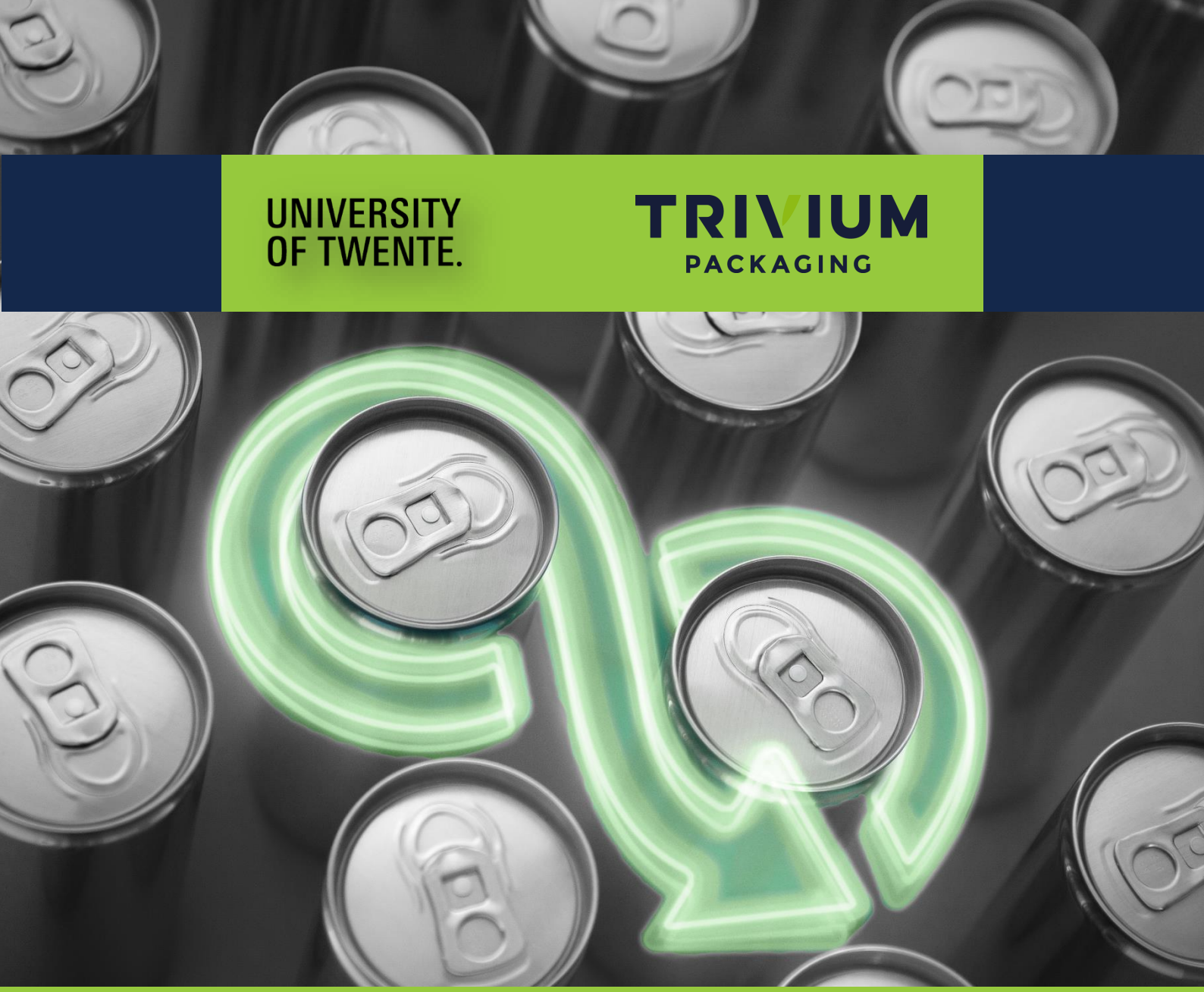


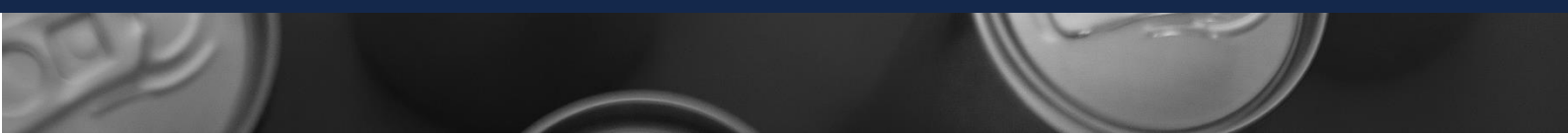
UNIVERSITY OF TWENTE.

