1ST OF MARCH 2024

OPTIMIZING THE SEQUENCE-DEPENDENT SETUP TIMES OF A MACHINE USING AN OPTIMAL LOT SIZE HEURISITIC FOR TKF

BACHELOR THESIS

MICHA KUIJPER
INDUSTRIAL ENGINEERING AND MANAGEMENT
M.R.KUIJPER@STUDENT.UTWENTE.NL

UNIVERSITY OF TWENTE.

Supervisors:

Dr. S.M. Meisel	University of Twente
Dr. B. Alves Beirigo	University of Twente
Ferran Haperink	Company supervisor

Management summary

Sequence-dependent setup times in a cable manufacturing environment play an important role in the efficient time planning of production but increase the complexity of models that determine optimal production quantities and sequences. Heuristics are needed to solve this type of problem. This thesis uses the solving procedure framework of Laguna (1999) adjusted to the specifications of the cable manufacturing process at TKF. First, the initial quantities and sequences are determined. Using this initial solution, the production times are further optimized. We present an alternative to the last optimization step of the framework of Laguna. This approach follows a heuristic that can be solved by hand, which improves setup times. Compared to planners at TKF, the sequence-dependent setup times can be improved by 30%. Another focus of this thesis is the comparison of two different production strategies on the machine to be optimized. The strategy called single decks has a higher output but longer sequence-dependent setup times, the other strategy called double decks has a lower output but also lower longer sequence-dependent setup times. Both strategies are optimized with the heuristic and the best option is chosen based on production time and financial implications.

Preface

Dear Reader,

before you lies my bachelor thesis with the title "Optimizing the sequence-dependent setup times of a machine using an optimal lot size heuristic for TKF". In the advancement of academic endeavors, I would like to thank my first supervisor, Stephan Meisel for his guidance, mentorship, and wisdom significantly enhancing the quality of this thesis. I would also like to thank my second supervisor, Breno Alves Beirigo, for providing valuable feedback.

I extend my deepest appreciation to Ferran Harperink, the company supervisor from TKF, whose invaluable guidance enabled me to comprehend the problem and provided me with insightful directives for its resolution. Additionally, I wish to express my gratitude to all the employees of TKF who generously supported me during this research endeavor.

Best,
Micha Kuijper

March 1st, 2024

Contents

List of terms

AIMMS A software to model supply chain scenarios. 15, 30

CLSDP Capacitated lot-sizing problems with sequence-dependent setups. iii, 10, 39

CLSP Capacitated lot-sizing problems. iii, 10

CPLEX A software to solve DPs. 15, 30

DP Dynamic Programming. 4, 5, 8, 9, 11, 12, 16–18

FT Fijntrek. 1, 2

GT Groftrek. 1

Haspel A haspel is a term used to describe a reel with about 500kg of wire. iv, 14, 20, 21, 23, 26–29, 32, 38

MPSM Managerial Problem Solving Method. 3, 7

MT Middentrek. 1, 2

TKF Twentsche Kabel Fabriek. i, iii, 1–3, 5, 6, 29, 37, 39, 44

TKH Group Twentsche Kabel Holding Group. 1

TSP Travelling Salesman Problem. iii, 11, 15–18, 31, 33

WL Wikkellijn. 1

1 Introduction

In this chapter I give a broad overview of the topic of the thesis. I will give an introduction to the company and the production process, explain the methodology used to solve the problem described, analyze the core problem, explain the key concepts used, measure the norm and reality, describe a problem-solving approach, describe the intended deliverables and finally list the research questions and model assumptions resulting from the problem description.

1.1 Twentsche Kabel Fabriek (TKF) – A brief introduction

TKF is a cable wire producer with headquarters in Haaksbergen. The company has grown over the years, and multiple plants have been opened worldwide in recent years, due to the energy transition and the growth of wind energy. The company is still family-owned and part of the TKH Group, a technology company with a €1.58 billion market cap.

1.2 Production process outline

My research project is at the beginning of the supply chain, where raw copper wire is first drawn into smaller diameters and then braided again into flexible cables with thicker diameters, see figure 1. The first machine, called Groftrek (GT), is the start of every cable type production and it draws 8mm thick copper wire that gets delivered on 6-ton coils into smaller wire sizes like 3mm or 2.4mm. The next step in the assembly occurs on machines, called Middentrek (MT) and Fijntrek (FT), which draw the wire into even smaller sizes like 1.15mm or 0.3mm. After that, Wikkellijn (WL) braids multiple thin wires into flexible cables, for 0.3mm FT is also able to do this step. Now, multiple machines are needed for the assembly but there exists a machine that combines the drawing step of MT, FT, and the braiding step of WL into one, see figure 1.

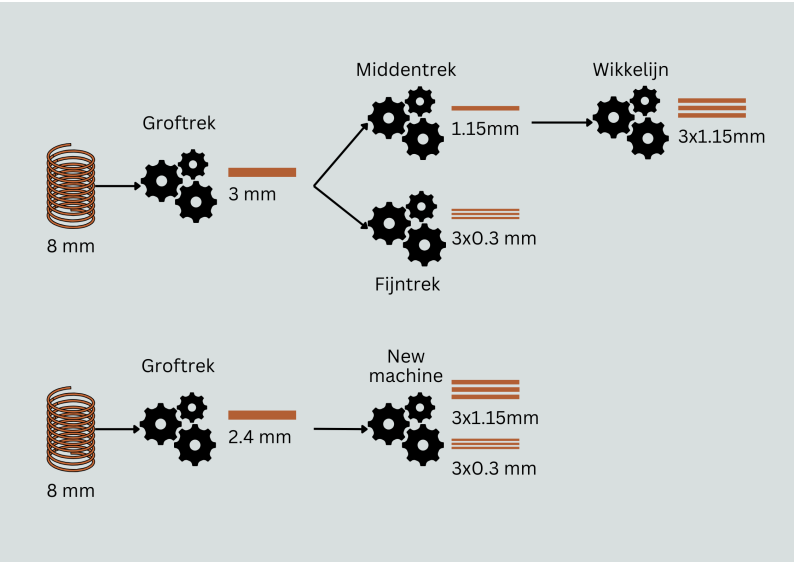


Figure 1: Old and new production process outline

1.3 Background - Investment in a new wire drawing machine

Management has decided to purchase this machine to replace three old machines. The new machine is faster and reduces costs because it combines two production steps into one. The old machines are not profitable to other competitors, as the production style and output do not meet modern standards anymore. This complicates the goal of pursuing the market leader position. Currently, the process is creating waste in lean terminology (Holloway & Hall, 1997), it is cheaper to buy from competitors than to produce in-house. An investment in a faster machine makes it possible to produce all formerly outsourced demand again in-house, this will reduce costs, and waste, and add value to the process. Being able to produce everything in-house is better for TKF, as they have less supplier dependability and control over the quality of products, which has been an issue with some suppliers in the past. Next to that, the new machine results in an improvement in the workforce and a reduction of 8.5 FTE, the main reason the investment is profitable. Although the investment involves workforce reductions, this is in line with the CSR goals of TKF. Operators perceive the current work on the old machines as labor-intensive, unpleasant, and as a waste of their knowledge and skills, as the work can be done without any diploma. After each drum is completed, it needs to be changed manually instead of automatically by the machine. Investing in a new machine leads to more automation with faster production and more value-adding technical work that operators are trained for.

1.4 Background - The new machine changes the supply chain flow

With the investment in a new machine, the flow of the supply chain will change:

1. There will be one new machine instead of three old drawing machines (MT1, MT2 and FT2). The demand for the three machines is now combined into one.

2. The cable types 3,4,5,6x1.15mm can now be drawn and braided on one machine. This notation means 1.15mm with three, four, five, or six wires. Before 3mm coils were drawn into 1.15mm barrels, then on another machine, 1.15mm barrels were braided into 3,4,5,6x1.15mm. With the arrival of the new machine, the drawing and braiding step is now done on a single machine.

3. New cable design and sizes: 3x0.4mm, 8x0.4mm, 3x0.5mm, 8x0.5mm, 3x0.6mm and 8x0.6mm wire instead of 3x0.3mm, 8x0.3mm. TKF used to produce 3x0.3mm and 8x0.3mm as input for conductors, this is not the industry standard as thicker cable sizes like 0.4mm can be also used to produce conductors with cross sections of $10mm^2$ and larger. This is faster and cheaper to produce.

4. The new cable types also need intermediate storage; this place does not yet exist. The existing inventory storage needs to be revisited.

5. Shrinking inventory capacity (Assumption: 10%). The new machine will take up more space than the area of the three old divested machines affecting the inventory storage which will shrink as more space is needed for the machine. Now it is still unclear how severe this impact will be, in consultation with the Supply Chain department, the assumption of a ten percent shrinkage is made.

1.5 Methodology - Managerial-Problem-Solving Method

For the academic structure of this bachelor assignment, I will use the Managerial Problem Solving Method (MPSM) from Heerkens and van Winden (2017), see figure 2. Throughout the project plan, I will refer to this methodology.

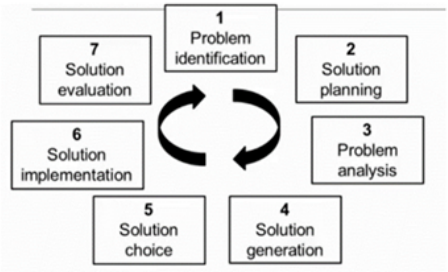


Figure 2: MPSM cycle, image taken from Heerkens and van Winden (2017)

1.6 Problem statement and core problems

For this research, several steps need to be executed before the machine is ready to produce and several core problems are hindering the successful implementation of the machine, see figure 3. This relates to the first step of the MPSM cycle, see figure 2. To support the TKF management in understanding the effect of changing to the new machine, this thesis scientifically investigates the missing production scheduling policy core problem based on the following task:

"How does the new machine efficiently cluster and sequence orders such that the total production time is minimized?"

The other core problems are taken care of by other departments within TKF:

- From a Human Resources perspective, it is a problem who to train on the new machine and whose contracts should not be extended.
- From an R&D perspective, it needs to be determined where to locate the machine. The issue is that the new machine takes up more space than the area of the three old machines, the definitive machine location is yet unknown as it cannot just take the place of the three old machines.

These three issues therefore result in the core problems of the action problem: Successful implementation of the new machine. By offering a scheduling policy to TKF, a large stake of the implementation action problem can be solved.

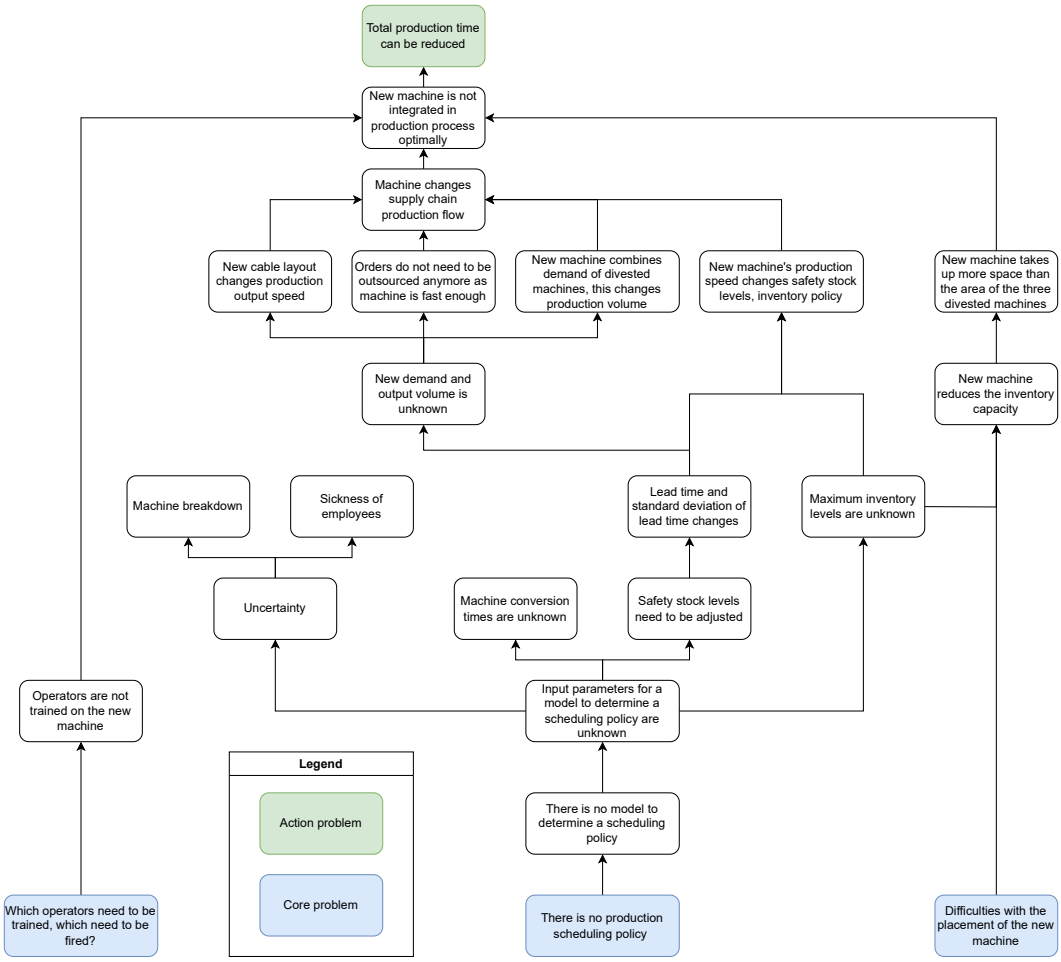


Figure 3: Core problems

1.7 Explanation of key concepts

This section explains the variables safety stock, inventory, total production time, demand and the concepts of operations research, dynamic programming, heuristics, policy and lean which are needed for a basic understanding of the proposed research:

- Safety stock is needed, because we want to know what the minimum amount of inventory is needed to satisfy demand on time without getting out of stock. This parameter will be computed using the standard deviation of demand assuming a 95% service level, the corresponding formula gathered from the book of Chopra (2019, p. 354) is

$$SafetyStock = Z * \sigma_{Demand} * \sqrt{L}, \tag{1}$$

  where L is the average lead time, that is the time needed to produce an item, and σ_{Demand} is the standard deviation of demand.

- The inventory level of item x is the current number of items x in stock, where the maximum inventory level of item x is the maximum amount of items x able to have in stock.

- The research is rooted in operations research, with Dynamic Programming (DP) and Heuristics to solve a problem (Winston & Goldberg, 2004). A DP model is a set of

mathematical equations called constraints with a minimizing or maximizing statement called an objective function that derives the optimal solution based on the constraints. Heuristics are a method to easily solve a DP model problem with large computational times as they do not compute the exact solution but yield a solution that approximates the optimal solution (Dixon & Silver, 1981). For further explanation see section 2.

- The solution of a heuristic can be called policy, in this case, a scheduling policy. A scheduling policy can be defined as "a rule that specifies how each period's decision is chosen"(Winston & Goldberg, 2004, p.1038). In this case, a period is a production day with 24 hours where the decision lies in optimally choosing which items and which quantity of items to produce.
- Another important construct is the company's production strategy. TKF is a lean-oriented company (Holloway & Hall, 1997). The inventory is inspired by a Kanban system (Kiran, 2019) with an assigned space for each item to limit the buildup of excess inventory with a pulling scheme where the consecutive production takes the item when needed instead of a pushing scheme where a finished product is placed at the next machine. So, the principle is to only produce when things are pulled away, so there is space to produce again. Just in time is used for scheduling in general for TKF (Kiran, 2019), but in the department the new machine will be installed this approach is neglected because large batch sizes are more important than small production quantities, otherwise there are too many machine conversions.
- The new machine has two decks. A deck can be seen as a side of the machine where each deck/side can produce the items needed. This means that a machine with two decks can be seen as two single machines combined into one in terms of output.
- Total production time is the sum of production needed to produce the orders, setup time, and penalties. The objective function aims to find a trade-off between large lot sizes to work around long setup times (restricted by maximum inventory) and meet production demand on time with inventory penalties to produce more often.
- Demand will be measured by looking back at historical data starting from 01-01-2022 until the date of collection, which is 22-11-2022.

The goal of this research is not to minimize production cost but to create a scheduling policy that reduces total production time, see section 1.8.

1.8 Measurement of norm and reality

It is essential to outline the present situation and our starting point to establish objectives and targets for this bachelor thesis and TKF. By clearly defining the current status, we can subsequently articulate the anticipated goals and targets to be accomplished.

1.8.1 Reality - unknown lot sizes and unknown scheduling policy

Currently, the machine is purchased but not yet implemented. It can be said that the machine is not yet integrated into the production process. This problem is related to two missing items: Newly determined parameters for the DP model and a scheduling policy derived from solving the DP model through a heuristic. In this case, the missing input parameters are demand, safety stock, and maximum inventory levels. The reality is that the previously mentioned parameters are based on old demand and production volume and have not yet adapted to the new production constraints.

1.8.2 Norm - optimal lot sizes and scheduling policy

It is expected that a new scheduling policy will be implemented that reduces total production time and determines optimal lot sizes which are based on the newly determined parameters. The quality of this scheduling policy will be measured by comparing the total production time after using the heuristics with the total production time before using the heuristic. In consultation with TKF, the goal formulated as stated above is to reduce the total production time after employing the heuristic compared to using no heuristic with producing orders based on the nearest due date. The norm is to have parameters and policy determined optimally, see section 1.7 for definitions. For this research optimality is reached when the heuristic method is solved. The model must also make assumptions about shrinking inventory capacity and uncertainty.

1.9 Problem-solving approach

My framework which can be seen in figure 4 structures the problem-solving approach of finding a solution for the scheduling policy core problem (see figure 3), it covers the 2nd till the 7th step of the MPSM from Heerkens and van Winden (2017).

