directly applicable to other problems where one minimize the amount of storage space used, by products of uniform size. Furthermore, with minor extensions to the implementation, it can be applied to storages where one for example is interested in:

- keeping the average storage space used to the minimum.
- keeping the maximum storage space used to the minimum.
- letting one type of product be stored in front of another.
- letting the model implementation store the products in bulk, like in silos, instead of in entire bags.

In short, with just small changes to the parameters or to the implementation, the opportunities are many.

8.2 Implementation

To propose placements of feed, I have developed a mixed integer linear programming model, which I have implemented in the algebraic modeling language of AMPL (A Mathematical Programming Language). The model implementation is multi-periodic, and utilizes a rolling horizon for when it moves from one time frame on to the next. To find a solution as good as possible for each time frame, the model implementation uses three separate objective functions arranged in a hierarchical order. This means that there might be multiple solutions all yielding the same optimal value after the first, and most important objective function has been solved to within a sufficient gap. Hence, the implementation tries to find a preferred solution among these, by minimizing the second objective function. Still, there might be multiple solutions to choose from, and by minimizing the third objective function, the implementation might find a preferred solution among these. The three objective functions have the following objectives:

- For the first objective function, the model implementation minimizes the amount of occupied storage space at the end of the time frame.

- For the second objective function, the model implementation minimizes the amount of storage slots used during the entire time frame.

- For the third objective function, the model implementation selects the best ranked slots first, if the implementation has multiple slots of the same capacity to choose from.

A solution is considered sufficiently good enough if the gap between the lower bound, and the current objective function value found by the branch-and-bound algorithm is below 1%. If the gap is fulfilled, the remaining time that the model implementation does not use for the objective function in question, is distributed to solve the objective function below in the hierarchy. The model implementation has a time cap of 15 minutes to solve all three objective functions. The implementation is run three separate times – once per collection of time frames. The collections differ by having time resolutions of either 6, 12 or 24. The goal for the three separate runs is to determine which time resolution is to be used, to achieve the best result.

8.3 Results

Coming to a conclusion as to which time resolution is to be preferred, is not a simple task. It is pretty clear that the 24-hour collection has the average smallest gaps, and uses the least amount of time, but it does not necessarily give the most accurate placements of feed. It has the roughest time resolution, and one can observe in several of the Figures how its curve has been smoothened out, by avoiding some of the sharp turns that the 6-, and 12-hour time resolutions make. At the sharp turns the storage might see a sudden increase in stored/shipped feed, and as a result it might be difficult/easy to store additional feed. The reasons that the 24-hour time resolution, to a greater extent avoids these, are addressed in the following paragraph.

There are a few undesired model features which can result in imprecise objective function values. The values can sway in both directions, and the longer the time period, the larger the possible imprecision. The reasons are listed in the bullet points below, and further explained in the two following paragraphs:

- Slots are made free in the beginning of the time period that their stored feed is due to be shipped.
- Slots are allocated (marked as occupied) in the beginning of the time period that their bound feed is due to be produced, or arrive as internal transfer.
- Orders which are due to be shipped shortly after they are produced, may be shipped directly, instead of placed at the storage. If so, the variable d_{jt} is assigned a positive value equal to the amount of feed j going straight from production to shipping, in time period t.

The model may consider slots available for up to 24 hours, when it in reality might be up to 24 hours before they are emptied. This can result in the model proposing feed placed in slots which has not been emptied yet – creating problems for the employees. The opposing fact, that slots can be set as occupied for up to 24 hours before there is actually placed any feed there, does not cause any trouble, but it is inefficient. While an empty slot is "falsely" marked as occupied, some other feed could be stored intermediately in its place, if that feed was to be moved or shipped out before the scheduled feed is due to be stored.

Intermediate storing could be useful if a feed is to be both produced and shipped during the same time period. In the model implementation this feed is shipped directly, and never put on storage. As an example, say there is an order which takes 3 hours to produce, and in 20 hours it is due to be shipped – which takes less than 1 hour. In this case the employees will have to make up where to store the feed for 20 hours, because the model has determined that it does not need to be stored. This can upset the whole storing process. If the time periods were smaller, the model would see that the order needed to be stored, and would try to find a suitable slot for it. This does not directly affect the primary objective function values in the 54 comparable time frames in the Figures, as the feed is shipped before the end of the time frame in all three collections. It does, however, affect the values if one were to examine the time frames in between the full 24-hour marks that are being compared. E.g. at the intermediate 6, 12 or 18 hours after any of the 54 time frames which are being compared. Furthermore, it affects the secondary and tertiary objective function values, as well as the usability of the model. The secondary and tertiary objective function values are affected as they are related to what happens before the end of the time frame, and when the model ships feed directly instead of using storage space for it, these values become lower. Such direct shipment is accomplished by assigning a positive value identical to the amount shipped, to variable d_{jt}. All in all, the greater the size of the time periods, the greater these three issues become. Hence, I consider the 24-hour time resolution to be the least relevant for this problem.