TRIVIUM PACKAGING

Production planning of the 3PC department at Trivium Packaging

Author:
A.H. Schrader, University of Twente
Master: Industrial Engineering & Management
Track: Production & Logistics Management

Supervisors:
Dr. Ir. J.M.J. Schutten, University of Twente
Dr. P.C. Schuur, University of Twente
Ir. J. Kars, Trivium Packaging B.V.



Author
A. (Anne) Schrader

University
University of Twente

Master program
Production & Logistics Management

Graduation date
20th March 2020

Graduation committee
Dr. Ir. J.M.J. Schutten
University of Twente

Dr. P.C. Schuur
University of Twente

Ir. J. Kars
Trivium Packaging B.V.

Trivium Packaging B.V.
Zweedsestraat 7
7418 BG Deventer
Netherlands

University of Twente
Drienerlolaan 5
7522 NB Enschede
The Netherlands

MANAGEMENT SUMMARY

This thesis is about improving the planning and scheduling process at Trivium Packaging Deventer.

Trivium Packaging is a global manufacturer of metal packaging with around 60 production facilities around the world. Trivium Packaging Deventer is one of four production plants in The Netherlands and focusses on the food and pet-food industry. During this research, we focus on the planning and scheduling of the three-piece can making department, which consists of 6 production lines and makes over 50 different types of cans.

During planning and scheduling, a batch size and time slot of production is determined. Between different cans the machines must be cleaned and set-up for new production, which requires time and personnel. Producing in large batches or sequencing cans efficiently can reduce the overall set-up time and reduce time and personnel lost on set-up. Additionally, the internal storage capacity is limited at Trivium and high costs are associated with external storage. To keep costs low, usage of external storage must be kept to a minimum.

The need to reduce time spent on set-ups, combined with the limited internal storage capacity and high costs for external storage, makes the creation of a good schedule difficult.

The research question of this thesis is as follows:

> ***What can Trivium do to structurally improve their planning process to minimize the production- and inventory costs, while meeting all demand?***

To answer this research question, we divide our research in several phases. During the first phase, we analyze the current situation of the scheduling and production process. Next, we evaluate available literature related to our research which we use as input for our solution alternatives. We made a literature review for lot sizing and scheduling problems and identified several methods that can be used to solve the lot sizing and scheduling problem.

To improve the current planning process, we design a proof-of-concept method that automates the planning and scheduling process. The method determines the batch size and production week and sequences the production within the week. We identify and evaluate two alternative methods for the planning process:

- **Adaptive search:** A constructive heuristic that generates a schedule by picking a new production order to add to the schedule each iteration, until all production is assigned.
- **Simulated annealing**: A meta-heuristic that requires an initial schedule and optimizes it by looking at multiple neighboring schedules and accepting a neighbor schedule based on a changing probability.

We evaluate these alternatives by creating schedules for each of them with several datasets. Additionally, we create an MIP model and use it to benchmark our heuristics. We find that the best results are found using simulated annealing with the classical neighborhood strategies of moving and swapping production. Compared to the MIP model solved using CPLEX, the simulated annealing - classic strategy creates schedules that cost approximately 25% less than those created by CPLEX. For simulated annealing – optimal neighbor strategy the found objective value is an 80% increase on average. Adaptive search performs the worst, with an increase of 190% in the objective value compared to CPLEX.

Overall, we find that lower costs are associated with both a lower average inventory value and a lower number of changeovers. The greatest cost reduction is found by reducing the number of changeovers, but while we would expect this to be at the expense of higher inventory levels, this is not the case. Production is sequenced more efficiently, resulting in less changeovers, and production is distributed more equally over all weeks.

We compare the best alternative simulated annealing -classic strategy with the schedules created by the planner. Overall, we find that it should be possible to decrease the inventory levels below the current average. The schedules created by the algorithm result in lower average inventory values, on average approximately 34%, but this changes strongly depending on the dataset used. Nonetheless, a decrease is noted in all datasets. The algorithm also balances the inventory levels between weeks more evenly.