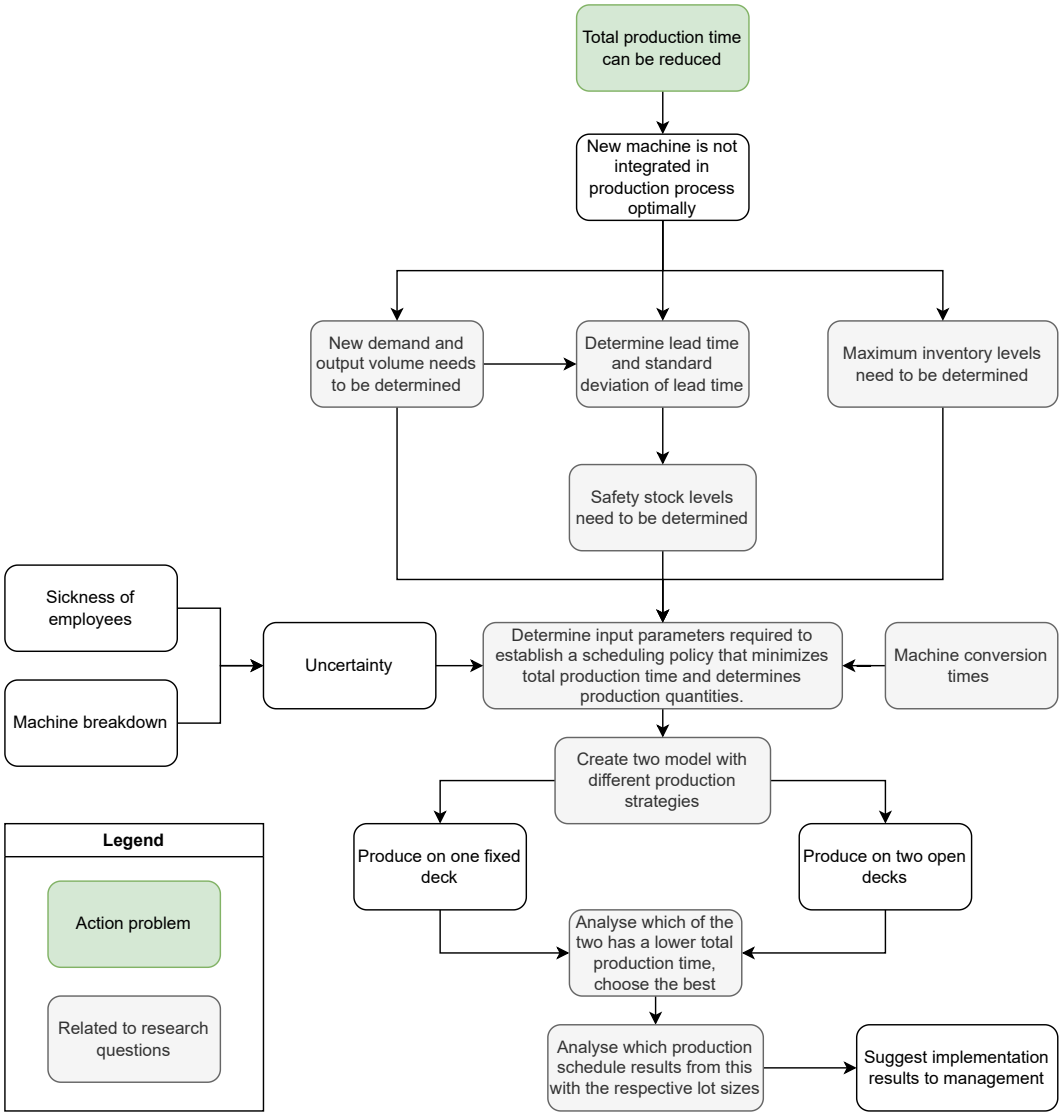


Figure 4: Problem solving approach

To further elaborate on the framework, when input parameters are determined and a suitable model framework is found, two different production policies are tested. We will analyze what effect a different strategy has on the total production time and the lot sizes. The two different production strategies entail the following:

- Produce on one fixed deck: This means that one deck of the machine is fixed on 1.15mm. The other deck produces 0.4mm* and 0.5mm* sizes. This results in a lower output as either the upper or lower deck is producing in total less meters per hour output but less conversion times as the conversion from x1.15mm to x0.4mm, x0.5mm and vice versa does not need to be executed.
- Produce on two open decks: This means that both decks are not fixed to one cable product. This results in higher output (2x8x0.4mm) but also higher conversion times as the long conversion from x1.15mm to x0.4mm, x0.5mm and vice versa needs to be made occasionally.

1.10 Intended deliverables

Related to the problem-solving approach in figure 4 and the outcomes of the systematic literature review (section 8), the intended deliverables are:

- Input parameters for a mathematical model: Demand, lead time, standard deviation, safety stock, maximum inventory per cable type, and assumptions about shrinking capacity and uncertainty.
- DP model with an objective function and constraints.
- Two heuristics for the two production strategies: DP model solved employing a heuristic that reduces total production time and finds out the optimal batch sizes with two different production strategies.
- Analysis of the test results which one performs better with a computation of total production time.

1.11 Knowledge problems and research questions

As stated in figure 4, several parameters are needed to determine a mathematical model, and those are related to the sub-research questions. The main research questions are related to the performance and design of the mathematical model.

1.11.1 Main research questions

1. Which production policy (1 fixed deck vs 2 open decks) yields a better performance in terms of the lowest total production time and which batch sizes result from this?
2. Which mathematical heuristic fits the goals to reduce the total production time and to determine optimal lot sizes in a manufacturing environment? (Related to finding a correct mathematical framework method)
3. How to minimize the total production time considering the input constraints (from no policy to approximation of minimal total production time) for the newly purchased machine? (Related to the correctness of the model equations)

1.11.2 Sub research questions

4. What are the safety stock levels needed, considering a 95% service level, based on demand for wires drawn on the new machine?
5. What is the demand per type of wire on the new machine?
6. What is the maximum possible mix of inventory of wire drawn, assuming a 10% shrinking inventory capacity, on the new machine (from 100% inventory storage to 90%)
7. How to implement uncertainty in a DP model and heuristic?

1.12 Model assumptions

Specific assumptions can be made regarding the design of the model:

- The machine represents a multi-item production because multiple items are drawn on the machine. The production is single level as there is one machine involved.
- The inventory per item is bound by a maximum amount of inventory per item.
- The production amount is bound by the machine capacity. The processing time per item plus the machine conversion times must be larger or equal to the machine capacity.
- Machine conversions depend on what is produced beforehand.
- Batch sizes are an integer number of a predetermined unit (reel 400kg).
- Backorders or backlogging are not possible.
- Overtime is not possible as the machine already runs 24 hours and 5 days per week. Weekend work is restricted by the company due to economic restrictions.
- No standard inventory costs are incurred per item, but an inventory penalty is incurred when the inventory level is lower than the safety stock.
- The lot sizes per product need to be determined per period.
- It needs to be investigated how exactly uncertainty can be implemented.

2 Literature review

In this chapter, I will investigate the literature related to the integration of the theory from chapter 1. At the end of this chapter, I will compare the papers of Laguna (1999) and Meyr (2000), with the conclusion, that Laguna (1999) best fits the assumptions.

2.1 Capacitated lot-sizing problems (CLSP)

The scope of this research is to determine the optimal total production time with respective optimal batch size quantities. The most well-known formula for finding an optimal batch size is the Economic Order Quantity formula (EOQ) (Bahl, Ritzman, & Gupta, 1986; Maes & Wassenhove, 1988). To solve more complicated problems with more constraints, lot sizing models like the Wagner Whitn algorithm or the Silver-Meal heuristic are useful. These models can be classified as uncapacitated lot-sizing problems, the sum of setup cost and inventory holding cost are minimized and there is no interdependence between items, lot-sizing decisions can therefore be made based on a single item (Maes & Wassenhove, 1988). For the lot-sizing problem of this research, multiple items are produced on a single machine and the sequence of the items with the respective lot-sizes needs to be determined. This leads to a classification of lot-sizing models called Capacitated lot-sizing problems (CLSP) (Bahl et al., 1986; Maes & Wassenhove, 1988; Quadt & Kuhn, 2008). An example is the Dixon-Silver heuristic (Dixon & Silver, 1981) that determines the lot sizes with limited capacity for the multi-item and single-level cases. Due to the increase in variables when multi-item problems are solved, the computational complexity increases. Models in CLSP are NP-hard (Maes & Wassenhove, 1988), which means that it is extremely difficult to solve the problem in a reasonable time. Heuristics like Dixon-Silver are a solution to this problem as they do not compute the exact solution but yield a result that approximates the optimal solution (Dixon & Silver, 1981). The Dixon-Silver heuristic assumes that setup times are negligible, machine capacity constraints only concern the run time of items. Other researchers have included setup cost and/or times in their model that depends on what is produced beforehand leading to a subcategory of CLSP called Capacitated lot-sizing problems with sequence-dependent setups (CLSDP).

2.2 Capacitated lot-sizing problems with sequence-dependent setups (CLSDP)

When setup times are non-zero the problem becomes NP-complete, meaning that one cannot efficiently say if a solution to the problem exists at all. In this case, it is necessary to employ heuristics that do not ensure an optimal solution, but instead discover a reasonably satisfactory solution within a reasonable amount of computational time (Quadt & Kuhn, 2008). Another property included in the CLSDP heuristics is setup carry-over, the setup state of the last item in a period is taken as the starting state in the next period. An item produced at the end of a period can be produced at the start of the next period without any setup. Relevant literature is the following (Quadt & Kuhn, 2008):

Selen and Heuts (1990) examines a lot-sizing and scheduling problem in a chemical environment, where batch sizes are an integer number of predetermined unit sizes. Therefore, they developed a heuristic that assumes fixed batch sizes with sequence-dependent setup cost, carryover is possible. First, a feasible production plan is determined on a lot-for-lot basis, then lots are shifted to earlier periods if this yields lower costs based on the trade-off between setup and inventory cost. A numerical example is stated with 15 products over a 10-week planning horizon, where lot sizes and production sequences are computed. Haase (1996) solves the CLSDP with sequence-dependent setup cost by a backward randomized regret-based heuristic, backorders and backlogging are not considered. A small sample is tested on the reliability of the heuristic compared to an exact algorithm. Haase proves that

the deviation from the optimal result is small and therefore the heuristic yields acceptable results. Haase and Kimms (2000) introduce a model where sequences are pre-determined, the branch and bound heuristic chooses one of the following sequences. The heuristic is scheduling backwards and experiments with 3 products and 15 periods and 10 products and 3 periods are executed. Fleischmann and Meyr (1997) propose a solution approach that involves Threshold Accepting and utilizes a model with a single machine that establishes a complete product order, disregarding setup times and back-orders. The method generates new setup sequences by executing neighborhood operations, such as inserting a setup for a specific product, exchanging two setups, or deleting one, starting from an infeasible solution. Each setup sequence is assessed using a heuristic procedure that determines the production and inventory volumes. The computational tests are conducted using the instances from (Haase, 1996). Meyr (2000) builds upon the solution approach from Fleischmann and Meyr (1997) by adding setup times. The solving procedure is also split into two parts. First, Meyr determines the sequence of items per period, second, he determines the respective lot sizes per item. Threshold accepting (TA) and simulated annealing (SA) are presented as two options to solve the local search heuristics. He conducts an experiment with 18 items and 8 periods. Laguna (1999) states a model for a single machine with sequence-dependent setup times instead of costs. He adds the option to produce overtime. The lot-sizing problem is divided into sub-problems. The first step consists of solving a DP model to determine initial quantities and the second step consists of solving a Travelling-Salesman algorithm to determine initial sequences. The third step again solves a DP model but now with the sequences and their setup times from the Travelling Salesman Problemincluded. Next, a tabu search heuristic is employed to further improve the solution. Each solution from the heuristic is evaluated by using the Traveling Salesman procedure to re-sequence the products and the DP from the third step to find lot sizes. All models described above either include sequence-dependent setup cost with (optional) constant setup time or sequence-dependent setup time with (optional) constant cost. The first model to include sequence-dependent setup costs and sequence-dependent setup times is from Almada-Lobo, Klabjan, Carravilla, and Oliveira (2007). They constructed a five-step heuristic to efficiently solve a model with sequence-dependent setup costs and times. More recent papers add more complexity by including both sequence-dependent setup costs and sequence-dependent setup times like Almada-Lobo et al. (2007).

2.3 Domain specific literature

For this research model, it is not necessary to include both sequence-dependent setup costs and times. Setup costs are equal to setup time because the cost for a setup is the time it takes, therefore we can disregard models like the one from Almada-Lobo et al. (2007). Furthermore, it is important to note that for this research the capacity is restricted by the setup time from one item to another and the times the items themselves take in a period. One can also add this constraint to models that do not include it yet, however often the heuristic solving the problem statement is not fully suitable as it needs to be adjusted to consider sequence-dependent setup time. This means that for the sake of simplicity Haase (1996), Fleischmann and Meyr (1997) and Haase and Kimms (2000) can be disregarded too. Selen and Heuts (1990) introduce an interesting assumption where batch sizes are an integer number of a predetermined unit size, this is also the case for this research. The production unit size is a reel, and the quantity is an integer number of reels. A downside of their model is that only sequence-dependent setup costs are considered but their notation of lot sizes is helpful to use in another heuristic model. The two models left are Meyr (2000) and Laguna (1999). Both are suitable with a few adjustments such as adding the assumption from Selen and Heuts (1990) with a predetermined unit size. The objective function of Laguna (1999) does not include sequence-dependent setups, only inventory costs, whereas Meyr's (2000) objective function does include them. Laguna (1999) includes them in the second step of the heuristic where a travelling-salesman problem is solved. Both heuristic solving approaches

consist of several steps, after having read both papers Meyr's (2000) approach is more complicated but might be faster. There is no comparison of computational times possible as Laguna (1999) does not include them in his paper. Evaluating all these considerations, Laguna (1999) is a better fit compared to Meyr (2000) for this research because the approach to solving the problem is better to implement and there is more overlap in the assumptions made.

2.4 Analysis on the framework of Laguna (1999)

It needs to be investigated and tested if the sequence-dependent setup times can be included in the objective function of the first step and third step of the solving approach. If not, the traveling salesman application already optimizes the sequence-dependent setup times such that the result, in the end, can be assumed approximately optimal. During the execution of this bachelor's thesis and when implementing the code of the DP models it needs to be investigated how to implement uncertainty (Eq.5.7). The option to use overtime is not needed because the machine already runs 24 hours and 5 days per week. Weekend work is restricted by the company due to economic reasons.

3 Mathematical model and Laguna's solving procedure of the model

Based on the problem description in chapter 1 and the literature research in chapter 2, I developed a mathematical model, which can be heuristically solved using the framework of Laguna (1999). At the end of this chapter, I will present an alternative-solving approach to the last step of Laguna's approach.

3.1 Mathematical model

The following mixed integer linear programming is the mathematical representation of the problem formulated where constraints (5) - (11) are based on Laguna (1999). Laguna also adds variables for possible overtime in constraints (5) - (7), those are not needed for this case. Constraint (13) is inspired by Selen and Heuts (1990), as their paper also restricts quantities to an integer number of predetermined unit sizes. The notation style is based on the book of Winston and Goldberg (2004).

$$min \sum_{i=1}^{N}\sum_{t=1}^{T} r_i x_{it} + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T} c_{ij} q_{ijt} + \sum_{i=1}^{N}\sum_{t=1}^{T} p_i P_{it}, \tag{2}$$

subject to

$$I_{i,t-1} + x_{it} - d_{it} = I_{it} \quad \forall i,t. \tag{3}$$

$$P_{it} \geq S_i - I_{it} \quad \forall i,t. \tag{4}$$

$$\sum_{i=1}^{N} r_i x_{it} + \sum_{i=1}^{N}\sum_{j\neq i}^{N} c_{ij} q_{ijt} \leq A_t \quad \forall t. \tag{5}$$

$$r_i x_{it} \leq A_t (y_{i,t-1} + \sum_{j\neq i}^{N} q_{ijt}) \quad \forall i,t. \tag{6}$$

$$\sum_{j\neq i}^{N} r_j x_{jt} + A_t (y_{i,t-1} + y_{i,t}) \leq 2A_t \quad \forall i,t. \tag{7}$$

$$y_{i,t-1} + \sum_{j\neq i}^{N} q_{jit} - y_{it} - \sum_{j\neq i}^{N} q_{ijt} = 0 \quad \forall i,t. \tag{8}$$

$$\sum_{j\neq i}^{N} q_{ijt} \leq 1 \quad \forall i,t. \tag{9}$$

$$\sum_{j\neq i}^{N} q_{jit} \leq 1 \quad \forall i,t. \tag{10}$$

$$\sum_{i=1}^{N} y_{jit} = 1 \quad \forall i,t. \tag{11}$$

$$I_{it} \leq H_i \quad \forall i,t. \tag{12}$$

$$I_{it}, x_{it}, d_{it}, P_{it}, H_i, S_i \in \mathbb{Z} \quad \forall i,t. \tag{13}$$

$$I_{it}, x_{it}, d_{it}, P_{it}, H_i, S_i, r_i, h_i, c_{ij}, A_t \geq 0 \quad \forall i,t. \tag{14}$$

$$i,j = [1,..,N], \quad t = [1,..,T]. \tag{15}$$

$$q_{ijt}, y_{it} \in [0,1] \quad \forall i,j,t. \tag{16}$$

$$I_{i0} = S_i \quad \forall i. \tag{17}$$

N = Products, T= Days
i, j = Items to produce
I_{it} = Inventory of item i at the end of period t
x_{it} = Integer number of Haspels i to produce in period t
d_{it} = Demand of item i in period t
P_{it} = Penalty of item i in period t
H_i = Maximum inventory of item i
S_i = Safety stock of item i
r_i = Production time per Haspel of item i
p_i = Penalty cost per Haspel of item i
c_{ij} = Setup time from item i to j
A_t = Machine capacity per period t
$q_{ijt} = 1$ when a setup from item i to j occurs in t, 0 otherwise
$y_{it} = 1$ when item i is the last setup of period t, 0 otherwise.

The objective function (2) in this model minimizes the sum of production time, sequence-dependent setup time, and penalty costs bounded by constraints (3) - (17). The first constraint handles the flow of inventory, which is based on the inventory of the previous period plus the production quantity minus the demand. Constraint (4) handles the inventory penalty, which is only incurred when inventory is below the safety stock, as $P_{it} \geq 0$. The penalty is the difference between safety stock and inventory. Production time with related setup times per period is bound by the machine capacity with constraint (5). Additionally, constraint (6) regulates that the production of product i in day t is only allowed, if the facility was set up to produce i at the end of $t-1$ or a changeover into i occurs during period t. Constraint (7) addresses the scenario where the facility produces product i at the end of $t-1$ and is also set up to produce the same product i at the end of month t. In such cases, we consider product i as the only product manufactured in month t, prohibiting the production of any other product during that time and ensuring that each month's sequence of jobs does not schedule a product twice (Laguna, 1999, p.128). To balance the changeovers, constraint (8) regulates that if the facility is set up to produce j at the end of period $t-1$, or a changeover into j occurs during month t, then a changeover from j must occur or the facility will be set up to produce j at the end of t (Laguna, 1999, p.128). Constraints (9) and (10) limit the number of setups from product i to any other product j to a maximum of 1 and vice versa. The binary last setup of period variable y_{it} is restricted by constraint (11) that only one product can be stored as the last setup in a period. Inventory-wise, the Inventory of product i in period t is bound by the maximum inventory of product i in constraint (12). As explained above, quantities are restricted to an integer amount of predetermined unit sizes in constraint (13). Additionally, all variables must be larger or equal to zero (constraint (14). Finally, the starting Inventory i is equal to the safety stock i.

3.2 Solving Procedure based on Laguna's framework

The dynamic programming model described in section 3.1 is not possible to solve because it is too complex, based on Laguna (1999). Thus, a heuristic is needed that approximates the optimal solution of a dynamic programming model. Laguna (1999) developed a heuristic that divides the problem into smaller sub-problems. I will use this framework to solve the problem described throughout chapter 1.