The main task at Skretting is to make sure everything fits at the storage, so the primary objective function is weighted heaviest. From Figure 7.4 I can observe that the objective function values from the two smallest time resolutions follow each other closely, except right after the respective storages are being reset. Their average gaps are also relatively close to each other. A low primary objective function value is not necessarily what is the best result in this model. A value which is low might be subject to the issues raised in the first and third bullet points mentioned above, where the model allocate feed to a slot which is already occupied, or does not store it at all. This may result in the model being able to obtain a value lower than what is realistic.

8.4 Further development

The fact that the model implementation using the new constraints gave better optimal values in the secondary objective function, when running the relaxation in Section *7.4*, tells me that there is still room for improving the model. This would require further analysis of new constraints, and is a task beyond the scope of this thesis.

If further extending the thesis was to be done, I would also have tried to make a model implementation where a slot cannot be used until the model knows that the slot has been made free. This could for example be accomplished with the use of a passive model such as the one proposed in Section *6.3*.

When the model implementation stops because no feasible integer solution can be found, the solver simply prints the error and terminates. As of now, the implementation has to be reset manually, and then be started over again. Effortlessly resetting the storage can only be done in simulations such as the ones performed in this thesis, as there is not any actual feed to be shuffled around. If the same problem happens at Skretting, when previous solutions already have been committed, it will be too much labour to rearrange the entire storage. I believe one solution to this problem could be to extend the implementation with an option for the workers to choose between either: to give the model implementation a longer time limit and see if it comes up with a feasible solution, or choose to move feed between (multiple) slots, and re-run the same instance. Moving feed would only make sense if the workers can see that bad decisions have previously been committed, and that freeing up one, or multiple slots, should help the implementation on its way to propose a feasible solution. In the case of the storage actually being full, the workers would have to see this by themselves, and stop production until there is room enough to continue production. Otherwise, the model implementation would just continue to terminate with no feasible integer solution.

As stated earlier, it is preferred by Skretting to store all feed due in the same shipping order on one slot, but as I have spent more time thinking about it, this might not necessarily be best. If the space available in a slot is to be maximized, the big bags are carefully stacked three in height. By spreading the feed out on multiple slots, the feed can perhaps be stacked one or two in height on certain slots instead. By not having to store three in height, the feed is easier and less time consuming to both store and collect. Also, if the feed is stored on multiple slots, several forklift drivers can collect feed at the same time instead of having to wait for each other in line. It would be interesting to examine if it is better to store all feed from one order on one slot, or instead spread it out on as many slots as possible, to make storing and collecting easier. This new way of storing would at all times use as much storage space as possible, so one would possibly need to have a greater prediction of the amount of feed to be stored in the near future. This might be a problem for some other thesis, as this thesis is about making sure that there is always as much available storage space as possible.

Chapter 9

Conclusion

I have in this thesis created a minimization model which fulfills the optimization needs at Skretting Stavanger. The model implementation can help them optimize the placement of feed at their storage. Computational experiments indicate that the model implementation using the 6-hour time resolution, without the additional constraints deliver the best results. This is supported by the fact that the additional constraints have been proven to be redundant, and that the solutions proposed using the 6-hour time periods are the most detailed. The limitations exposed by the three bullet points in Section *8.3* affects the model less if the time resolution is smaller. The increased accuracy achieved using the 6-hour collection, weighs up for the flaws in speed, compared to using the two other collections.

The majority of the comparable 6-hour instances are also solved to within the desired gap, for the primary objective function. The fact that the model implementation does not spend as much time to solve the secondary or tertiary objective functions does not concern me. The workers can easily override the decisions proposed by the model regarding which slots of a given size to select. They know which slots they prefer to store and collect from, as the best slots normally are the slots closest to the loading area.

The next steps for this model would be to compare how the placements proposed by the model compare to the placements committed by the workers. If the results prove to be good, the model implementation can be considered developed into a user friendly application.

References

[1] Global Aquaculture Alliance. *What Is Aquaculture and Why Do We Need It?* URL: https://www.aquaculturealliance.org/blog/what-is-aquaculture-why-do-we-need-it/. (accessed: 12.10.2020).

[2] Samuel Amponsah. *TRAINING MANUAL ON FISH FEED STORAGE AND HANDLING*. URL: https://www.researchgate.net/publication/321836110_TRAINING_MANUAL_ON_FISH_FEED_STORAGE_AND_HANDLING. (accessed: 03.07.2021).

[3] Skretting AS. *Company facts*. URL: https://www.skretting.com/en/this-is-skretting/company-facts/. (accessed: 22.06.2021).