Next to the inventory levels, the algorithm also manages to sequence the products more efficiently, reducing the average number of changeovers for each instance. In a timespan of 12 weeks, this leads to a reduction of approximately 8 changeovers. Another improvement is the distribution of production over the week, which is likely connected to the lower average inventory levels reported by the algorithm.

We find that generally, improvements are possible to reduce the costs that result from scheduling decisions. We recommend a strict evaluation of the inventory levels to determine what stock is required and where could stock be reduced. In practice, this may be challenging due to unreliability of the forecast and no agreements on safety stock levels that should be kept. In future research, we also recommend including the forecast reliability, which could help in making accurate decisions about the required inventory levels. Overall, we find that cost reductions are possible but to facilitate this, additional steps must be taken to stabilize the forecast and production levels.

CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Meaning	Introduced on
3PC department	Three-piece can department	1
DA department	Ends department	1
DWI department	Two-piece department	1
SP department	Specials department	1
BKL	Coil cutting lines	1
LAK	Lacquering lines	1
OT	Open top	6
EO	Easy open	6
OEE	Overall equipment effectiveness	9
HFI pallets	Held-for-inspection pallets	11
WACC	Weighted average cost of capital	18
DIO	Days inventory outstanding	21
LSSP	Lot sizing and scheduling problem	25
CLSP	Capacitated lot sizing and scheduling problem	27
DLSP	Discrete lot sizing and scheduling problem	27
AS	Adaptive search	29
SA	Simulated annealing	29
TS	Tabu search	30
VNS	Variable neighborhood search	30
GA	Genetic algorithm	30
SA-CS	SA – classic strategy	44
SA-ONS	SA - Optimal Neighborhood Structure	44

1 INTRODUCTION

This project is completed as part of a master graduation project for the master Industrial Engineering and Management at the University of Twente. This research aims to improve the planning and scheduling process at Trivium Deventer. Trivium Deventer is part of Trivium Packaging B.V. and is specialized in the production of metal packaging. On behest of Trivium Deventer, we research the planning and scheduling process, as currently there is a lack of knowledge on the trade-off between production and inventory costs.

This chapter gives a general description of Trivium Packaging B.V. and the Deventer production plant in Section 1.1. We introduce the production process in Section 1.2. Section 1.3 contains a short problem description. Finally, Section 1.4 identifies the research questions.

1.1 TRIVIUM PACKAGING

Trivium Packaging is a global leader in the metal packaging industry and was founded at the end of 2019 by the merging of Exal Corporation and the Food & Specialty section of Ardagh Group. With a yearly revenue of 2.7 billion US dollars and nearly 8,000 employees, Trivium Packaging currently operates over 60 facilities around the world (Ardagh Group, 2019). The Deventer plant is specialized in the production of metal packaging. Initially founded as Thomassen & Drijver in 1919, they became part of Trivium Packaging in 2019.

Figure 1: 3PC can with a cylindrical bottom, attached bottom end and easy-open end (yellow part)

Figure 2: 2PC can with cylindrical body with integrated bottom end. The top is an easy-open end (yellow)

Trivium Deventer mainly produces steel packaging for the food and pet food industry. The customers include Saturn Petcare, Royal Friesland Campina, H.J. Heinz Company, Zwanenberg Food Group and several others. There are four different production departments in Deventer, each produce a different type of product. Due to several expansions over the years, the departments are a mix of old and new machinery. The three piece department (3PC) produces three-piece cans (see Figure 1), the ends department (DA) manufactures easy-open ends (yellow part in Figure 1), the two-piece department (DWI) manufactures two-piece cans (see Figure 2) and the specials department (SP) produces custom products. The custom products are split into cigar tins, syrup cans and special cans. To support these four departments, there are two coil-cutting lines (BKL) and four lacquering lines (LAK). The finished products are stored in the end product warehouse located at the Deventer facility or externally at Van Opijnen. For the biggest customer, Saturn, there is safety stock located in a warehouse close to their production location. Material warehouses are located all throughout the facilities, close to or inside

the departments. Figure 3 shows the location of the departments and the finished products warehouse. Material warehouses are not indicated but are in general found in the production area of next to the production area.