3.2.1 Step 1 - Determine initial production quantities

The first step is to determine initial production quantities per period

$$min \sum_{i=1}^{N} \sum_{t=1}^{T} r_i x_{it} + \sum_{i=1}^{N} \sum_{t=1}^{T} h_i P_{it}, \tag{18}$$

subject to

$$I_{i,t-1} + x_{it} - d_{it} = I_{it} \quad \forall i, t. \tag{19}$$

$$P_{it} \geq S_i - I_{it} \quad \forall i, t. \tag{20}$$

$$\sum_{i=1}^{N} r_i x_{it} \leq A_t \quad \forall t. \tag{21}$$

$$I_{it} \leq H_i \quad \forall i, t. \tag{22}$$

$$I_{it}, x_{it}, d_{it}, P_{it}, H_i, S_i \in \mathbb{Z} \quad \forall i, t. \tag{23}$$

$$I_{it}, x_{it}, d_{it}, P_{it}, H_i, S_i, r_i, h_i, A_t \geq 0 \quad \forall i, t. \tag{24}$$

$$i, j = [1, .., N], \quad t = [1, .., T]. \tag{25}$$

$$I_{i0} = S_i \quad \forall i. \tag{26}$$

For this step, the sequence-dependent setup times are not considered to reduce complexity. The objective function minimizes the sum of production time and penalty cost. Equations (18) - (26) are solved in AIMMS with CPLEX, a computer program for solving operations research-type problems.

3.2.2 Step 2 - Determine initial sequences with a TSP

The next step is to use the solution from section 3.2.1 and determine a sequence from the initial quantities per period with a TSP. This method finds the shortest path between products to be produced per day by solving matrices which are the element-wise product of the changeover times between products found in tables 13 and 14 and a matrix of the production quantities per day, being 1 if the production quantity amount is strictly > 0 and 0 otherwise. The matrices are solved in sequential order starting from $t = 1$ (Laguna, 1999) and take into account what the starting setup is, which is the last setup of the day before L_{t-1}. This step is programmed and solved in Python 3.12 (see section 9.4 in the appendices) based on the following equations that define the matrices to be solved per period:

$\delta_{ij} = c_{ij}$ if i and j > 0,
$\delta_{ij} = 0$ if i = j or j = 0,
$\delta_{0j} = c_{L_{t-1},j}$ if j > 0 and t > 1,

where c_{ij} is the setup time between i and j from tables 13 and 14 and δ_{i0} is a row of containing only zeros such that the distance matrix does calculate the time to go back to the start but stops at the last item. This means that every period has a unique matrix to be solved.

3.2.3 Step 3 - Sequence evaluation

After having determined the initial quantities and sequences, the next step is to determine if the initial solution is feasible by adding the sequence-dependent setup times (Laguna, 1999).

$$min \sum_{i=1}^{N} \sum_{t=1}^{T} r_i x_{it} + \sum_{t=1}^{T} (C_t + LF_t) + \sum_{i=1}^{N} \sum_{t=1}^{T} h_i P_{it}, \tag{27}$$

subject to

$$I_{i,t-1} + x_{it} - d_{it} = I_{it} \quad \forall i, t. \tag{28}$$

$$P_{it} \geq S_i - I_{it} \quad \forall i, t. \tag{29}$$

$$\sum_{i=1}^{N} r_i x_{it} + L_t + F_t \leq A_t - C_t \quad \forall t. \tag{30}$$

$$L_{t-1} + F_t = LF_t \quad \forall t | t > 1. \tag{31}$$

$$I_{it} \leq H_i \quad \forall i, t. \tag{32}$$

$$q_{it} x_{it} \geq x_{it} \quad \forall i, t. \tag{33}$$

$$I_{it}, x_{it}, d_{it}, P_{it}, H_i, S_i \in \mathbb{Z} \quad \forall i, t. \tag{34}$$

$$I_{it}, x_{it}, d_{it}, P_{it}, H_i, S_i, r_i, h_i, A_t \geq 0 \quad \forall i, t. \tag{35}$$

$$i, j = [1, .., N], \quad t = [1, .., T], \quad q_{it} = [0, 1]. \tag{36}$$

$$F_1 = LF_1. \tag{37}$$

$$I_{i0} = S_i \quad \forall i. \tag{38}$$

Here, the objective function (27) also includes the setup times incurred per period. C_t covers the sum of all setups from the first item to the last item to produce in a sequence per day gathered from solving the TSP matrices, LF_t covers all setups made from the last item of a period to the first item of the next period. In this DP, the setup time LF_t can be shared by both periods as described in constraint (31), this is helpful for the machine capacity constraint (30), where setup times L_t and F_t are incurred based on the resting machine capacity. Furthermore, the setup to the first item on day one is entirely charged to F_1, see constraint (38). Constraint 33 ensures that the production of an item is only set up if the product has been scheduled in the TSP matrices of step 2. When $q_{it} = 1$, the product has been scheduled in the TSP, if not $q_{it} = 0$.

3.2.4 Step 4 - Coordinating procedure

Laguna's last and final step is to implement a short-term memory Tabu Search heuristic to further improve the sequence. Once an initial solution is found, the coordinating procedure performs schedule changes to search for an improved outcome, which is when the total production time decreases. In this step one loops over all days and all items, where P_t is the set of products i to be produced in t and the four possible moves to select from are:

For each $i \in P_t$:
  Move M1: Delete product i from the set.
  Move M2: If i is not the last product in the set, make it the last product to sequence.
For each $i \notin P_t$
  Move M3: Add product i to sequence.
  Move M4: Make i the last product to produce in the sequence.

For each possible move, the TSP and DP of step 3 are solved again. In contrast to the TSP matrices of Step 2, in this step the TSP matrices add the setup times to the first item on

the next day F_{t+1} in row 0 such that $\delta_{i0} = c_{i,F_{t+1}}$ if i > and t < T. Per iteration, the best move, which is the one with the lowest value found for the total production time of the DP based on the sequences found from the new TSP matrices, is selected and is denoted as a move, see the second code listing in the appendices, see 9.4.

3.2.5 Discussion on the Tabu Search of Step 4 and presentation of an alternative heuristic

The moves Laguna (1999) presents assume certain patterns in the found sequences and production quantities. Laguna's DP models have no penalty costs when inventory is below safety stock, but there are holding costs. This means that building up inventory is prevented instead of stimulated. In terms of the number of setups of a product, the production of a product will be set up less often in Laguna's case to prevent inventory from building up compared to my case where more setups are enforced to prevent penalty costs. This means that adding an item that is not part of the sequence already ($i \notin P_t$) is not going to decrease the total production time in a case where the model stimulates setups. Based on this analysis, I conclude that moves M3 and M4 can be disregarded for the research on this application.

Furthermore, deleting an item from a sequence (Move M1) only makes sense, when its production quantity on that day is low, the low production quantity can then be added to other days with leftover machine capacity. But when the production quantity on a day is high and Move 1 is selected, the machine capacity and maximum inventory restrict the model from moving a high production quantity from one to another day.
Move M2 is only of use when there is setup time included in the transition of two days, so when the Last Item$_{t-1} \neq$ First Item$_t$. If the setup time between two days is zero, it means that Last Item$_{t-1}$ = First Item$_t$. In this case the TSP matrices have already found the best solution possible because the matrices take into account the setup to the first item of a day and to the first item of the next day.
Based on these considerations, Lagunas Tabu Search with the respective Moves explained in section 3.2.4 can be adjusted and simplified to the problem of this thesis as follows, see figure 5:
Applying the reasoning above Moves 3 and 4 are not applicable, Moves 1 and 2 can be translated into a heuristic which can be solved by hand. The goal of this heuristic is to further improve the sequences and reduce the setup times. The first step is to use the results of the initial TSP solution of step 2. We check if there is a setup time incurred between Last Item$_{t-1}$ and First Item$_t$. If this is the case, both sequences need to be further investigated for possible improvement. This means that we want to find an item i which is in both sequences that we can move to Last Item$_{t-1}$ and First Item$_t$ such that the setup time between stages becomes zero, but without changing the First Item$_{t-1}$ and Last Item$_t$. An example would be to have a sequence A_{t-1} on day$t-1$ with items I_i scheduled in the order $[I_1, I_2, I_4]$ and a sequence A_t on the following day$_t$ with items scheduled in the order $[I_2, I_3, I_1]$. Here, we can not move I_1 because it is the First Item$_{t-1}$ and Last Item$_t$, but we can move I_2 in sequence A to the Last Item such that the setup time between two stages reduces to zero since then Last Item$_{t-1}$ == First Item$_t$. The optimized sequence A_{t-1} would then be $[I_1, I_4, I_2]$, this improvement relates to Move 2 because we move an item to be the last item to schedule. If there is no such overlapping item as in this example, we can not further improve the sequence by solely observing the sequences without the related production quantities. Therefore, as a next step, we solve the DP of step 3 again.

After that, we can further improve the setup time by deleting setups made. As explained above, deleting setups is only feasible for smaller production quantities because those can be added to other sequences without colliding with the maximum machine capacity and maximum inventory. This means that we restrict ourselves to production quantities of less than 3. If a setup is deleted, we know that the setup time will decrease because we need to schedule for one product less. But this might increase inventory penalties, thus we can establish a rule that we only delete a setup from a sequence if the decrease in setup time outweighs the increase in penalties. Using this, we check for all production quantities less than 3 one for one, if deleting their setup q_{it} (see constraint 33) results in a lower feasible solution of the DP of step 3. If this is the case we apply the change, this improvement step relates to Move M1. For the case of $q_{it} = 1$ with $x_{it} = 0$ as production quantity, deleting the setup will always result in a lower feasible solution, because no penalty costs are incurred when we delete a setup with no production related to it.

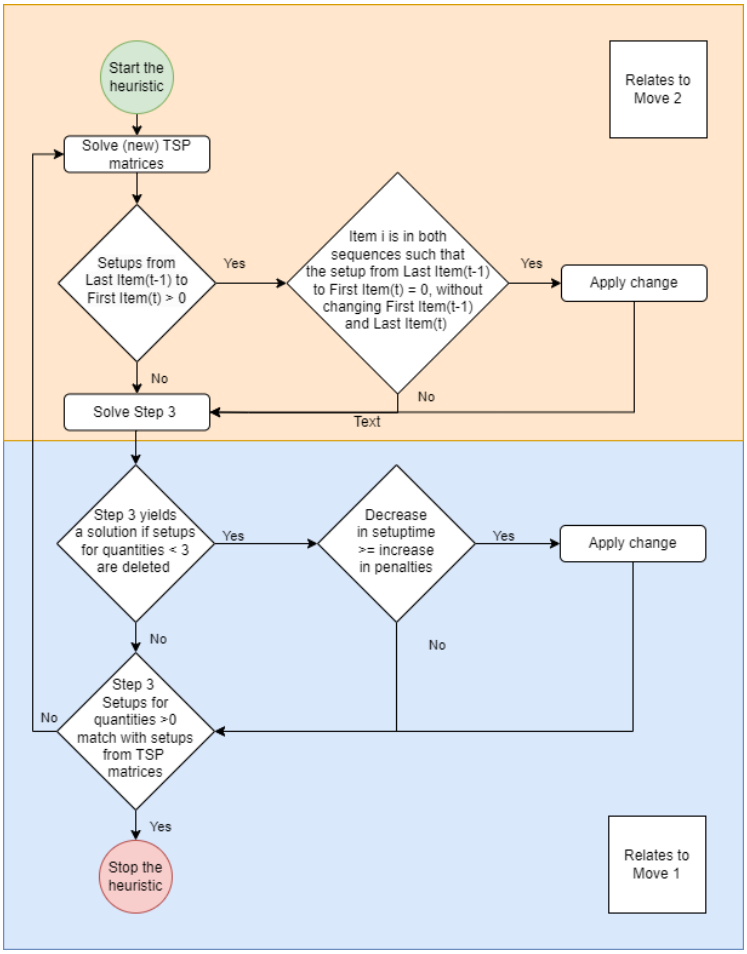


Figure 5: Sequence improvement heuristic

Now we have further optimized the model by deleting setups q_{it} we know, that the setups scheduled in the TSP matrices are not identical to the leftover q_{it} for which the model is allowed to start production. This means that we have to solve the TSP matrices again for all $q_{it} = 1$ to update the sequences to the latest adjustments. As we have recomputed the TSP matrices, we need to also repeat the other steps to check for possible improvement. We can stop the heuristic if we can not delete any items from the sequence anymore such that the q_{it} scheduled in the TSP matrices are identical to the re-optimization of q_{it} after all improvement steps.

4 Data manipulation

The model is now formulated and a solving method is selected, the next step is to gather, clean and analyze data needed to solve the problem formulated to find lot sizes and production sequences. All these steps are needed because the existing data does not specify each of the products to be manufactured, as the new machine changes the supply chain flow (see section 1.4). In this chapter, I will determine a dataset for all 9 products to be produced on the new machine, from which the mean and standard deviation of the dataset follow.

4.1 Data gathering

For this research the data needed is purely secondary and quantitative, the model is based on two types of data: Demand data and data about machine specifications. The data needs to be
Demand data is gathered from one database within the ERP system. Data to collect is historical bookings of wire used, this includes formerly outsourced wire. This database needs to be filtered on the article code of the desired product and the specific date range. In this case, the output is a listing of all booked amounts of the specified product during a given time period. So per production day, multiple bookings can be listed. Per booking the used amount can highly deviate, as operators do not book amounts systematically but every once in a while. To get a better overview of the quantity of products booked, the gathered data is merged to the sum of bookings of a product booked per day. To gather all data necessary the database needs to be filtered twice, for 1.15 multiwire (3,4,5x1.15mm) and 8x0.3mm. Data about machine specifications entails machine conversion times and machine speed per cable type and is gathered from the R&D department.

4.2 Data analysis method

Demand data will be gathered from the last year and needs to be plotted to check which distribution fits the data. This will also clarify if there is seasonality in demand. With this knowledge, the standard deviation of demand per item can be computed and resulting from this the safety stock (equation (1) and chapter 5). Lead time per item depends on the machine specifications data.
The historical demand data utilized in this study is derived from bookings when wire is used. The weight booked when used depends on the weight booked when produced in the production step before. The weight when booked is listed in “terugmeldingen” of production orders, indicating the number of products manufactured. After each finished coil of an order the associated barcode on the production document is scanned and the weight of the product is entered in the ERP system by an operator. If the weight of a coil highly deviates from the prescribed weight of the production document a defect note is added on the document which needs to be discussed with the team leader. Once all barcodes have been scanned the production document is handed back to the planner that verifies whether the ERP data aligns with the prescribed weights on the document. All changes in volume are adjusted and then booked in the ERP system. If there is an irregularity the planner checks with the team leader what the reason is.
Thus, the people responsible for a correct data representation of reality are the operators, the planner, and the team leader consulting in case of irregularities. According to planners, the most errors that occur are with respect to small deviations in kilos between operators and the data from the ERP system, those “small errors” are adjusted by planners.
From a content validity viewpoint, the above proves that the “content of the items adequately represents the universe of all relevant items under study.” (Cooper & Schindler, 2004, p.257). Criterion-related validity regards the “Degree to which the predictor is adequate in capturing the relevant aspects of the criterion“ (Cooper & Schindler, 2004, p.257).

This can be tested by measuring the correlation which is not done for this research. In general, validity investigates if the data gathered measures what it aims to measure (Cooper & Schindler, 2004, p.257). All stakeholders involved in reviewing the data ensure that the data obtained from this process can be deemed reliable, trustworthy, and an accurate reflection of the actual production activities. According to Heerkens and van Winden (2017) quality of research is measured by its validity, where reliability is a subset of. Reliability in this context is guaranteed by filtering the data within the ERP consistently.
For the data from the R&D department, I must assume that the given data is correct, justified by the knowledge that their data is linked directly to information from the machine supplier. Therefore, both types of data, demand, and machine specifications can be deemed reliable and trustworthy.

4.2.1 1.15 multiwire

Currently for the production of 1.15 multiwire wire the ERP system does not specify which type of x1.15mm wire is used per order. This means that the data cannot be used immediately as it first needs to be specified how much of the different types of x1.15mm is used. As the database also stores for which type of final cable x1.15mm wire is used, it can be traced back to what the corresponding wire design and combination of different types of x1.15mm wire per type of final cable is.
The different types of final cables are listed in table 1 and describe the distribution of threefold, fourfold, and fivefold 1.15mm wire needed per type of final cable, retrieved from the shaft number. Table 2 shows the sum of types of x1.15mm used per shaft number, from this distribution one can derive the percentages of types of x1.15mm needed (e.g. 10/12 for 4x1.15mm in a shaft 50 cable). The demand per type of x1.15mm wire can be determined by multiplying the different percentages with the amount of unspecified x1.15mm wire used.

$$Demand_i = BookedLength_{Unspecified} * Percentage_i \tag{39}$$

where i is the type of product. Next, the $Demand_i$ is computed in meters, but to determine the number of Haspels needed the corresponding weight to each length(meters) needs to be computed. Dividing the $Demand_i$ in kilograms by the weight that fits on a Haspel yields the number of Haspels needed, see table 3.

Wire design																				
Shaft number	**Number of wires**	**Type of 1.15 used for each Korf (places) in the machine**																		
50	46	4		4	4		4	4		3	4		4	4			4	4		3
25	23	4					4	4					4	4						3
70	67	5		4	5	4	5	4		5	4	5	4	5			4	5	4	4
35	32	4			4	3		4			3	4		3				4	3	
95	90	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

Table 1: Basket configuration per shaft

Shaft number	**Sum of places used**	**3x1.15** **3**	**4x1.15** **4**	**5x1.15** **5**
50	12	2	10	0
25	6	1	5	0
70	15	0	8	7
35	9	4	5	0
95	18	0	0	18

Table 2: Sum of types 1.15 per shaft number

Weights	**1 km to kg**	**Weight haspel**	**Km on haspel**
1x1.15	9.265	400	43172.7
3x1.15	27.795	400	14390.9
4x1.15	37.060	400	10793.2
5x1.15	46.326	400	8634.5

Table 3: 1.15 multiwire haspels

4.2.2 8x0.3mm

With the implementation of the new machine, some production of 3x0.3mm and 8x0.3mm will change to 3x0.4, 8x0.4, 3x0.5, 8x0.5, 3x0.6 and 8x0.6mm depending on the specifications of the final cable. Table 4 specifies the transition for the number of wires needed. Every type of final cable uses a fixed distribution of 3x0.3mm and 8x0.3mm wire, with this knowledge one can compute the percentage of 3x0.3mm and 8x0.3mm used in a final cable (streng curs). The booked data 8x0.3mm is given in kilograms but needs to be transformed into meters. When knowing the length of 8x0.3mm booked, the length of 3x0.3mm is linear based on the number of wires needed, see equations (40) - (41).

$$TotalLengthPerOrder = BookedLength_{8x0.3}/Percentage_{8x0.3} \tag{40}$$

$$BookedLength_{3x0.3} = TotalLengthPerOrder * percentage_{3x0.3} \tag{41}$$

$$NewLength_{eightfold} = TotalLengthPerOrder * percentage_{eightfold} \tag{42}$$

$$Newlength_{threefold} = Totallengthperorder * percentage_{threefold} \tag{43}$$

This means that with the information for which final cable the amount is produced one can derive the corresponding demand of 3x0.3mm, without having to collect this data from the ERP system. Following, the total amount of wire used per order of final cable is the sum of 8x0.3, 3x0.3, and 1x0.3mm, or equation (40). Single wire 0.3mm can be disregarded, as the new machine will not produce this type of wire. The $TotalLengthPerOrder$ is needed to compute the weights of eightfold and threefold cables used, based on the new percentages listed in table 4. The new percentage times the $TotalLengthPerOrder$ used yields the new length of eightfold and threefold Haspels needed, see equations (42) - (43). Based on the type of final cable the cables have a diameter of either 0.402, 0.502, or 0.602 mm.