[4] Skretting AS. *Our history*. URL: https://www.skretting.com/en/this-is-skretting/our-history/. (accessed: 22.06.2021).

[5] Skretting AS. *What ingredients are in Skretting feed?* URL: https://www.skretting.com/en/transparency--trust/faqs/what-ingredients-are-in-skretting-feed/. (accessed: 03.10.2021).

[6] Global Feed Sector Authors. *Fish feed storage*. URL: https://www.allaboutfeed.net/home/fish-feed-storage/. (accessed: 03.07.2021).

[7] Claude E. Boyd. *Assessing the carbon footprint of aquaculture*. URL: https://www.globalseafood.org/advocate/assessing-carbon-footprint-of-aquaculture/. (accessed: 03.10.2021).

[8] European Commission. *A short history - aquaculture*. URL: https://ec.europa.eu/fisheries/cfp/aquaculture/aquaculture_methods/history_en. (accessed: 12.10.2020).

[9] Elebia. *Shipping of Big Bags*. URL: https://elebia.com/shipping-of-big-bags/. (accessed: 03.07.2021).

[10] fish-feed-extruder.com. *Extruded Aquafeed and Pelleted Feed.* URL: https://www.fish-feed-extruder.com/Application/extruded-aquafeed-and-pelleted-feed.html. (accessed: 23.07.2021).

[11] Derwent Group. *Pellets versus extruded feed.* URL: https://derwent.es/en/pellets-versus-extruded-feed/. (accessed: 03.10.2021).

[12] Washington State Department of Health. *Health Benefits of Fish.* URL: https://www.doh.wa.gov/communityandenvironment/food/fish/healthbenefits. (accessed: 27.07.2021).

[13] Shannon Hunt and Jude Isabella. *A Short History of Aquaculture Innovation.* URL: https://www.hakaimagazine.com/features/a-short-history-of-aquaculture-innovation/. (accessed: 19.10.2020).

[14] Svein Johansen. *Skretting i hundre 1899-1999.* Ed. by Lillian Høivik Finn Hallingstad Torleif Abrahamsen and Vidar Julien. Bryne Offset AS, 1999, pp. 8, 9, 25, 26, 29, 30, 32.

[15] Joel K. Bourne JR. *How to Farm a Better Fish.* URL: https://www.nationalgeographic.com/foodfeatures/aquaculture/. (accessed: 02.11.2020).

[16] Frauke Liers Lukas Glomb and Florian Røsel. *A rolling-horizon approach for multi-period optimization.* URL: http://www.optimization-online.org/DB_HTML/2020/05/7809.html.

[17] Greg Lutz. *Assessing the carbon footprint of aquaculture.* URL: https://thefishsite.com/articles/assessing-the-carbon-footprint-of-aquaculture. (accessed: 03.10.2021).

[18] Kanij Rukshana Sumi Nazmul Hossen and Tawheed Hasan. *Effect of Storage Time on Fish Feed Stored at Room Temperature and Low Temperature.* URL: https://www.researchgate.net/publication/261830658_Effect_of_Storage_Time_on_Fish_Feed_Stored_at_Room_Temperature_and_Low_Temperature. (accessed: 03.07.2021).

[19] NOAA. *What is aquaculture?* URL: https://oceanservice.noaa.gov/facts/aquaculture.html. (accessed: 12.10.2020).

[20] Constanze Pietsch. *Food Safety: The Risk of Mycotoxin Contamination in Fish.* URL: https://www.intechopen.com/chapters/68947. (accessed: 03.10.2021).

[21] Dave H.F. Robb and Viv O. Crampton. *On-farm feeding and feed management: perspectives from the fish feed industry*. URL: `http://www.fao.org/fishery/docs/CDrom/T583/root/19.pdf`. (accessed: 23.07.2021).

[22] SalMar. *Havbasert fiskeoppdrett*. URL: `https://www.salmar.no/havbasert-fiskeoppdrett-en-ny-aera/`. (accessed: 02.11.2020).

[23] seafood.no. *Sjømateksport for 107,3 milliarder kroner i 2019*. URL: `https://seafood.no/aktuelt/nyheter/sjomateksport-for-1073-milliarder-kroner-i-2019/`. (accessed: 02.11.2020).

[24] The Fish Site. *The fish feed story*. URL: `https://thefishsite.com/articles/the-fish-feed-story`. (accessed: 28.07.2021).

[25] The Fish Site. *The main ingredients of fish feed*. URL: `https://thefishsite.com/articles/the-main-ingredients-of-feed`. (accessed: 28.07.2021).

[26] SSB. *Utenrikshandel med varer*. URL: `https://www.ssb.no/utenriksokonomi/statistikker/muh/aar`. (accessed: 02.11.2020).