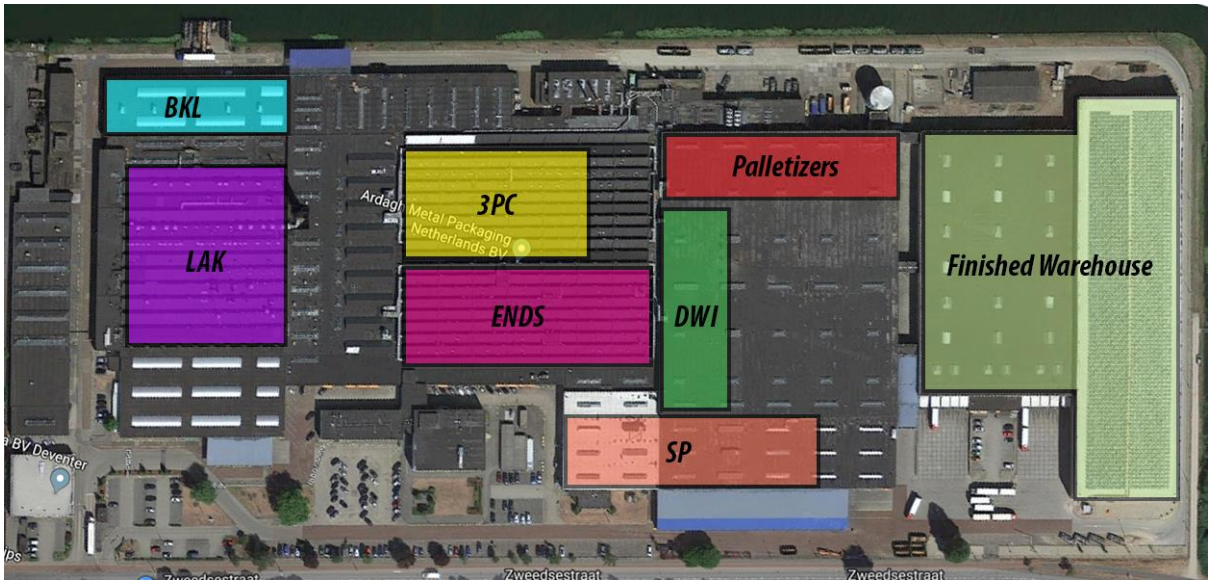


Figure 3: Trivium Deventer production location

1.2 Production process at Trivium Deventer

This study focusses on the production planning and scheduling of the 3PC department. The 3PC department operates five days a week, 24 hours per day. Figure 4 shows a simplified graphical representation of the production process, which we explain further in this section.

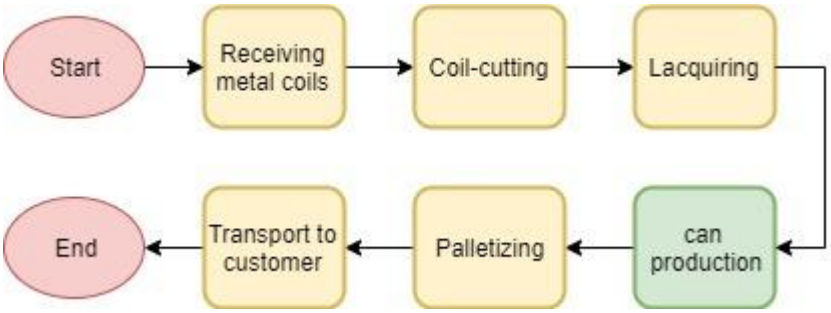


Figure 4: Process flowchart of the 3PC production excluding storage of materials

The production process starts with the receiving of metal coils, which are then placed in the material storage. The metal coils serve as input for the coil-cutting lines, which cut the big coils into sheets. These sheets can differ in thickness, size, color and material. In the next step the metal sheets are lacquered, depending on the specifications of the customer. After lacquering the sheets are placed into storage again until the production commences. The next step moves the sheets to the 3PC department where can production takes place. The 3PC department converts the metal sheets in three-piece cans. The palletizers are connected to the 3PC department, thus after production the palletizing immediately commences. The pallets are placed in their final storage location until shipping starts.

Within the 3PC department there are six production lines, where each line is able to make metal cans with specific circumferences. The height of the cans can be adjusted and may differ between products. Some specifications can be produced on multiple lines, where the choice of a specific line is left to the

planner. The lines operate in parallel to each other, which means that several lines can produce cans at the same time. Each product must be set-up before production can take place on a line. The set-up required depends on the previous product produced on the line. A set-up from one item to another is called a changeover.