Cable design	Sum	Wires	New construction	Wire size	No. new wires	Old machine			New machine			Old		New	
						8	3	1	8	3	1	percentage 8x	percentage 3x	percentage 8x	percentage 3x
Streng Curs 23x0,303	100438.8754	23	13x0,402	0,402	13	2	2	1	1	1	2	0.696	0.174	0.615	0.231
Streng Curs 25x0,303	113010.3163	25	14x0,402	0,402	14	3	0	1	1	2	0	0.960	0.000	0.571	0.429
Streng Curs 33x0,303	189283.2834	33	19x0,402	0,402	19	4	0	1	2	1	0	0.970	0.000	0.842	0.158
Streng Curs 41x0,303	196349.1523	41	15x0,502	0,502	15	5	0	1	1	2	1	0.976	0.000	0.533	0.400
Streng Curs 43x0,303	212678.5362	43	16x0,502	0,502	16	5	1	0	2	0	0	0.930	0.047	1.000	0.000
Streng Curs 47x0,303	137237.2774	47	17x0,502	0,502	17	5	2	1	2	0	1	0.851	0.085	0.941	0.000
Streng Curs 48x0,303	297849.5771	48	17x0,502	0,502	17	6	0	0	2	0	1	1.000	0.000	0.941	0.000
Streng Curs 51x0,303	187347.0389	51	18x0,502	0,502	18	6	1	0	2	0	2	0.941	0.039	0.889	0.000
Streng Curs 53x0,303	431348.8178	53	19x0,502	0,502	19	6	1	2	2	1	0	0.906	0.038	0.842	0.158
Streng Curs 62x0,303	429170.4567	62	22x0,502	0,502	22	7	2	0	2	2	0	0.903	0.065	0.727	0.273
Streng Curs 64x0,303	1419.4692	64	23x0,502	0,502	23	8	0	0	2	2	1	1.000	0.000	0.696	0.261
Streng Curs 66x0,303	138251.7035	66	24x0,502	0,502	24	8	1	0	3	0	0	0.970	0.030	1.000	0.000
Streng Curs 67x0,303	253971.7877	67	24x0,502	0,502	24	8	1	0	3	0	0	0.955	0.030	1.000	0.000
Streng Curs 74x0,303	9483.1624	74	19x0,602	0,602	19	9	0	2	2	1	0	0.973	0.000	0.842	0.158
Streng Curs 94x0,303	38820.3774	94	24x0,602	0,602	24	11	2	0	3	0	0	0.936	0.043	1.000	0.000
Streng Curs 104x0,303	4397.1779	104	26x0,602	0,602	26	13	0	0	3	0	2	1.000	0.000	0.923	0.000

Table 4: Cable design

To determine the number of Haspels needed, the corresponding weight to each $NewLength$ must be computed. This is done by first converting the $NewLength$ in kilograms and then as a second step dividing it by the weight that fits on a haspel, this yields the number of Haspels needed which can be seen in table 5.

Weights	1 km to kg	1 kg to mtr	Weight haspel	Km on haspel
8x0.3	5.044	198.250	520	103.1
3x0.3	1.892	528.666	520	274.9
8x0.4	9.102	109.861	510	56.0
3x0.4	3.413	292.964	510	149.4
8x0.5	14.180	70.521	505	35.6
3x0.5	5.318	188.056	505	95.0
8x0.6	20.379	49.071	500	24.5
3x0.6	7.642	130.855	500	65.4

Table 5: Weights 0.3, 0.4, 0.5, 0.6mm

4.3 Data cleaning

A constraint to ensure the trustworthiness of data is to have a minimum of 100 data points (Anderson & Gerbing, 1984). Therefore, the data set ranges from 01-01-2022 till 22-11-2023, the date on which the data was collected, and thus is longer than the original time span of one year. The data range is 691 days, but not every day production of cables has taken place due to factory closings. This leads to a total of 619 possible production days.

4.3.1 1.15 multiwire

For 1.15 multiwire, 11 data points are not possible to integrate into the dataset because their shaft number is unclear such that no distribution of wire can be found, 3130 data points are left.

4.3.2 8x0.3mm

90% of booked orders 8x0.3mm correspond to table 4, this means that for 90% of all bookings, the new wire design with diameters 0.4, 0.5, and 0.6mm will be implemented. For the remaining 10%, 0.3mm wire will still be used, these are final cables with a diameter smaller than $10mm^2$, where norm restrictions do not allow for thicker wire diameters. 0.3mm wire will not be produced on the new machine, therefore this data can be disregarded for further analysis.

4.4 Data analysis

The possible total amount of production days is as said 619 days, for this data set however, the count of actual production days is 575 days, found by listing all orders in a demand matrix with products in columns and production days in rows.
With the data collected, filtered, and specified to the demand of products per day, the goal is to determine parameters like e.g. standard deviation of demand and safety stock. To determine those parameters, a distribution needs to be found that fits the demand data per day per product. This step is done in Wolfram Mathematica 13.3 with the $FindDistribution[data, n = 3]$ function. For every product, this function yields three distributions that best fit the demand data per day. To analyze which of the three distributions best fits, the data is further analyzed in Excel with the *Analysis ToolPak Descriptive Statistics* and $Histogram$ functions. For the histogram, the number of bins and width is determined by

$$NumberOfBins \approx \sqrt{n} \tag{44}$$

$$BinWidth = (Max - Min)/\sqrt{n} \tag{45}$$

where n is the number of observations, so the number of production days of a product, $NumberOfBins$ is rounded up, Max is the largest observation and Min is the smallest observation. The $Histogram$ function yields the frequency of data per bin interval, determined by equation (45) from the smallest to the largest observation. The frequency needs to be compared with the expected frequency of the three distributions given by Wolfram Mathematica employing a Chi-Squared test.

4.4.1 Chi-Squared test on the expected frequency for 3x1.15mm

The following data analysis step will be explained for one example: 3x1.15mm wire, and for the other products the conclusion of the test will be discussed. For 3x1.15mm, the three best-found distributions in Wolfram Mathematica are: $GammaDistribution[1.86374, 3.76955]$, $WeibullDistribution[1.2871, 7.23766, 0.335283]$, $LogNormalDistribution[1.65789, 0.821158]$.

Those three distributions with corresponding parameters are used to determine the expected amount of products within the listed bin interval.

$$CDF_{1,x} = GAMMA.DIST(x, 1.86374, 3.76955, TRUE) * n \tag{46}$$

$$CDF_{2,x} = WEIBULL.DIST(x - 0.335283, 1.2871, 7.23766, TRUE) * n \tag{47}$$

$$CDF_{3,x} = LOGNORM.DIST(x, 1.65789, 0.821158, TRUE) * n \tag{48}$$

$$ExpectedFrequency_x = CDF_x - CDF_{previousBin} \tag{49}$$

where CDF = Cumulative Distribution Function and is computed for all x, x = corresponds to the column bin interval in table 6 and $n = 547$. CDF computes the total amount of observations expected from 0 until x for all three distributions. To determine the $ExpectedFrequency$ per bin interval the difference between two consecutive bin intervals is determined. From equations (44) - (45), $NumberOfBins \approx 24$, and $BinWidth = 1.5833$ follows ranging from 0 to 38, the smallest to the largest observation of Haspels produced on a day. The next step is to compute the Error between the expected and observed frequency.

Bin interval	*Frequency*	CDF Gamma	Expected Gamma	x - μ	CDF Weibull	Expected Weibull	CDF LogNormal	Expected LogNormal	Error Gamma	Error Weibull	Error LogNormal
0.00	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1.58	41	46.926	46.926	1.248	54.082	54.082	39.512	39.512	0.748	3.164	0.056
3.17	84	132.053	85.127	2.831	141.285	87.204	147.251	107.739	0.015	0.118	5.231
4.75	100	219.847	87.794	4.415	224.793	83.508	247.058	99.807	1.697	3.257	0.000
6.33	89	297.136	77.289	5.995	297.416	72.622	322.875	75.817	1.775	3.693	2.292
9.50	107	410.190	113.055	9.165	405.910	108.494	418.482	95.607	0.324	0.021	1.358
11.08	44	447.841	37.651	10.748	443.362	37.453	447.821	29.338	1.071	1.145	7.327
12.67	24	475.851	28.010	12.331	471.884	28.522	469.523	21.702	0.574	0.717	0.243
14.25	14	496.366	20.515	13.915	493.212	21.328	485.783	16.261	2.069	2.518	0.314
15.83	10	511.210	14.844	15.498	508.910	15.698	498.121	12.338	1.581	2.068	0.443
17.42	11	521.846	10.637	17.081	520.303	11.393	507.596	9.475	0.012	0.014	0.246
19.00	3	529.408	7.562	18.665	528.467	8.164	514.953	7.357	2.752	3.266	2.581
20.33	2	534.025	4.617	19.998	533.466	4.999	519.902	4.949	1.483	1.799	1.757
20.58	3	534.748	0.724	20.248	534.249	0.783	520.726	0.823	7.162	6.271	5.756
22.17	0	538.499	3.751	21.831	538.302	4.053	525.298	4.572	3.751	4.053	4.572
23.75	4	541.121	2.622	23.415	541.114	2.812	528.952	3.654	0.725	0.502	0.033
25.33	1	542.945	1.825	24.998	543.048	1.933	531.895	2.944	0.373	0.451	1.283
26.92	3	544.211	1.266	26.581	544.365	1.318	534.285	2.389	2.377	2.148	0.156
28.50	0	545.086	0.875	28.165	545.256	0.891	536.238	1.953	0.875	0.891	1.953
30.08	1	545.689	0.603	29.748	545.853	0.597	537.845	1.607	0.261	0.271	0.229
31.67	0	546.104	0.415	31.331	546.251	0.398	539.175	1.330	0.415	0.398	1.330
33.25	3	546.389	0.285	32.915	546.513	0.263	540.283	1.107	25.909	28.508	3.235
34.83	1	546.584	0.195	34.498	546.686	0.173	541.209	0.927	3.327	3.969	0.006
36.42	0	546.717	0.133	36.081	546.799	0.113	541.989	0.779	0.133	0.113	0.779
38.00	2	546.808	0.091	37.665	546.871	0.073	542.647	0.659	40.124	50.927	2.733
More	0							**SUM**	**99.531**	**120.280**	**43.915**
								Chi alpha 0.05	**36.415**		

Table 6: CDF and Chi-Squared test 3x1.15mm

This value is needed for the Chi-Squared test. A Chi-Squared test can be used to determine whether a distribution is suited to fit a data set. The hypotheses for the Chi-Squared test are:
h_0 = There is not enough proof to reject the distribution
h_1 = The distribution can not be used
The test statistic is:

$$X^2 = \sum ((Frequency_{Observed} - Frequency_{Expected})^2 / Frequency_{Expected}) \tag{50}$$

with alpha = 0.05 and 24 degrees of freedom, the critical value is:

$$X^2_{(0.05,24)} = CHISQ.INV.RT(0.05, 24) = 36.415 \tag{51}$$

when $X^2_{(0.05,24)} \geq X^2$ accept h_0, otherwise reject h_0

$$X^2_{(0.05,24)} < X^2_{LogNormal} = 43.914 \tag{52}$$

The decision, from which distribution to choose from, is made based on the first distribution for which X^2 is smaller than the critical value $X^2_{(\alpha,df)}$, assuming that Wolfram Mathematica's order of best-found distributions is correct and the Chi-Squared test is a good test statistic to verify the result from Wolfram Mathematica. The test statistic Error value for the LogNormal Distribution is the lowest of all distributions but still lies above the rejecting value, see table 6. Therefore by means of the Chi-Squared test, the distribution must be

rejected. With a smaller value of 0.007 for alpha the h_0 hypothesis can be accepted and therefore there is not enough proof to reject the LogNormal distribution, as

$$X^2_{(0.007,24)} = 44.321 \geq 43.914. \tag{53}$$

This shows the danger of the test moving with values for alpha: The smaller alpha, the larger the critical value, thus the more likely it is to accept h_0, when in fact h_0 should be rejected. This is a type II error. An academic standard is to use an alpha level of 0.05 to prevent type I and type II errors from happening (Moore, 2009). In this case, a distribution is needed to continue determining the parameters and therefore the assumption is made that the LogNormal distribution fits the observed frequency accurately enough. When comparing the Error values of all distributions given by Wolfram Mathematica, the LogNormal distribution performs by far the best, see table 6. This can also be observed when analyzing the visual representations, see figure 6. The other distributions deviate more from the observations on the right tail of the frequency histogram.

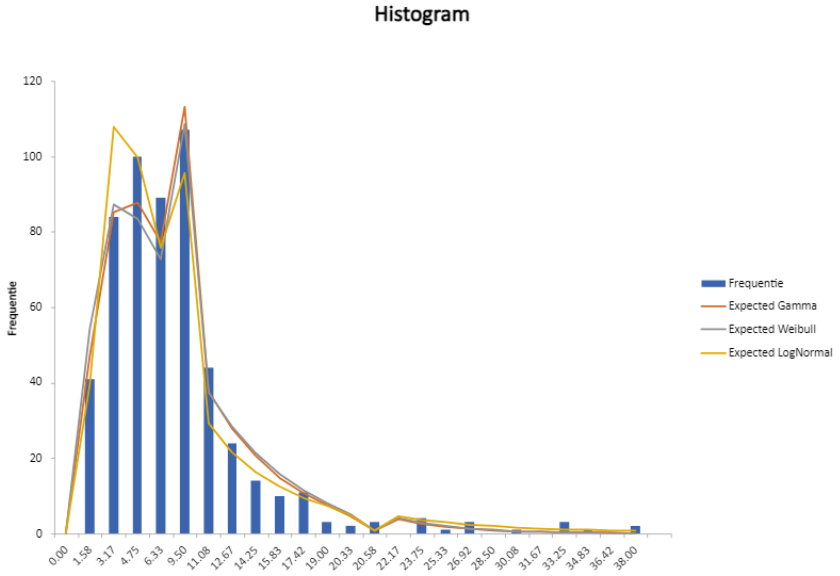


Figure 6: Observed and expected frequencies 3x1.15mm

This means that the LogNormal Distribution can represent product 3x1.15mm with parameters $[\mu = 1.65789, \sigma = 0.821158]$. The mean and variance of this distribution are given by

$$mean = exp(\mu + \sigma^2/2) \tag{54}$$

$$variance = exp(2\mu + \sigma^2)(exp(\sigma^2) - 1) \tag{55}$$

$$standardDeviation = \sqrt{variance} \tag{56}$$

yielding a mean of 7.352, a variance of 56.156, and a standard deviation of 7.493 during days with production of 3x1.15mm.

4.4.2 Results of the Chi-Squared test for the other products

The results of the other products are summarized in table 7. A detailed approach to the computation of the results is listed in the appendices, see 9.4.

Product	Distribution	Mean	Variance	Std. Dev.
4x1.15mm	LogNormal	31,227.000	1,339.530	366.000
5x1.15mm	Weibull	11.73	53.112	7.288
3x0.4mm	Exponential	1.134	1.286	1.134
8x0.4mm	Exponential	8.328	69.347	8.328
3x0.5mm	Exponential	3.169	10.043	3.169
8x0.5mm	Weibull	33.034	369.756	19.229

Table 7: Results of the Chi-Squared test

4.5 Limitations of research design

Limitations are that the demand is only based on historical demand. Forecasted future demand is not available and therefore not implemented. Another limitation is that the problem cannot be solved optimally as the model is NP-hard (see chapter 2) but is solved utilizing a heuristic. The accurate representation of reality is limited when implementing uncertainty in a model that assumes fixed demand instead of stochastic demand. In this context, uncertainty cannot be solely regarded as a factor to be multiplied with (section 2.4) but is limited to this representation.

5 Determining the model parameters

In the previous chapter 4, I determined the mean and standard deviation per product. To further work towards the goal of finding lot sizes and production sequences per day, the findings from chapter 4 need to be utilized when determining the parameters: Demand, mean, and standard deviation of the actual production days, time per Haspel, lead time, safety stock, reorder point, maximum inventory level and machine conversion times. Those parameters are necessary to solve the dynamic programming model and heuristic (3.2).

5.1 Demand

In chapter 4, I determined the demand per day for all products to produce, this is our first parameter. Demand is presented in an integer amount of Haspels and can be found the the appendices, see table 19.

5.2 Mean and standard deviation

The mean and standard deviation of demand per product from chapter 4 are based on the days during which production occurs. To further determine the parameters needed, the mean and standard deviation per product of the actual production days, which are 575, need to be computed.

$$sample.mean = mean * d_{Prod.}/575 \tag{57}$$

where *mean* is the mean computed in chapter 4 per product and $d_{Prod.}$ is the number of observations per product. The *mean* only covers the average demand of Haspels during production days but in reality the production days are spread out over a longer time horizon (575 days), meaning that there are days with no demand of a product. Those days also need to be taken into account when e.g. computing the safety stock. The same holds for the standard deviation.

$$sample.std.dev. = \sqrt{\sum (x-\mu)^2/(n-1)} \tag{58}$$

Applied to this situation, where

$$\sum (x-\mu)^2 = (mean - sample.mean)^2 * d_{Prod.} + (0 - samaple.mean)^2 * (n - d_{Prod.}) \tag{59}$$

and x is either 0 or *mean*, the new values are listed in table 8. As explained in section 9.4, 3x0.6mm and 8x0.6mm do not have a distribution matched and are considered as no safety stock products, this decision is furthermore justified with a sample mean ≤ 1.

Products	**Production days**	**Mean on production days**	**Sample mean over total production**	**Std.dev on production days**	**Sample std.dev total production**
3x1.15	547	7.352	6.748	7.494	2.021
4x1.15	555	31.227	29.079	36.600	7.910
5x1.15	104	11.248	1.963	11.248	4.273
3x0,403	139	1.134	0.265	1.134	0.480
8x0,403	139	8.328	1.942	8.328	3.525
3x0,503	191	3.169	1.016	3.169	1.480
8x0,503	368	33.034	20.397	19.229	16.068
3x0,603	6	0.777	0.008	X	X
8x0,603	21	20.675	0.728	X	X

Table 8: Mean and standard deviation of products

The other option to determine the mean and standard deviation is to let Wolfram Mathematica find a distribution over 575 days, thus with 0 values included. In this case, the *FindDistribution* function splits the data into multiple intervals with different distributions found for each interval. 0 is not seen as a standalone interval, the data is not split

into an interval with only 0 and another interval covering the rest, although this is the case in reality. Either something is produced or nothing. Another downside is that it is much more complex to determine the expected frequency per distribution as this is dependent on the bin intervals and not on the distribution intervals currently. Therefore, it is more logical to determine a distribution over the observed frequencies (without 0) and then re-determine the mean and standard deviation by including the no production data and using the *sample.mean* and *sample.std.dev.* formulas.

5.3 Time per Haspel and lead time

The time per Haspel t is given in minutes by

$$t = (weight/(no.wires * 1/4 * \pi * diameter^2 * 8.92/1000)/speed/60) + ChangeTime \quad (60)$$

where the function variables are listed in tables 9 and 10. Time per Haspel is dependent on the machine speed, which is different for the two production strategies, therefore the time per Haspel is different for both production strategies.
When knowing the time per Haspel, the lead time lt per product can be determined, also listed in tables 9 and 10.

$$lt = t * sample.mean \quad (61)$$

Products	1 km to kg	Weight haspel	Km on haspel	Speed m/s	Haspel change time (min)	No. Wires	Time per haspel (min)	Time per haspel (day)	Sample mean over total production	Lead Time (days)
3x1.15	27.795	400	14.391	11	3	3	24.80	0.0172	6.748	0.116
4x1.15	37.060	400	10.793	11	3	4	19.35	0.0134	29.079	0.391
5x1.15	46.326	400	8.635	11	3	5	16.08	0.0112	1.963	0.022
3x0,403	3.413	510	149.412	32	3	3	80.82	0.0561	0.265	0.015
8x0,403	9.102	510	56.029	32	3	8	32.18	0.0223	1.942	0.043
3x0,503	5.318	505	94.968	32	3	3	52.46	0.0364	1.016	0.037
8x0,503	14.180	505	35.613	32	3	8	21.55	0.0150	20.397	0.305
3x0,603	7.642	500	65.427	29	3	3	40.60	0.0282	0.008	X
8x0,603	20.379	500	24.535	29	3	8	17.10	0.0119	0.728	X

Table 9: Time per haspel and lead time on single decks

Products	1 km to kg	Weight haspel	Km on haspel	Speed m/s	Haspel change time (min)	No. Wires	Time per haspel (min)	Time per haspel (day)	Sample mean over total production	Lead Time (days)
3x1.15	27.795	400	14.391	22	3	3	13.90	0.0097	6.748	0.065
4x1.15	37.060	400	10.793	22	3	4	11.18	0.0078	29.079	0.226
5x1.15	46.326	400	8.635	22	3	5	9.54	0.0066	1.963	0.013
3x0,403	3.413	510	149.412	64	3	3	41.91	0.0291	0.265	0.008
8x0,403	9.102	510	56.029	64	3	8	17.59	0.0122	1.942	0.024
3x0,503	5.318	505	94.968	64	3	3	27.73	0.0193	1.016	0.02
8x0,503	14.180	505	35.613	64	3	8	12.27	0.0085	20.397	0.174
3x0,603	7.642	500	65.427	58	3	3	21.80	0.0151	0.008	X
8x0,603	20.379	500	24.535	58	3	8	10.05	0.0070	0.728	X

Table 10: Time per haspel and lead time on double decks

5.4 Safety stock, reorder point and maximum inventory level

The next parameters to be determined are safety stock s, reorder point ROP and maximum inventory level $Max.Inv.$ given by the following formulas

$$s = Z * sample.std.dev. * \sqrt{LT} \tag{62}$$

where Z=1.645 with 95% service level,

$$ROP = lt * sample.mean + s \tag{63}$$

$$Max.Inv. = ROP + sample.mean - Min.Consumption * Min.LeadTime \tag{64}$$

which reduces to

$$Max.Inv. = s + sample.mean \tag{65}$$

as we can assume that $Min.Consumption$ * $Min.LeadTime$ = $sample.mean$ * lt. The results from these formulas are listed in tables 11 and 12.

Products	Sample mean over total production	Std.dev on production days	Sample std.dev total production	Lead Time (days)	Safety stock	Reorder Point	Maximum inventory level	Max inventory standardized multiples of 4 (1 rack)	Safety stock standardized
3x1.15	6.748	7.494	2.021	0.116	1.13	1.92	7.882	8	2
4x1.15	29.079	36.600	7.910	0.391	8.13	19.50	37.213	40	9
5x1.15	1.963	11.248	4.273	0.022	1.04	1.08	3.003	4	2
3x0,403	0.265	1.134	0.480	0.015	0.10	0.10	0.361	4	1
8x0,403	1.942	8.328	3.525	0.043	1.21	1.29	3.150	4	2
3x0,503	1.016	3.169	1.480	0.037	0.47	0.51	1.484	4	1
8x0,503	20.397	19.229	16.068	0.305	14.60	20.83	34.999	36	15
3x0,603	0.008	X	X	X	X	X	X	0	0
8x0,603	0.728	X	X	X	X	X	X	0	0

Table 11: Safety stock, reorder point and maximum inventory level on single decks

Products	Sample mean over total production	Std.dev on production days	Sample std.dev total production	Lead Time (days)	Safety stock	Reorder Point	Maximum inventory level	Max inventory standardized multiples of 4 (1 rack)	Safety stock standardized
3x1.15	6.748	7.494	2.021	0.065	0.85	1.29	7.597	8	1
4x1.15	29.079	36.600	7.910	0.226	6.18	12.74	35.261	36	7
5x1.15	1.963	11.248	4.273	0.013	0.80	0.83	2.764	4	1
3x0,403	0.265	1.134	0.480	0.008	0.07	0.07	0.334	4	1
8x0,403	1.942	8.328	3.525	0.024	0.89	0.94	2.835	4	1
3x0,503	1.016	3.169	1.480	0.02	0.34	0.36	1.356	4	1
8x0,503	20.397	19.229	16.068	0.174	11.02	14.57	31.418	32	12
3x0,603	0.008	X	X	X	X	X	X	0	0
8x0,603	0.728	X	X	X	X	X	X	0	0

Table 12: Safety stock, reorder point and maximum inventory level on double decks

The computed values for safety stock and maximum inventory are standardized and rounded up to integers given in the last two columns of tables 11 and 12. The inventory is stored in racks of four, therefore the maximum inventory is a multiple of four. In the research questions, see section 1.11.2, I stated that the inventory capacity will decrease to 90% of its original capacity. Currently, the sum of inventory for this type of product is 224 Haspels, 90% of this yields of around 201 Haspels. The sum of maximum inventory determined, see tables 11 and 12, is 100 for single and 80 Haspels for double decks, this lies far below the new upper bound.

5.5 Machine conversion times

The machine conversion times are another important parameter listed in tables 13 and 14

	3x1.15	4x1.15	5x1.15	3x0,403	8x0,403	3x0,503	8x0,503	3x0,603	8x0,603
3x1.15	0	15	15	30	30	30	30	30	30
4x1.15	15	0	15	30	30	30	30	30	30
5x1.15	15	15	0	30	30	30	30	30	30
3x0,403	15	15	15	0	15	30	30	30	30
8x0,403	15	15	15	15	0	30	30	30	30
3x0,503	15	15	15	30	30	0	15	30	30
8x0,503	15	15	15	30	30	15	0	30	30
3x0,603	15	15	15	30	30	30	30	0	15
8x0,603	15	15	15	30	30	30	30	15	0

Table 13: Changeover time on single decks

	3x1.15	4x1.15	5x1.15	3x0,403	8x0,403	3x0,503	8x0,503	3x0,603	8x0,603
3x1.15	0	15	15	180	180	180	180	180	180
4x1.15	15	0	15	180	180	180	180	180	180
5x1.15	15	15	0	180	180	180	180	180	180
3x0,403	45	45	45	0	15	30	30	30	30
8x0,403	45	45	45	15	0	30	30	30	30
3x0,503	45	45	45	30	30	0	15	30	30
8x0,503	45	45	45	30	30	15	0	30	30
3x0,603	45	45	45	30	30	30	30	0	15
8x0,603	45	45	45	30	30	30	30	15	0

Table 14: Changeover time on double decks

5.6 Machine capacity

To only produce a realistic amount of Haspels, there needs to be a restriction on the capacity of the machine, which is the maximum time the machine can produce on a production day. At TKF, the production is 24 hours per day, this means that the maximum machine capacity is $24 \cdot 60 = 1440$ minutes.

5.7 Uncertainty

As one of the sub-research questions in section 1.11.2 states, it needs to be investigated how uncertainty can implemented in the model. There is uncertainty in terms of machine breakdown and operator's availability due to sickness see figure 3, to account for this type of uncertainty one option is to include probabilities for machine breakdown and employee shortage or to already reduce the machine capacity to e.g. only 95% (Rockafellar, 2001), but when looking at my data it can be observed that there is a discrepancy between the observed production days and possible production days. There are 575 production days over 691 possible days such that the percentage of days left is $1 - (575/619) = 7.2\%$, which are days where production could be set up in case of irregularities like machine breakdown or sickness of employees. I assume that this percentage is enough to account for uncertainties in my model.

5.8 Penalty cost

When the inventory of a product i on a day is below the safety stock of i, a penalty cost is multiplied with the penalty, see constraint (4) and the objective function (2). This cost is set at 6 for all i, as this value yields the desired stock levels, preferably above safety stock when machine capacity allows for it.

6 Finding a solution

Now the models to solve have been formulated (see section 3.2) and all parameters are defined (see section 5), we can start the solving procedure. The first step is to determine initial production quantities, then we determine initial sequences which are evaluated in the third step, where we recompute the quantities. Finally, the model is further improved with a heuristic that improves the setup times between days and deletes setups for which production quantities are low, if possible. After each step, the results are visually presented. A comparison of the results from steps 3 and 4 with the results from the case if a planner would schedule the orders is done in chapter 7. The current demand spans over 691 days, with 575 actual production days. In the end, it is not of importance to find a solution for all 575 production days, to be able to compare the two production strategies with respective total production times. I assume that the most recent 30 days from the data set are enough, see table 19 in the appendices. This particular time frame contains a diverse mix of product demands per day, providing a valid portrayal of the whole dataset.

6.1 Step 1 - Determine initial production quantities per period

To determine the initial quantities, the problem stated in section 3.2.1 is implemented in AIMMS and solved with CPLEX. For Single Decks, no solution can be found, because of the machine capacity restriction: Demand for some days is higher than what the machine can produce plus the maximum inventory, this is the case for product 4x1.15mm. One option is to increase the maximum inventory, but this leads to overfull inventory on days when demand is low. The other option applied is to spread out the peaks in demand over two days when demand exceeds what can be delivered on a day. The spread occurs over the day of observation and the previous day and the adjusted demand can be found in table 20 in the appendices. This indeed, yields a solution, see figure 7. For Double Decks, a solution is found without adjustments needed, see figure 8. It is immediately visible that the demand on single decks is much more spread out compared to the double decks' production strategy, on the one hand, due to the manual spread out of demand but on the other hand mostly because the production speed is lower (see table 9) such that demand must be produced beforehand and put on stock to satisfy demand. Another proof for this is that for double decks the machine is not set up on day-21 as demand can be satisfied without producing on this day.

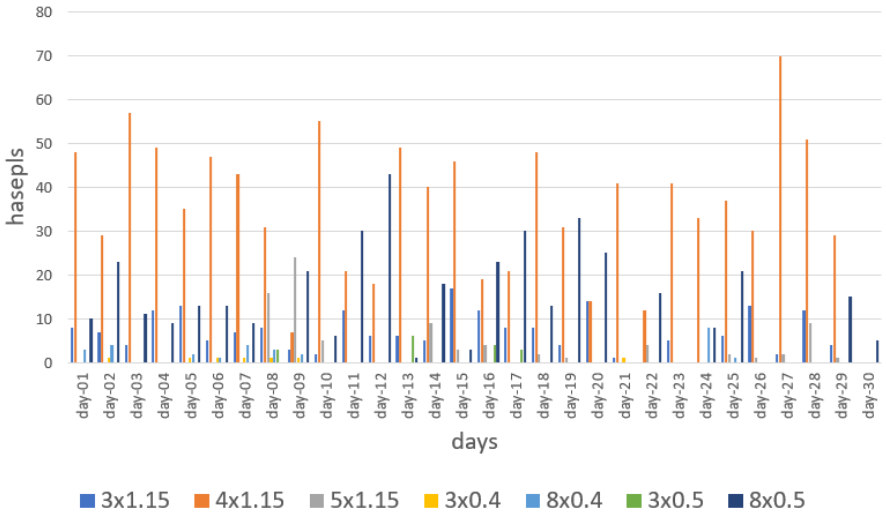


Figure 7: Initial quantities on single decks

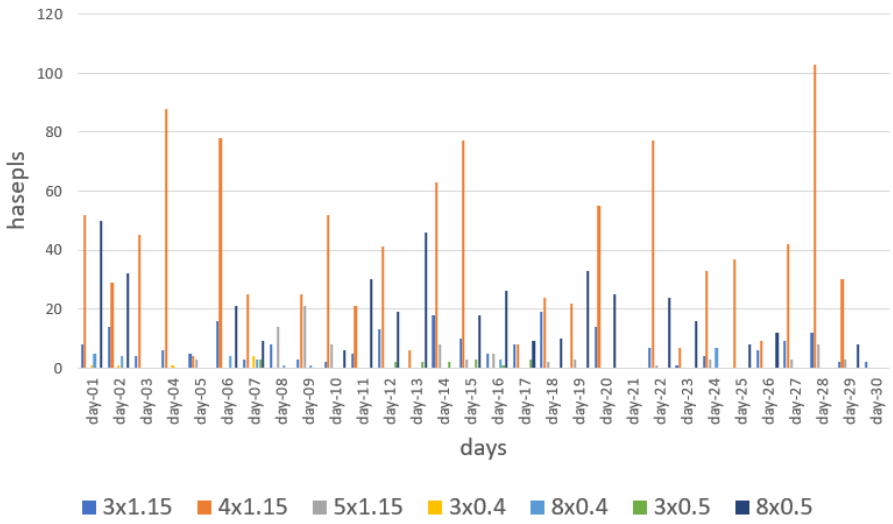
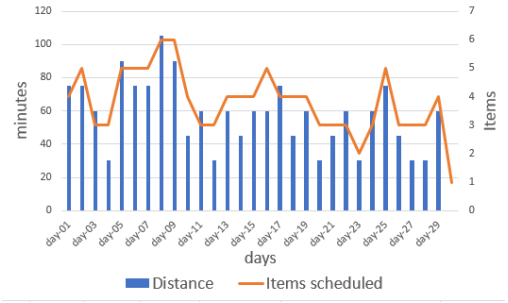


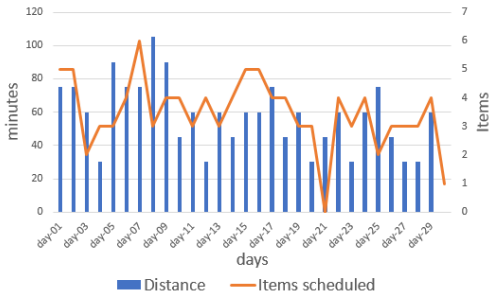
Figure 8: Initial quantities on double decks

6.2 Step 2 - Determine initial sequences

In this step, the initial quantities from section 6.1 are used to formulate setup matrices, explained in section 3.2.2, for the quantities to be produced per day which are solved in Python 3.12, specifically with a library written in pure Python for solving typical TSPs. The code loops over all production days and finds the shortest path solution between setups for all items to be produced on a day. As explained in section 3.2.2, the last setup of a day is the starting setup of the next day represented in row 0. Column 0 consists of entire zeros to restrict the TSP from going back to the start but end at the last product to schedule. A listing of the code can be found in the appendices, see 9.4



(a) Single decks



(b) Double decks

Figure 9: Setup times per day after solving the TSP matrices

When analyzing the figures 9a and 9b, it becomes clear that on double decks, the initial setup time per day is higher compared to single decks represented on the left hand horizontal axis, even tough the items to schedule represented on the right hand side horizontal axis are on average slightly lower (3.8 versus 3.467 items) see figure 9.

6.3 Step 3 - Sequence evaluation

Having determined the initial sequences, the following step is to solve the DP from section 3.2.3 with the initial quantities and sequences as input. This DP yields new production quantities per day the machine is set up from the predetermined sequences and is solved again in AIMMS with CPLEX. For single decks, no solution can be found because the starting inventory and machine capacity cannot meet the demand at day-01 for 4x1.15. Therefore, the starting inventory is increased by 6 Haspels to a total 12 such that the starting inventory and machine capacity can meet demand. As it can be seen in figure 10, the time needed to produce those 6 Haspels is later on added again to the production time, which results in producing more than technically possible. The sum of production and setup time is higher than the machine capacity for day-01, but over the total of 30 days, the sum of production time and setup time is still below the sum of maximum machine capacity over 30 days. When applying this case in reality, it would mean that the production would be delayed by this overtime. The figures 10 and 11 show the total time needed to satisfy demand, but when looking back at the recursion formula from step 3 in section 3.2.3, also penalty costs are included. To account for the delay in production caused by the overtime, another penalty needs to be incurred, which is letting the overtime count double.

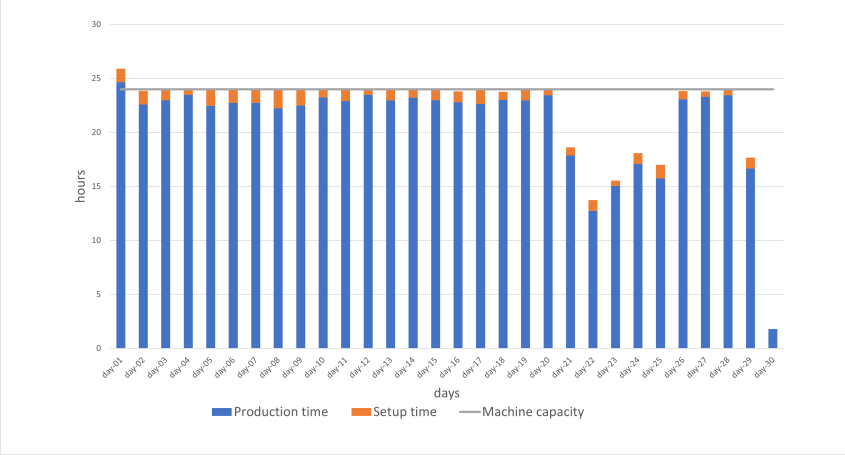


Figure 10: Step 3 - Production and setup times per day on single decks

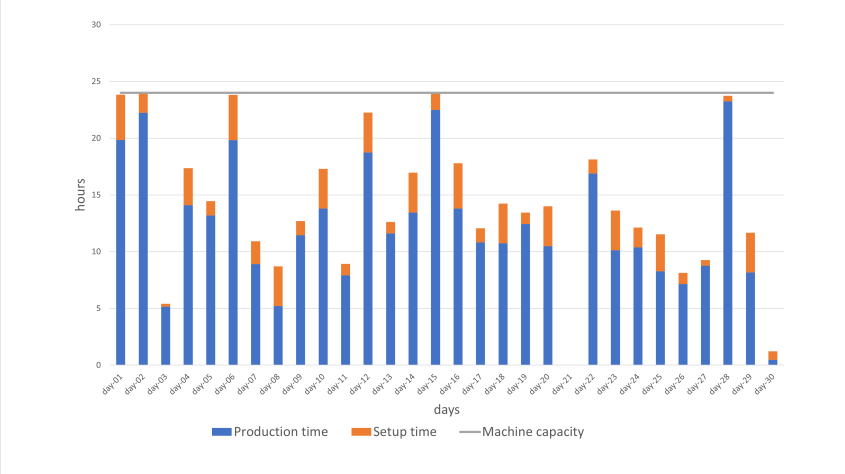


Figure 11: Step 3 - Production and setup times per day on double decks

For single decks, the time needed to produce orders is 654.93 hours, with inventory penalty costs of 11.9 hours and 1.94 hours for overtime. The total production time with costs is 668.77 hours. There are no extra overtime costs incurred for double decks. Demand can still be satisfied without production on day-21. Just the time needed to produce orders is 424.19 hours, with inventory penalty costs of 3.2 hours. The total production time with costs is 427.39 hours.
When comparing the setups scheduled in the TSP sequences with the output of the DP of step 3, there are 9 setups for single decks and 11 setups for double decks, for which the machine is set up, but the production quantity is zero. This indicates that Step 4 is needed to further improve the sequences by rescheduling and deleting items.