1.3 Problem description

Within this research we look at the planning and scheduling process within the 3PC department, which is made difficult by a shortage of information regarding the optimal planning and scheduling choices.

Within the 3PC department there are several steps that must be scheduled. The planners receive a weekly forecast on Monday, which is used as input for the planning. This forecast contains the expected demand of each customer and each product. The production plan and schedule are created manually by the planners based on their own experience and intuition. First, there is a large product mix, where each product has its own scheduling constraints and requirements. Machines must be set-up for each product and the Trivium management estimates the set-up costs to be high. The demand of products is variable, a customer can request a different number of products each week. Secondly, manning is shared between the six different production lines. Not all lines can run at the same time, due to the lack of workers available. Because of the multiple decisions that must be made with each a different effect, the total effect of a scheduling decision is difficult to track. We expect that there is room for improvement in the current schedules. Better schedules balance the inventory and production costs, reducing the total costs of the 3PC department, while meeting demand of the customer.

An important part of the scheduling is batch sizing and sequencing of orders. Future orders can be produced ahead of time, thus making it possible to batch multiple orders in one production run. By combining orders, the number of changeovers can be reduced. Reducing changeovers reduces the cost of changing products but is often paired with an increase in inventory costs. A good balance between the number of changeovers and the lot sizes is needed for an optimal schedule. In short, we need to find a way to create a feasible schedule while ensuring minimal costs. To do this, we must determine the batch size, sequence and period of production for each product.

We define the research problem as follows:

“The current planning and scheduling approach results in non-optimal schedules that do not optimally minimize the inventory and production costs.”

1.4 Research questions

The main problem is caused by the lack of a structured method and a lack of insight in the costs of the production process. While the current schedules do satisfy the demand, it is unknown what the exact costs are for this solution and what the possibilities are for lowering these costs. Therefore, a new planning method must be designed that can create an optimal planning or near-optimal planning.

The problem described in Section 1.3 leads to the following research question:

> ***What can Trivium do to structurally improve their planning process to minimize the production- and inventory costs, while meeting all demand?***

To find an answer to the main research question, we define several questions to be answered.

Q1. What is the current situation at Trivium Deventer?

Q1.1 How is the production process at Trivium structured?
Q1.2 What is the current planning process of Trivium?
Q1.3 What are the objectives and restrictions for the production schedule at Trivium?
Q1.4 What costs are associated with the production schedule?
Q1.5 How is the performance of a schedule currently measured?

Chapter 2 answers question 1 and its sub-questions. To improve the planning process, a clear view of the current production process and planning is essential. This research analyses the production process with the help of data analysis and informal interviews with employees of Trivium. With the main goal being the improvement of the planning process, we work together with team leaders of production and the planning employees. At this point, we also determine what the constraints and objectives are that should be taken into consideration. We identify the Key Performance Indicators that are currently used at Trivium Deventer to evaluate the schedules.

Q2. What literature is available to support improvement of the planning and scheduling process at Trivium Deventer?

Q2.1 How is the scheduling problem at Trivium described in literature?
Q2.2 What methods are available to solve the scheduling problem?
Q2.3 What methods are used in literature to measure and evaluate the production schedule?
Q2.4 How can we cope with uncertainty with regards to scheduling?

In Chapter 3, we conduct a literature study to describe the background of scheduling. Then, we explain several scheduling problems and introduce different models describing this problem. We look at methods to evaluate and measure the performance of a production schedule. Finally, we look at how uncertainty caused by stochastic variability can be dealt with in the model.

Q3. How can the scheduling problem at Trivium be solved?

Q3.1 How can the specific situation at Trivium Deventer be modelled?
Q3.2 What solutions are available to solve the scheduling problem at Trivium Deventer?

Chapter 4 formulates the general model and several solving methods that are usable at Trivium Deventer. We use the data gathered during literature study to formulate the model and alternative solutions.

Q4. What is the best solution for the scheduling problem at Trivium Deventer?

In Chapter 5, we analyze the alternatives and determine which alternative is the best for Trivium Deventer. Historical data is used to create a feasible schedule using a given forecast and determine the ability of each algorithm to create schedules.