6.4 Step 4 - Sequence improvement

In this last step, I will improve the sequence, by applying the heuristic described in figure 5. For single decks in total, I was able to delete 19 setups for which the production quantity was < 3 and improve two sequences such that the setup between those two stages became zero, see figure **??**, as there existed a product i in both sequences which could be scheduled as last and first item of the two respective sequences. The number of iterations of the heuristic for single decks was three. After this, the setups scheduled in the TSP matrices were identical to the re-optimization setups. There were no further setups that could be deleted, for which the solution was feasible. In total for double decks, I was able to delete 28 setups and improve four sequences for the same reasons as for single decks, see figure **??**. Also here, the number of iterations of the heuristic was three. There were no further setups that could be deleted, for which the solution was feasible.

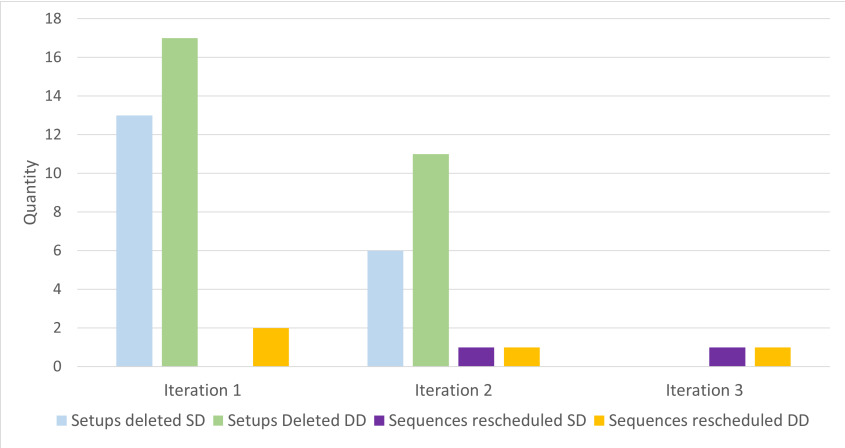


Figure 12: Setups deleted (Move 1) and sequences rescheduled per iteration(Move 2): SD: Single Decks, DD: Double Decks

For single decks, the time needed to schedule orders is 649.83 hours. With inventory penalty costs of 11.5 hours and 1.94 hours for overtime, the total production time with costs is 661.27 hours. The time needed per day for this strategy can be seen in figure **??**.

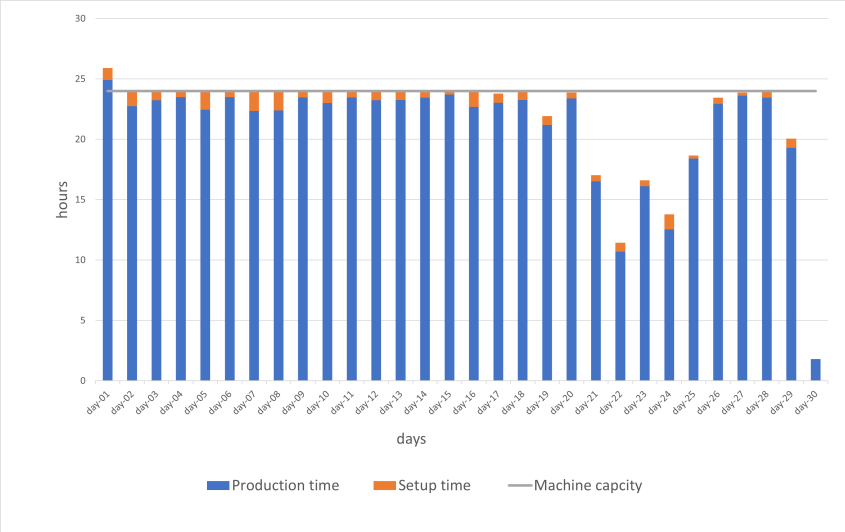


Figure 13: Step 4 Production and setup times per day on single decks

For double decks, the time needed to just schedule the orders is 403.44 hours. With inventory penalty costs of 6.95 hours, the total production time with costs is 410.39 hours. The time needed per day for this strategy can be seen in figure 14.

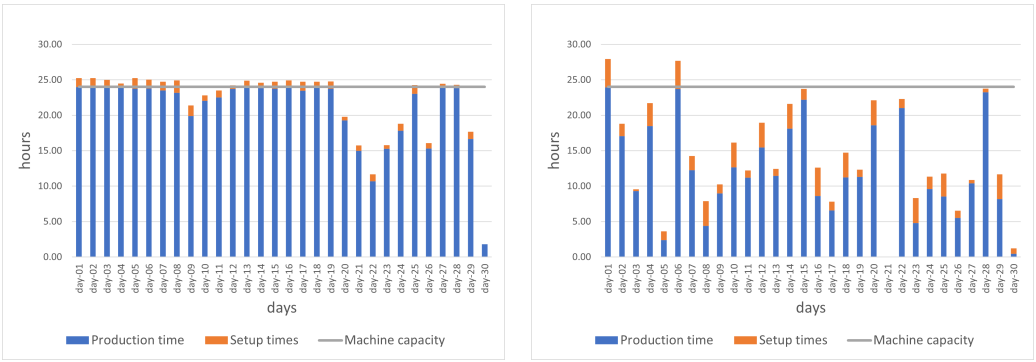
Figure 14: Step 4 Production and setup times per day on double decks

7 Experiments and further analysis of results

In this chapter, I will analyze the performance of the heuristic by comparing it to a situation where a planner would schedule the orders. Here, performance is based on just the time needed to schedule the orders, the total production time, and setup times. Next, I will analyze which of the two production strategies performs best time-wise and financially. After this, I will analyze which implications the results of the best strategy have for standardized lot sizes.

7.1 Total production time if a planner would schedule the orders

If planners would schedule the orders, their approach would be similar to the first two steps of the solving approach: They schedule to fulfill demand and to fill up the inventory. Additionally, they schedule ahead for the next 24 hours, as steps 1 and 2 of the solving approach do. This leads to overtime, meaning production delays. But their solution is still feasible as the sum of the production and setup time, which is 655.58 hours for single decks and 424.19 hours for double decks is still lower than the sum of the maximum machine capacity which is $30 \times 24 = 720$ hours. As more time is needed than available on some days, the overtime will be seen as a penalty cost and is therefore counted double like the case in section 6.3. For single decks, this overtime penalty is 14.55 hours and the penalty for inventory being below safety stock is 10.3 hours, this leads to a total production time with costs of 680.42 hours. For double decks, the overtime penalty is 7.64 hours and the inventory penalty is 2.4 hours, therefore the total production time with costs is 434.23 hours.



(a) Single decks

(b) Double decks

Figure 15: Production and setup times if planners would schedule the orders

7.2 Comparison of the planners and heuristic times

Now we have determined the scheduling time for planners and computed near-optimal solutions in chapter 6, we can compare both cases and analyze how the heuristic performs. In tables 15 and 16 we can see, that the improvement of the last step of the heuristic SP4 is fairly low when looking at the production and setup time, and the total production time. This is because the improvement steps do not influence the production time but only setup time and penalties incurred. No matter how the orders are scheduled, the time needed to produce the orders remains almost unchanged. The row production time shows, that there are some discrepancies, those can be traced back to the differences in inventory at the end of day-30. The models are not forced to end with a specific inventory level such that one model can produce more or less compared to another if it weighs up against the inventory penalty or reduced setup time or is related to maximum machine capacity. The percentage of time needed for the production of orders is approximately 97% for single decks and around 88% for double decks in the case of the last step, see figures **??** and 14. This means that only 3% and 12% of time for single and double decks respectively could be optimized. For further analysis, I will therefore disregard the production time and only look at elements that could be improved. For both single decks and double decks, the setup time can be decreased drastically due to sequence optimization and setup deletion by the sequence improvement heuristic. In the case of penalties, the behavior for single and double decks is different.
In table 15 we can see that for SP3, the penalties are decreased by 44.32%, related to lower costs incurred for overtime usage. The slight decrease in penalties from SP3 to SP4 is related to better coordination of setups that leads to lower inventory penalty costs. Combining setup times and penalties, the total decrease is 32.47%.

Variables (in hours)	Planners	SP3	Improvement	SP4	Improvement
Total production time	680.42	668.77	1.71%	663.26	2.52%
Production and setup time	655.58	655.93	0.10%	649.83	0.88%
Production time	627.58	626.93	0,00%	627.58	0,00%
Setup time	28	28	0.00%	22.25	20.54%
Penalties	24.85	13.83	44.32%	13.43	45.93%
Inventory penalty	10.30	11.90	-15.54%	11.50	-11.65%
Overtime penalty	14.55	1.93	86.70%	1.93	86.70%
Setup time and penalties	52.85	41.83	20.84%	35.68	32.47%

Table 15: Comparison of times on single decks

Variables (in hours)	Planners	SP3	Improvement	SP4	Improvement
Total production time	434.23	427.39	1,58%	410.39	5.49%
Production and setup time	424.19	424.19	0,00%	403.44	4.89%
Production time	359.69	359.69	0,00%	355.94	0,00%
Setup time	64.50	64.50	0,00%	47.50	26.36%
Penalties	10.04	3.20	68.13%	6.95	30.78%
Inventory penalty	2.40	3.20	-33.33%	6.95	-189.58%
Overtime penalty	7.64	0	100%	0	100%
Setup time and penalties	74.54	67.70	9.18%	54.45	26.95%

Table 16: Comparison of times on double decks

For penalties incurred in double decks, we first observed a decrease at SP3 related to no overtime usage anymore. Then, at SP4, we see an increase in inventory penalties, because the setup times are decreased and this decrease in setup time outweighs the increase in penalty costs. The question, of why we do not observe this increase in penalties for single decks, may arise during the comparison of results. This is simply related to the fact that setup times for single decks can not be reduced as drastically as for double decks due to

limited machine capacity which can be seen figure 15a. As the decrease in setup time is less significant compared to double decks, we do not observe an inventory increase but a slight decrease. When we combine setup time and penalties for double decks we can observe an improvement of 26.95%.
In figure 3 and in the research questions, see section 1.11.1, I stated the goal that the total production time can be optimized. With total production time being defined as the sum of production time, setup time, and penalties as is done for the objective functions in chapter 3 and the results in chapter 6, the improvement percentages are 2.52% and 5.49% in tables 15 and 16 respectively. But as explained above, the production time can not be optimized, we just have to meet the demand. So, suppose we disregard the almost constant variable production time and only take into consideration elements that are to be optimized which are setup time and penalties. In that case, reduction percentages are 32.47% and 26.95% for single and double decks respectively. This means, that there is available evidence that supports the effectiveness of the heuristic in its functioning, as indicated by the improvement percentages.

7.3 Financial analysis on which production strategy performs best

The final values for setup time and production time can be used to determine the financial implications of each production strategy. Table 17 denotes the price per minute it costs to run a machine during production or to set up a machine. The setup cost is also based on the amount of operators needed to set up the machine. Due to reasons of confidentiality, the price per minute cost presented is multiplied by a factor only known to people within the organization. This means that the costs to be discussed are different in reality, but the implications and advice presented to TKF remain unchanged.

	Single decks		Double decks	
	Production	Setup	Production	Setup
Price per min.	1.265	0.486	1.265	1.474
Price * time	794.137	10.824	450.401	70.025
Total price	804.961		520.426	

Table 17: Cost price analysis over the demand horizon of 30 days

The setup time price per minute for double decks is more expensive compared to single decks because more operators are needed to set up the machine. When just looking at the time needed to schedule the orders, the single decks production strategy seems attractive, because setup times are lower, and the machine can meet all demand over 30 days. But when we add the costs related to each production strategy it becomes clear that it is cheaper to operate on double decks even though more time is needed to setup the machine. The higher production output and thus lower production cost over 30 days simply outweighs the extra cost related to more operators needed and longer setups between orders. Summarizing this financial analysis, we can say that producing on double decks is the cheapest option.

7.4 Standardized lot sizes from the output of SP4

Based on the financial evidence, that producing on double decks is cheaper, we can determine standardized lot sizes see 18, which is the median of the observed production quantity per product i.

Product	3x1.15	4x1.15	5x1.15	3x0.4	8x0.4	3x0.5	8x0.5
Median	10	42	5	4	7	6	29

Table 18: Standardized lot sizes over the demand horizon of 30 days

8 Answers on research questions

Now that we have obtained all the necessary results, we can provide an answer to the question from section 1.6 by briefly summarizing the findings of the thesis and addressing each research question.

8.1 Answers on main research questions

1. Which production policy (1 fixed deck vs 2 open decks) yields a better performance in terms of the lowest total production time and which batch sizes result from this?

   Double decks has the lowest total production time and is also cheaper financially speaking, see figures **??** and 14 and section 7.3. The batch sizes that result from this can be found in table 18.

2. Which mathematical heuristic fits the goals to reduce the total production time and to determine optimal lot sizes in a manufacturing environment? (Related to finding a correct mathematical framework method)

   This research is based on the framework of Laguna (1999) and is adjusted to the specifications of the problem described throughout chapter 1. It reduces setup times and creates a near-optimal solution, see figures **??** and 14. The goal to reduce total production time is achieved of which the largest improvement is observed in the combination of setup times and penalties, see table 16.

3. How to minimize the total production time considering the input constraints (from no policy to approximation of minimal total production time) for the newly purchased machine? (Related to the correctness of the model equations)

   To minimize total production time, step 4 of the heuristic is the most significant where we optimize sequences and delete setups such that production gets more clustered and the setup time and number of setups are reduced, see figure 5 for the theoretical approach and figures **??** and 14 for the solution.

8.2 Answers on sub-research questions

4. What are the safety stock levels needed, considering a 95% service level, based on demand for wires drawn on the new machine?

   The safety stock levels based on the double decks production strategy can be found in table 12.

5. What is the demand per type of wire on the new machine?

   The demand per type of wire is determined throughout chapter 4 and can be found in the appendix, see table 19. For single decks the adjusted demand from table 20 is used.

6. What is the maximum possible mix of inventory of wire drawn, assuming a 10% shrinking inventory capacity, on the new machine (from 100% inventory storage to 90%)

   The maximum inventory determined is 96, see table 12. This is far below the current inventory size of 201 Haspels.

7. How to implement uncertainty in a dynamic programming model and heuristic?

   Uncertainty is implemented in the sense that there are production days with no demand registered that account for days that can not fulfill demand due to machine breakdown or sickness of employees, see section 5.7 for further explanation.

9 Conclusion

Based on the results of this bachelor's thesis, I can give recommendations to TKF and advice on further research to be executed. Before that, I state the contributions of this thesis on research and its limitations.

9.1 Contributions to research

My contribution to the research on CLSDPs is the development of a heuristic, see figure 5 that improves setup times and therefore also total production time. This heuristic is specific to the case, where inventory below safety stock is treated as a penalty instead of the case where inventory is treated as a holding cost and the objective is to reduce production time.

9.2 Limitations

This thesis considers only historical demand, instead of forecasted demand, see section 4.5 for further explanation. Furthermore, the heuristic makes assumptions on which items to consider for improvement. This might reduce the set of all improvements that could be made, such that there might exist a solution that yields a lower total production time.

9.3 Recommendations

My first recommendation for TKF, is to use the double decks production strategy as I have proven that it is faster and cheaper than single decks to satisfy demand. Furthermore, based on the double decks strategy, I recommend using the accompanying parameters for safety stock and maximum inventory to use for the actual inventory management policy. The new maximum inventory needed is 50% of the size of the current inventory and demand can still be satisfied, such that the area not needed for inventory anymore can be used for other projects that need more surface, which was not available beforehand. Based on the results of the heuristic, I recommend clustering demand to avoid production setups for small production quantities whenever possible. This improves total production time and reduces costs. With these points mentioned the machine can be successfully implemented from a supply chain perspective.

9.4 Further research

Regarding further research, there are a few points that can be investigated. First, it can be further analyzed how the production schedule would change if demand is not scheduled per day but per week. This would lead to larger production quantities of a product certainly, but the specific impact on safety stock, maximum inventory, lead time, and total setup time needs to be investigated. Second, it can be analyzed if the result of the total production time changes significantly if Laguna's Tabu Search heuristic is employed. And if so, how the total production time is further improved. Third, in the light of automation, the possibilities for scheduling orders automatically based on the methods from Lagunas paper and my research can be investigated.

Appendix I: Background - Purchasing motivation of the new machine

The core problem before the investment of the new machine, that is outside of the scope of this research, can be traced back to the issue that the current production steps are not combined into one, see Figure 16. This leads to inefficient production and a low degree of automation on the single machines compared to what could be possible as explained above. My research, however, does not focus on the reasons for the purchase but on developing a scheduling policy, it has already been purchased. For the sake of completeness, this part is necessary to understand my research on integrating the machine into the production process.

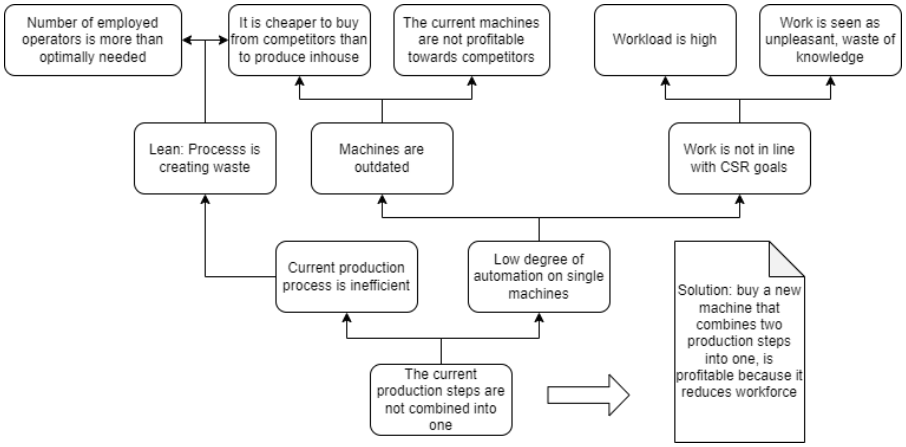


Figure 16: Purchasing motivation

Appendix II: Computation of results from the Chi-squared test

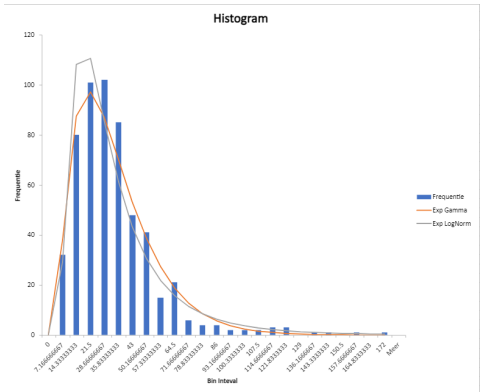
For 4x1.15mm the output by Wolfram Mathematica is: *FrechetDistribution* [5.66711, 74.3673, -54.3665], *GammaDistribution* [2.20892, 13.7142], *LogNormalDistribution* [3.16782, 0.739563]. The number of observations is $n = 555$. The Frechet distribution can not be plotted in excel, therefore the test is only performed for the Gamma and LogNormal distribution (figure 17b). From the Chi-Squared test (figure 17a), it follows, that the h_0 hypothesis must be rejected when alpha is 0.05 for both distributions. For a value of 0.01 for alpha, the h_0 hypothesis can be accepted for the LogNormal distribution.

$$X^2_{(0.01,24)} = 42.980 \geq X^2 = 42.255 \tag{66}$$

Based on the same assumption as for 3x1.15mm (4.4.1), the LogNormal distribution can represent product 4x1.15mm with parameters $[\mu = 3.16782, \sigma = 0.739563]$. With equations (54) - (56) for the LogNormal distribution, the mean of 4x1.15mm is 31.227, the variance is 1339.530, and the standard deviation is 36.600.

Bin Inteval	*Frequency*	Error Gamma	Error LogNorm
0	0	0	0
7.166666667	32	0.957242972	0.272770785
14.33333333	80	0.628937563	7.281095631
21.5	101	0.148460266	0.81584786
28.66666667	102	2.576351506	3.210774872
35.83333333	85	3.148810491	9.115208658
43	48	0.492598586	0.513608174
50.16666667	41	0.149376455	3.543046676
57.33333333	15	5.499515562	2.127154992
64.5	21	0.255896515	1.762821856
71.66666667	6	3.585176838	2.622802667
78.83333333	4	2.419335524	2.376670848
86	4	0.486943637	0.87023206
93.16666667	2	0.790817081	1.634639602
100.3333333	2	0.072157543	0.75826332
107.5	2	0.121906031	0.242646141
114.6666667	3	3.958830951	0.29022639
121.8333333	3	8.640115509	0.938903354
129	0	0.40947107	1.365324636
136.1666667	1	2.109921223	0.007027426
143.3333333	1	4.255765631	0.018798317
150.5	0	0.103426198	0.703668738
157.6666667	1	13.45584081	0.32179414
164.8333333	0	0.04071006	0.466363958
172	1	37.32389899	0.995393769
Meer	0		
	SUM	91.63150701	42.25508487
	Chi alpha 0.05	36.4150285	

(a) Chi-Squared test



(b) Observed and expected frequencies

Figure 17: Distributions for 4x1.15mm

For 5x1.15mm the output by Wolfram Mathematica is: $WeibullDistribution[0.93422, 10.7001, 0.689546]$, $LogNormalDistribution[2.0028, 1.01037]$, $GammaDistribution[1.22971, 9.53361]$. The number of observations is $n = 104$ and the first distribution given by Wolfram Mathematica is the Weibull distribution. Here, for a value of 0.05 for alpha, the h_0 hypothesis can be accepted (figure 18). This means that the Weibull distribution can be used to represent product 5x1.15mm with parameters $[\alpha = 0.93422, \beta = 10.7001, \mu = 0.689546]$. The mean and variance of this distribution are given by

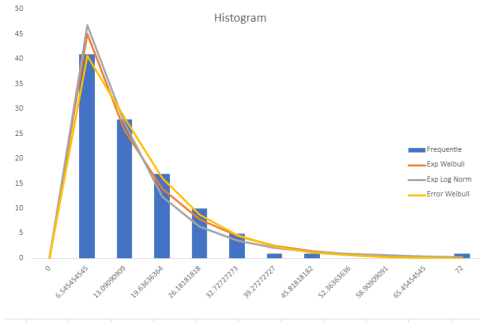
$$mean = \mu + \beta\Gamma(1 + 1/\alpha) \tag{67}$$

$$variance = \beta^2[\Gamma(1 + 2/\alpha) - \Gamma(1 + 1/\alpha)^2] \tag{68}$$

where the mean is 11.73, the variance is 53.112, and the standard deviation is 7.288 ((56)).

Bin Interval	*Frequentie*	Error Weibull	Error LogNorm	Errror Gamma
0	0	0	0	0
6.545454545	41	0.381592533	0.746847131	0.001216123
13.09090909	28	0.17943609	0.019456064	0.005241993
19.63636364	17	0.593781241	1.705087136	0.043700965
26.18181818	10	0.535943152	2.015548519	0.162085793
32.72727273	5	0.045663236	0.506556627	0.019124305
39.27272727	1	1.012707368	0.670770876	0.882244828
45.81818182	1	0.18882414	0.128183526	0.067998109
52.36363636	0	0.90611796	0.957392973	0.674792293
58.90909091	0	0.53661788	0.664066428	0.349612282
65.45454545	0	0.319391562	0.473687543	0.180546954
72	1	3.428461436	1.236531335	8.846098811
Meer	0			
	SUM	8.128536599	9.124128159	11.23266246
	Chi alpha 0.05	19.67513757		

(a) Chi-Squared test



(b) Observed and expected frequencies

Figure 18: Distributions for 5x1.15mm

For 3x0.4mm the output by Wolfram Mathematica is: $ExponentialDistribution[0.881691]$, $GammaDistribution[0.986955, 1.14918]$, $WeibullDistribution[0.999077, 1.13375]$. The number of observations is $n = 139$ and the first distribution given by Wolfram Mathematica is the Exponential distribution. Here, for a value of 0.05 for alpha, the h_0 hypothesis can be accepted (figure 19). This means that the Exponential distribution can be used to represent product 3x0.4mm with parameter $[\lambda = 0.881691]$. The mean and variance of this distribution are given by

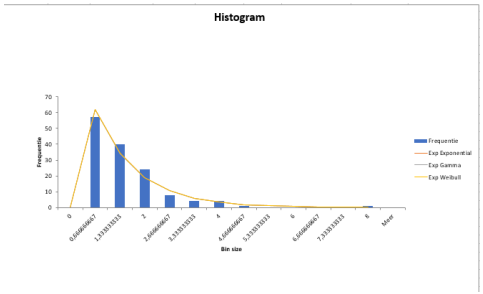
$$mean = 1/\lambda \tag{69}$$

$$variance = 1/\lambda^2 \tag{70}$$

where the mean is 1.134, the variance is 1.286, and the standard deviation is 1.134 ((56)).

Bin size	*Frequentie*	Error Exponential	Error Gamma	Error Weibull
0	0	0	0	0
0.666666667	57	0.369588626	0.415471459	0.375504179
1.333333333	40	0.93967014	1.029969831	0.949687291
2	24	1.276197523	1.349754311	1.286560339
2.666666667	8	0.634623178	0.619495385	0.631732962
3.333333333	4	0.603682027	0.60681703	0.603364732
4	4	0.163302651	0.153691029	0.16231788
4.666666667	1	0.366853918	0.381533744	0.368687158
5.333333333	0	1.009035461	1.026604712	1.011452832
6	0	0.560571144	0.573778614	0.562496713
6.666666667	0	0.311426129	0.320749221	0.312844076
7.333333333	0	0.173013247	0.179329119	0.174006453
8	1	8.500021703	8.072951195	8.428470597
Meer	0			
	SUM	14.90798575	14.73014565	14.86712521
	Chi alpha 0.05	21.02606982		

(a) Chi-Squared test



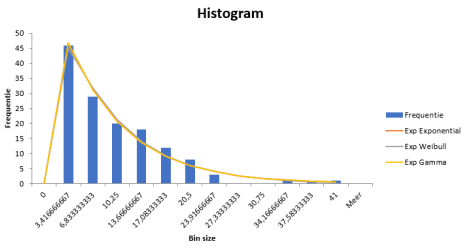
(b) Observed and expected frequencies

Figure 19: Distributions for 3x0.4mm

For 8x0.4mm the output by Wolfram Mathematica is: $ExponentialDistribution[0.120084]$, $WeibullDistribution[1.03609, 8.43979]$, $GammaDistribution[1.01026, 8.24293]$. The number of observations is $n = 139$ and the first distribution given by Wolfram Mathematica is the Exponential distribution. Here, for a value of 0.05 for alpha, the h_0 hypothesis can be accepted (figure 20). This means that the Exponential distribution can be used to represent product 8x0.4mm with parameter $[\lambda = 0.120]$. With equations (69), (70), and (56) for the Exponential distribution, the mean of 8x0.4mm is 8.328, the variance is 69.347, and the standard deviation is 8.328.

Bin size	*Frequentie*	Error Exponential	Error Weibull	Error Gamma
0	0	0	0	0
3.416667	46	0.012975364	0.01956163	0.005124743
6.833333	29	0.133569676	0.230133015	0.152138679
10.25	20	0.01697229	0.082340738	0.024429731
13.66667	18	1.377847596	1.043139218	1.326700712
17.08333	12	0.951177699	0.766555072	0.92717124
20.5	8	0.656246978	0.584053398	0.649728404
23.91667	3	0.245505572	0.244573642	0.243653864
27.33333	0	2.647000817	2.593417025	2.636956969
30.75	0	1.756178659	1.682028991	1.74441695
34.16667	1	0.023409636	0.007168322	0.020504103
37.58333	1	0.066638954	0.125770617	0.073559942
41	1	0.462664443	0.660951652	0.486316507
Meer	0			
	SUM	8.350187685	8.039693319	8.290701843
	Chi alpha 0,05	21.02606982		

(a) Chi-Squared test



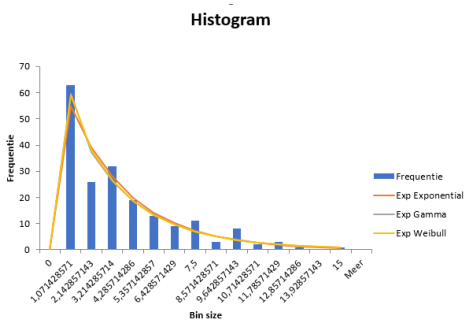
(b) Observed and expected frequencies

Figure 20: Distributions for 8x0.4mm

For 3x0.5mm the output by Wolfram Mathematica is: $ExponentialDistribution[0.31555]$, $GammaDistribution[0.890536, 3.55861]$, $WeibullDistribution[0.942782, 3.08925]$. The number of observations is $n = 191$ and the first distribution given by Wolfram Mathematica is the Exponential distribution. Here, for a value of 0.05 for alpha, the h_0 hypothesis can be accepted (figure 21). This means that the Exponential distribution can be used to represent product 3x0.5mm with parameter $[\lambda = 0.316]$. With equations (69), (70), and (56) for the Exponential distribution, the mean of 3x0.5mm is 3.169, the variance is 10.043, and the standard deviation is 3.169.

Bin size	*Frequentie*	Error Exponential	Error Gamma	Error Weibull
0	0	0	0	0
1.071428571	63	1.229626573	0.193739705	0.289708991
2.142857143	26	4.374400244	3.474830236	3.825649034
3.214285714	32	0.613683221	1.327788033	1.176337461
4.285714286	19	0.038199498	0.008132789	0.005100949
5.357142857	13	0.096734504	0.011598957	0.009760439
6.428571429	9	0.12096937	0.049777058	0.039856026
7.5	11	1.996681271	2.222666641	2.342170905
8.571428571	3	0.890510562	0.88488444	0.838515371
9.642857143	8	5.127495942	4.834625427	5.034366516
10.71428571	2	0.144066395	0.198763383	0.180920111
11.78571429	3	0.692551175	0.495302365	0.527082208
12.85714286	1	0.081522007	0.149246704	0.141762104
13.92857143	0	0.947877742	1.076514762	1.070506517
15	1	0.155335597	0.055858439	0.056102134
Meer	0			
	SUM	15.40644076	13.85135574	14.41123012
	Chi alpha 0,05	23.6847913		

(a) Chi-Squared test



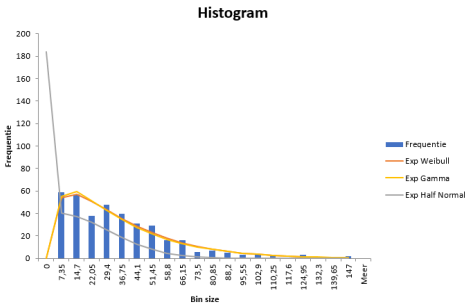
(b) Observed and expected frequencies

Figure 21: Distributions for 3x0.5mm

For 8x0.5mm the output by Wolfram Mathematica is: $WeibullDistribution[1.18395, 34.9943]$, $HalfNormalDistribution[0.0299915]$, $GammaDistribution[1.2684, 26.0889]$. The number of observations is $n = 368$ and the first distribution given by Wolfram Mathematica is the Weibull distribution. Here, for a value of 0.05 for alpha, the h_0 hypothesis can be accepted (figure 22). This means that the Weibull distribution can be used to represent product 8x0.5mm with parameter $[\alpha = 1.18395, \beta = 34.9943]$. With equations (67), (68), and (56) for the Weibull distribution, the mean of 8x0.5mm is 33.034, the variance is 369.756, and the standard deviation is 19.229.

Bin size	*Frequentie*	Error Weibull	Error HalfNormal	Error Gamma
0	0	0	0	0
7,35	59	0,530275126	8,349892486	0,259450256
14,7	57	0,000199652	10,06235819	0,100412016
22,05	38	3,284195153	1,064569953	3,60928108
29,4	48	0,537588635	19,96427595	0,654083264
36,75	40	0,550508098	24,45715191	0,877189914
44,1	31	0,178252454	26,70303151	0,450959273
51,45	29	1,642854778	56,13394661	2,464311181
58,8	16	0,219144456	28,35807141	0,059710569
66,15	16	0,284782496	74,44738664	0,560679079
73,5	6	2,139723612	18,69747735	1,807044847
80,85	7	0,19819435	73,98744731	0,124365193
88,2	5	0,268875455	95,31729783	0,226454555
95,55	3	0,654526902	90,43117513	0,657405657
102,9	4	0,048059833	463,5542484	0,028756454
110,25	3	0,037556143	783,4182686	0,010656928
117,6	1	0,49838696	280,2247414	0,631678491
124,95	3	1,557068108	8829,243784	1,068615755
132,3	0	1,093725378	0,000270882	1,276845702
139,65	0	0,80439469	6,66313E-05	0,977849945
147	2	3,375888524	263838,8469	2,093875285
Meer	0			
	SUM	**9,834594266**	**437,5429072**	**11,19394123**
	Chi alpha 0,05	**31,41043284**		

(a) Chi-Squared test



(b) Observed and expected frequencies

Figure 22: Distributions for 8x0.5mm

For 3x0.6mm the number of observations is $n = 6$, and for 8x0.6mm the number of observations is $n = 21$. These are below the boundary of 100 data points to be able to confidently plot a distribution over the observed frequency (Anderson & Gerbing, 1984). But as the days of production (number of observations) are minimal compared to the other products, there is no need to determine parameters like the safety stock, thus no distribution is needed for those two products. Demand of the two products will be handled as make to order (MTO), this is a term used by TKF to specify that the quantity of production must never exceed the demand for a production day, no inventory is allowed for this product.

Appendix III: Demand

Date	1.15x3	1.15x4	1.15x5	0.4x3	0.4x8	0.5x3	0.5x8
10/18/2023	8	52	0	1	5	0	41
10/19/2023	7	29	0	1	4	0	17
10/20/2023	4	14	0	0	0	0	0
10/23/2023	13	88	0	0	0	0	0
10/24/2023	5	35	0	0	0	0	0
10/25/2023	9	47	0	1	1	0	21
10/26/2023	10	25	0	1	3	0	9
10/27/2023	1	15	14	1	4	0	0
10/29/2023	3	41	24	0	0	0	0
10/30/2023	2	21	5	1	1	0	6
10/31/2023	12	21	0	0	0	0	30
11/1/2023	6	41	0	0	0	2	43
11/2/2023	6	37	0	0	0	5	22
11/3/2023	12	32	11	0	0	2	18
11/5/2023	17	77	3	0	0	0	0
11/6/2023	5	18	2	0	0	4	26
11/7/2023	8	21	2	0	0	1	9
11/8/2023	15	24	0	0	0	2	34
11/9/2023	4	22	3	0	0	0	33
11/10/2023	7	48	0	0	0	0	25
11/11/2023	1	7	0	0	0	0	0
11/12/2023	6	46	4	0	0	0	0
11/13/2023	1	7	0	0	0	0	16
11/14/2023	5	33	0	1	9	0	8
11/15/2023	6	37	0	1	1	0	0
11/16/2023	6	40	3	0	0	0	12
11/17/2023	2	11	0	0	0	0	0
11/20/2023	19	134	11	0	0	0	0
11/21/2023	2	17	3	0	0	0	24
11/22/2023	3	18	0	0	0	0	20

Table 19: Demand over 30 days

Date	1.15x3	1.15x4	1.15x5	0.4x3	0.4x8	0.5x3	0.5x8
10/18/2023	8	52	0	1	5	0	21
10/19/2023	7	29	0	1	4	0	27
10/20/2023	4	58	0	0	0	0	10
10/23/2023	13	44	0	0	0	0	0
10/24/2023	5	35	0	0	0	0	0
10/25/2023	9	47	0	1	1	0	21
10/26/2023	10	25	0	1	3	0	9
10/27/2023	1	15	14	1	4	0	0
10/29/2023	3	41	24	0	0	0	0
10/30/2023	2	21	5	1	1	0	6
10/31/2023	12	21	0	0	0	0	30
11/1/2023	6	41	0	0	0	2	43
11/2/2023	6	37	0	0	0	5	22
11/3/2023	12	32	11	0	0	2	18
11/5/2023	17	77	3	0	0	0	0
11/6/2023	5	18	2	0	0	4	26
11/7/2023	8	21	2	0	0	1	9
11/8/2023	15	24	0	0	0	2	34
11/9/2023	4	22	3	0	0	0	33
11/10/2023	7	48	0	0	0	0	25
11/11/2023	1	7	0	0	0	0	0
11/12/2023	6	46	4	0	0	0	0
11/13/2023	1	7	0	0	0	0	16
11/14/2023	5	33	0	1	9	0	8
11/15/2023	6	37	0	1	1	0	0
11/16/2023	6	40	3	0	0	0	12
11/17/2023	2	78	0	0	0	0	0
11/20/2023	19	67	11	0	0	0	0
11/21/2023	2	17	3	0	0	0	24
11/22/2023	3	18	0	0	0	0	20

Table 20: Adjusted demand for single decks over 30 days

Appendix IV: Python code for step 2 and 4 of the solving procedure

The following two listings are the code used to solve the TSP matrices. The first listing is used for step 2 and the second for step 4 of the solving procedure, including the setup to the first item of the next day.

```
# Determine the Initial Sequence
import pandas as pd
import numpy as np
from pandas import DataFrame
from python_tsp.exact import solve_tsp_dynamic_programming

# df_InitialQuantitiesDay1 = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\InitialQuantities.xlsx", usecols="B")
# df_InitialQuantities = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\InitialQuantities.xlsx", usecols="C:AE")

df_InitialQuantitiesDay1 = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\SP3 quantities.xlsx", usecols="B")
df_InitialQuantities = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\SP3 quantities.xlsx", usecols="C:AE")
df_ChangeoverSingleDeck = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\ChangeoverMatrix.xlsx",
                                        index_col=0, usecols="A:H", nrows=7)

df_InitialQuantitiesDay1[df_InitialQuantitiesDay1 > 0] = 1
df_InitialQuantities[df_InitialQuantities > 0] = 1

# Initialize an empy list to store sequences
data_sequences = []

# zero arrays for later use
rowJ = np.arange(8) * 0
rowX = np.arange(7) * 0

# sequence dependent matrix
changeover_matrix = np.array(df_ChangeoverSingleDeck)

# setup matrix where row 0 is used for the index of product k
Setup_matrix = np.array(df_ChangeoverSingleDeck)
Setup_matrix = np.append(rowX.reshape(1, 7), Setup_matrix, axis=0)

print("The changeover times are:")
print(changeover_matrix)
print("The setup times are:")
print(Setup_matrix)

print("This is the TSP per production day")

# determine the sequence for day 1
for coll in df_InitialQuantitiesDay1:
    Day1 = np.asarray(df_InitialQuantitiesDay1[coll])
    QuantitiesDay1_Matrix = np.multiply.outer(Day1, Day1)
    TSP_Day1 = np.multiply(QuantitiesDay1_Matrix, changeover_matrix)

    # delete all zero rows and columns
    TSP_Day1 = TSP_Day1[~np.all(TSP_Day1 == 0, axis=1)]
    TSP_Day1 = TSP_Day1[:, ~np.all(TSP_Day1 == 0, axis=0)]
    # no turning back to starting state
    TSP_Day1[:, 0] = 0

    # Keep track of item indices (index +1) to sequence
    TrackVar1 = np.arange(1, len(Day1) + 1) * Day1
    TrackVar1 = TrackVar1[TrackVar1 != 0]

```

```
    # solve TSP day 1
    permutation, distance = solve_tsp_dynamic_programming(TSP_Day1)
    SequenceDay1 = solve_tsp_dynamic_programming(TSP_Day1)[-2]
    SequenceDay1 = np.array([i + 1 for i in SequenceDay1])
    LastItem = SequenceDay1[-1]
    FirstDistance = TSP_Day1[0, 0]
    Distance = solve_tsp_dynamic_programming(TSP_Day1)[-1]
    DistanceWithoutFirstItem = Distance - FirstDistance

    # ensure correct representation of sequence
    Indices1 = np.append(TrackVar1.reshape(1, len(TrackVar1)), SequenceDay1.
    reshape(1, len(SequenceDay1)), axis=0)
    for i in range(len(Indices1[0])):
        Indices1[1, i] = Indices1[0, Indices1[1, i] - 1]
    SequenceDay1 = Indices1[1, :]
    LastItem = SequenceDay1[-1]
    FirstItem = SequenceDay1[0]
    sequences_string = "[" + ", ".join(map(str, SequenceDay1)) + "]"
    output_string = "(" + sequences_string + ", " + str(Distance) + ")"

    print(coll, "TSP matrix")
    print(TSP_Day1)
    print(coll, "Items to sequence:", TrackVar1)
    print(coll, "Solution:", output_string)
    print(coll, "Last item:", LastItem)

    # store data
    data_sequences.append({'Days': coll, 'Sequence': sequences_string, "
    Distance": Distance,
                           "Last Item": LastItem, "First Item": FirstItem,
                           "First Distance": FirstDistance, "Distance without
     First Item": DistanceWithoutFirstItem})

# determine sequences for days 2-30
for col in df_InitialQuantities:
    # create a matrix from the initial quantities
    vectorA = np.array(df_InitialQuantities[col])
    InitialQuantities_Matrix = np.multiply.outer(vectorA, vectorA)

    # compute the matrix product of the quantities matrix and the changeover
    matrix
    TSP_per_Day = np.multiply(InitialQuantities_Matrix, changeover_matrix)

    # enter last Item of t-1 sequence into row 0 as setup state
    SequenceLastItem = np.multiply(Setup_matrix[LastItem, :], vectorA)
    TSP_per_Day_K = np.append(SequenceLastItem.reshape(1, 7), TSP_per_Day,
    axis=0)
    # add a 0 values column at column index 0 for an open TSP, it is not
    required to go back to the origin
    TSP_per_Day_KJ = np.append(rowJ.reshape(8, 1), TSP_per_Day_K, axis=1)

    # Find columns where all values are zero and delete, starting from the
    second column
    zero_columns_starting_from_second = np.all(TSP_per_Day_KJ[:, 1:] == 0,
    axis=0)
    TSP_per_Day_KJ = TSP_per_Day_KJ[:, ~np.concatenate(([False],
    zero_columns_starting_from_second))]

    # Find columns where all values are zero and delete, starting from the
    second column
    if TSP_per_Day_KJ.shape[1] > 2:
        TSP_per_Day_KJ = TSP_per_Day_KJ[~np.all(TSP_per_Day_KJ == 0, axis=1)]
    else:
        vector0 = np.arange(2) * 0
        TSP_per_Day_KJ = np.append(TSP_per_Day_KJ[:2, :], vector0.reshape(2,
    1), axis=1)

```

```
    print(col, "TSP matrix")
    print(TSP_per_Day_KJ)

    # Keep track of item indices (index +1) to sequence
    TrackVar = np.arange(len(vectorA) + 1) * np.insert(vectorA, 0, 1)
    # Identify non-zero and delete zero elements starting from the second
    element
    non_zero_indices = np.nonzero(TrackVar[1:])[0] + 1
    TrackVar = np.concatenate(([TrackVar[0]], TrackVar[non_zero_indices]))
    print(col, "Items to sequence:", TrackVar)

    # solve TSP
    permutation, distance = solve_tsp_dynamic_programming(TSP_per_Day_KJ)
    Sequence = solve_tsp_dynamic_programming(TSP_per_Day_KJ)[-2]
    Sequence = np.array(Sequence)
    LastItem = Sequence[-1]
    Distance = solve_tsp_dynamic_programming(TSP_per_Day_KJ)[-1]
    FirstDistance = TSP_per_Day_KJ[0, Sequence[1]]
    DistanceWithoutFirstItem = Distance - FirstDistance

    # When LastItem equals 0, there is no transition into another state.
    # So there is no changeover happening as the machine is already setup
    # from the previous state for the one product to be made.
    # Take the index from the one product scheduled as LastItem
    if LastItem == 0:
        # we need an extra 0 to account for the setup state as new index 0
        IndexDummy = np.append([0], vectorA)
        LastItem = np.where(IndexDummy == 1)[0].tolist()[0]
        Sequence = TrackVar
    else:
        # Sequence solution is based on the indices of manipulated TSP matrix
     after row, column deletion
        # Determine original indices of matrix before deletion, to get
    correct sequences
        Indices = np.append(TrackVar.reshape(1, len(TrackVar)), Sequence.
    reshape(1, len(Sequence)), axis=0)
        for i in range(len(Indices[1])):
            Indices[1, i] = Indices[0, Indices[1, i]]
        Sequence = Indices[1, :]
        LastItem = Sequence[-1]

    FirstItem = Sequence[1]
    sequences_string = "[" + ", ".join(map(str, Sequence)) + "]"
    output_string = "(" + sequences_string + ", " + str(Distance) + ")"

    print(col, "Solution:", output_string)
    print(col, "Last item:", LastItem)

    data_sequences.append({'Days': col, 'Sequence': sequences_string, "
    Distance": Distance,
                           "Last Item": LastItem,  "First Item": FirstItem,
                           "First Distance": FirstDistance, "Distance without
     First Item": DistanceWithoutFirstItem})

# save initial quantities in excel
df = pd.DataFrame(data_sequences)
df = df.set_index(df.columns[0], drop=True)
df_transposed = df.T
df_transposed.to_excel(r"C:\Users\Micha\PycharmProjects\Thesis_TSP\SP3
    sequences.xlsx")
```

This is the second listing of the code:

```
import pandas as pd
import numpy as np
from pandas import DataFrame
from python_tsp.exact import solve_tsp_dynamic_programming

# df_InitialQuantities = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\InitialQuantities.xlsx", sheet_name="Sheet1", usecols="B:AE")
df_InitialQuantities = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\SP3 quantities.xlsx",
                                     sheet_name="Sheet1", usecols="B:AE")
df_ChangeoverSingleDeck = pd.read_excel(r"C:\Users\Micha\PycharmProjects\
    Thesis_TSP\ChangeoverMatrix.xlsx", index_col=0,
                                        usecols="A:H", nrows=7)
# df_FirstItem = pd.read_excel(r"C:\Users\Micha\PycharmProjects\Thesis_TSP\
    Initial Sequences Single Decks.xlsx", index_col=0, skiprows=[1, 2, 3, 5,
    6])
df_FirstItem = pd.read_excel(r"C:\Users\Micha\PycharmProjects\Thesis_TSP\SP3
    sequences.xlsx",
                             index_col=0, skiprows=[1, 2, 3, 5, 6])
df_InitialQuantities[df_InitialQuantities > 0] = 1
print(df_FirstItem)
# zero arrays for later use
rowJ = np.arange(8) * 0
rowX = np.arange(7) * 0

# Initialize an empy list to store sequences
data_sequences = []
LastItem = 0

# sequence dependent matrix
changeover_matrix = np.array(df_ChangeoverSingleDeck)

# setup matrix where row 0 is used for the index of product k
Setup_matrix = np.array(df_ChangeoverSingleDeck)
Setup_matrix = np.append(rowX.reshape(1, 7), Setup_matrix, axis=0)

print("The changeover times are:")
print(changeover_matrix)
print("The setup times are:")
print(Setup_matrix)

print("This is the TSP per production day")

# determine sequences for days 1-30
columns = df_InitialQuantities.columns
for i, col in enumerate(columns):
    # create a matrix from the initial quantities
    vectorA = np.array(df_InitialQuantities[col])
    InitialQuantities_Matrix = np.multiply.outer(vectorA, vectorA)

    # compute the matrix product of the quantities matrix and the changeover
    matrix
    TSP_per_Day = np.multiply(InitialQuantities_Matrix, changeover_matrix)

    if col == "day-01":
        # add a row 0, with only zeros
        TSP_per_Day_K = np.append(rowX.reshape(1, 7), TSP_per_Day, axis=0)
        print(col, "we are inside the if statement")
    else:
        # enter last Item of t-1 sequence into row 0 as setup state
        SequenceLastItem = np.multiply(Setup_matrix[LastItem, :], vectorA)
        TSP_per_Day_K = np.append(SequenceLastItem.reshape(1, 7), TSP_per_Day
    , axis=0)

    # add transition times to First Item t+1 in col 0
    if i < len(columns) - 1:
```

```
        next_col = columns[i + 1]
        NextFirstItem = np.array(df_FirstItem[next_col])
        VectorNextFirstItem = np.array(Setup_matrix[:, NextFirstItem - 1])
        TSP_per_Day_KJ = np.append(VectorNextFirstItem.reshape(8, 1),
    TSP_per_Day_K, axis=1)
        print(col, "Next First Item", NextFirstItem)
    else:
        # if it is the last day there is no setup to the next day, col 0 = 0
        TSP_per_Day_KJ = np.append(rowJ.reshape(8, 1), TSP_per_Day_K, axis=1)

    # Find columns where all values are zero and delete, starting from the
    second column
    zero_columns_starting_from_second = np.all(TSP_per_Day_KJ[:, 1:] == 0,
    axis=0)
    TSP_per_Day_KJ = TSP_per_Day_KJ[:, ~np.concatenate(([False],
    zero_columns_starting_from_second))]
    # Find rows where all values are zero and delete, excluding row 0 and col
     0
    if TSP_per_Day_KJ.shape[1] > 2:
        zero_rows_starting_from_second = np.any(TSP_per_Day_KJ[1:, 1:] != 0,
    axis=1)
        TSP_per_Day_KJ = np.vstack([TSP_per_Day_KJ[0, :], TSP_per_Day_KJ[1:][
    zero_rows_starting_from_second]])
    else:
        vector0 = np.arange(2) * 0
        TSP_per_Day_KJ = np.append(TSP_per_Day_KJ[:2, :], vector0.reshape(2,
    1), axis=1)
    print(col, "TSP matrix")
    print(TSP_per_Day_KJ)

    # Keep track of item indices (index +1) to sequence
    TrackVar = np.arange(len(vectorA) + 1) * np.insert(vectorA, 0, 1)
    # Identify non-zero and delete zero elements starting from the second
    element
    non_zero_indices = np.nonzero(TrackVar[1:])[0] + 1
    TrackVar = np.concatenate(([TrackVar[0]], TrackVar[non_zero_indices]))
    print(col, "Items to sequence:", TrackVar)

    # solve TSP
    permutation, distance = solve_tsp_dynamic_programming(TSP_per_Day_KJ)
    Sequence = solve_tsp_dynamic_programming(TSP_per_Day_KJ)[-2]
    Sequence = np.array(Sequence)
    LastItem = Sequence[-1]
    Distance = solve_tsp_dynamic_programming(TSP_per_Day_KJ)[-1]
    LastDistance = TSP_per_Day_KJ[LastItem, 0]
    FirstDistance = TSP_per_Day_KJ[0, Sequence[1]]
    DistanceWithoutSetups = Distance - LastDistance - FirstDistance
    # When LastItem equals 0, there is no transition into another state.
    # So there is no changeover happening as the machine is already setup
    # from the previous state for the one product to be made.
    # Take the index from the one product scheduled as LastItem
    if LastItem == 0:
        # we need an extra 0 to account for the setup state as new index 0
        IndexDummy = np.append([0], vectorA)
        LastItem = np.where(IndexDummy == 1)[0].tolist()[0]
        Sequence = TrackVar
    else:
        # Sequence solution is based on the indices of manipulated TSP matrix
     after row, column deletion
        # Determine original indices of matrix before deletion, to get
    correct sequences
        Indices = np.append(TrackVar.reshape(1, len(TrackVar)), Sequence.
    reshape(1, len(Sequence)), axis=0)
        for x in range(len(Indices[1])):
            Indices[1, x] = Indices[0, Indices[1, x]]
        Sequence = Indices[1, :]
        LastItem = Sequence[-1]
```

```
114
115     FirstItem = Sequence[1]
116     sequences_string = "[" + ", ".join(map(str, Sequence)) + "]"
117     output_string = "(" + sequences_string + ", " + str(Distance) + ")"
118
119     print(col, "Solution:", output_string)
120     print(col, "Last item:", LastItem)
121
122     data_sequences.append({'Days': col, 'Sequence': sequences_string, "
        Distance": Distance,
123                            "Last Item": LastItem, "First Item": FirstItem, "
        First Distance": FirstDistance,
124                            "LastDistance": LastDistance, "Distance without
        setups": DistanceWithoutSetups})
125
126 # save initial quantities in excel
127 df = pd.DataFrame(data_sequences)
128 df = df.set_index(df.columns[0], drop=True)
129 df.to_excel(r"C:\Users\Micha\PycharmProjects\Thesis_TSP\Sequences Single
        Decks.xlsx")
```

References

Almada-Lobo, B., Klabjan, D., Carravilla, M. A., & Oliveira, J. F. (2007). Single machine multi-product capacitated lot sizing with sequence-dependent setups. *International Journal of Production Research*, *45*(20), 4873–4894. doi: 10.1080/00207540601094465

Anderson, J. C., & Gerbing, D. W. (1984). The effect of sampling error on convergence, improper solutions, and goodness-of-fit indices for maximum likelihood confirmatory factor analysis. *Psychometrika*, *49*, 155–173.

Bahl, H. C., Ritzman, L. P., & Gupta, J. N. D. (1986). Determining lot sizes and requirements a review.pdf. *Operations Research*, *35*(3), 329–345. Retrieved from http://www.jstor.org/stable/170535

Chopra, S. (2019). *Supply chain management : strategy, planning and operation LK - https://ut.on.worldcat.org/oclc/1076802995* (Seventh ed ed.). Harlow, United Kingdom: Pearson Education Limited. Retrieved from https://www.vlebooks.com/vleweb/product/openreader?id=Bradford&isbn=9781292257914&uid=%5Eu

Cooper, D., & Schindler, P. (2004). *Business research methods.* Tata McGra-Hill. Retrieved from https://books.google.nl/books?id=0djAtwAACAAJ

Dixon, P. S., & Silver, E. A. (1981). A heuristic solution procedure for the multi-item, single-level, limited capacity, lot-sizing problem. *Journal of Operations Management*, *2*(1), 23–39. Retrieved from https://www.sciencedirect.com/science/article/pii/0272696381900334 doi: https://doi.org/10.1016/0272-6963(81)90033-4

Fleischmann, B., & Meyr, H. (1997). The general lotsizing and scheduling problem. *OR Spectrum*, *19*(1), 11–21. doi: 10.1007/BF01539800

Haase, K. (1996). Capacitated lot-sizing with sequence dependent setup costs. *Operations-Research-Spektrum*, *18*(1), 51–59. Retrieved from https://doi.org/10.1007/BF01539882 doi: 10.1007/BF01539882

Haase, K., & Kimms, A. (2000). Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics*, *66*(2), 159–169. doi: 10.1016/S0925-5273(99)00119-X

Heerkens, H., & van Winden, A. (2017). *Solving managerial problems systematically.* Noordhoff Uitgevers. (Translated into English by Jan-Willem Tjooitink)

Holloway, L. E., & Hall, A. (1997). Principles of Lean Manufacturing. *Industry and Higher Education*, *11*(4), 241–245. Retrieved from https://doi.org/10.1177/095042229701100410 doi: 10.1177/095042229701100410

Kiran, D. (2019). Chapter 26 - just in time and kanban. In D. Kiran (Ed.), *Production planning and control* (p. 369-379). Butterworth-Heinemann. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780128183649000263 doi: https://doi.org/10.1016/B978-0-12-818364-9.00026-3

Laguna, M. (1999). A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. *IIE Transactions (Institute of Industrial Engineers)*, *31*(2), 125–134. doi: 10.1080/07408179908969811

Maes, J., & Wassenhove, L. V. (1988). Multi-Item Single-Level Capacitated Dynamic Lot-Sizing Heuristics : A General Review. *The Journal of the Operational Research Society*, *39*(11), 991–1004.

Meyr, H. (2000). Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. *European Journal of Operational Research*, *120*(2), 311–326. doi: 10.1016/S0377-2217(99)00159-9

Moore, D. S. (2009). *Introduction to the practice of statistics.* WH Freeman and company.

Quadt, D., & Kuhn, H. (2008). Capacitated lot-sizing with extensions : a review. *4OR*, *6*, 61–83. doi: 10.1007/s10288-007-0057-1

Rockafellar, R. (2001). *Optimization under uncertainty lecture notes [pdf document].* University of Washington. Retrieved from https://sites.math.washington.edu/~rtr/uncertainty.pdf

Selen, W. J., & Heuts, R. M. (1990). Operational production planning in a chemical manufacturing environment. *European Journal of Operational Research*, *45*(1), 38–46. doi: 10.1016/0377-2217(90)90154-4

Winston, W. L., & Goldberg, J. B. (2004). *Operations research : applications and algorithms LK - https://ut.on.worldcat.org/oclc/52815313* (4th ed. ed.). Belmont, CA SE - xvi, 1418 pages : illustrations ; 26 cm + 1 CD-ROM (4 3/4 in.): Thomson/Brooks/Cole.