Finally, Chapter 6 answers the main research question, discusses the implementation and gives the conclusion and recommendations.

2 CURRENT SITUATION

This chapter describes the current situation as it can be found at Trivium Deventer, to answer the first research question: "Q1. What is the current situation at Trivium Deventer?". Section 2.1 describes the overall production process and the different departments at Trivium Deventer. Section 2.2 explains the 3PC production process in detail. We describe the storage of finished products in Section 2.3. Then, Section 2.4 describes the planning department with its current scheduling processes. We analyze the inventory and production costs associated with the schedule in Section 2.5. In Section 2.6 we discuss the current performance measures used while scheduling. Section 2.7 aims to fully identify the problems encountered during scheduling and in Section 2.8 we conclude this chapter.

2.1 THE OVERALL PRODUCTION PROCESS

Recalling the introduction in Chapter 1, there are four production departments at Trivium Deventer, the three-piece department (3PC), the two-piece department (DWI), the specials department (SP) and the ends department (DA). The coil cutting lines (BKL) and lacquering lines (LAK) supply these production departments.

The process starts with the delivery of materials. Coils are received at the coil cutting lines, where they are cut into sheets. Figure 5 and Figure 6 show a received coil and a pallet of rectangular sheets cut from the coil. These sheets are lacquered and placed into storage until production in Deventer or transported to other Trivium plants for production.

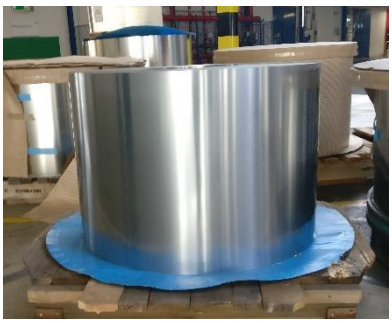


Figure 5: Metal coil as received by Trivium (unpacked).

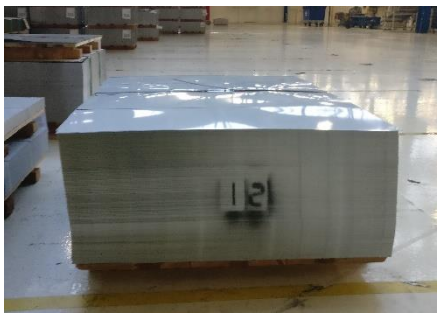


Figure 6: Metal sheets cut for can production.

The DA department uses metal sheets to create easy-open ends for the cans, which are produced for the 3PC department or for usage at other plants. At Trivium Deventer only easy-open ends are produced, which are a specific type of top end. Figure 7 shows an example of an easy-open end, which contains a tab to make opening the can easy.

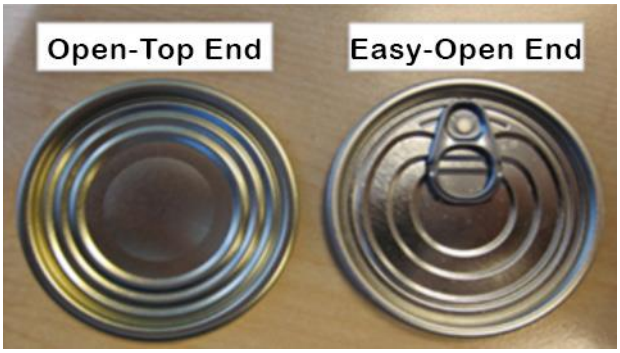


Figure 7: Open top end and easy open end produced at Trivium.

The 3PC department takes the lacquered sheets and ends and combines these into cans. Either the top or bottom end is attached to the can. The other end will be attached by the customer, after filling is complete. The cans are palletized and moved into storage. When the actual order comes in, the pallet is retrieved and shipped to the customer. Section 2.2 elaborates further on the production process in the 3PC department.

The SP department produces products with sizes and shapes that do not fit the generic item specifications. Syrup cans and cigar boxes are two of the products created in the SP department. The SP department receives (printed) sheets through intragroup deliveries and creates the final can. The products in the SP department are make-to-order, as the products are specific for the customer.

Finally, there is the DWI department. This department receives metal coils, which are directly fed into the production line. Within the production line, these coils are transformed into cans where the bottom is integrated in the body, recall Figure 2. These cans are then palletized and moved to storage. As with the generic products, cans are kept in storage until the actual order comes in.

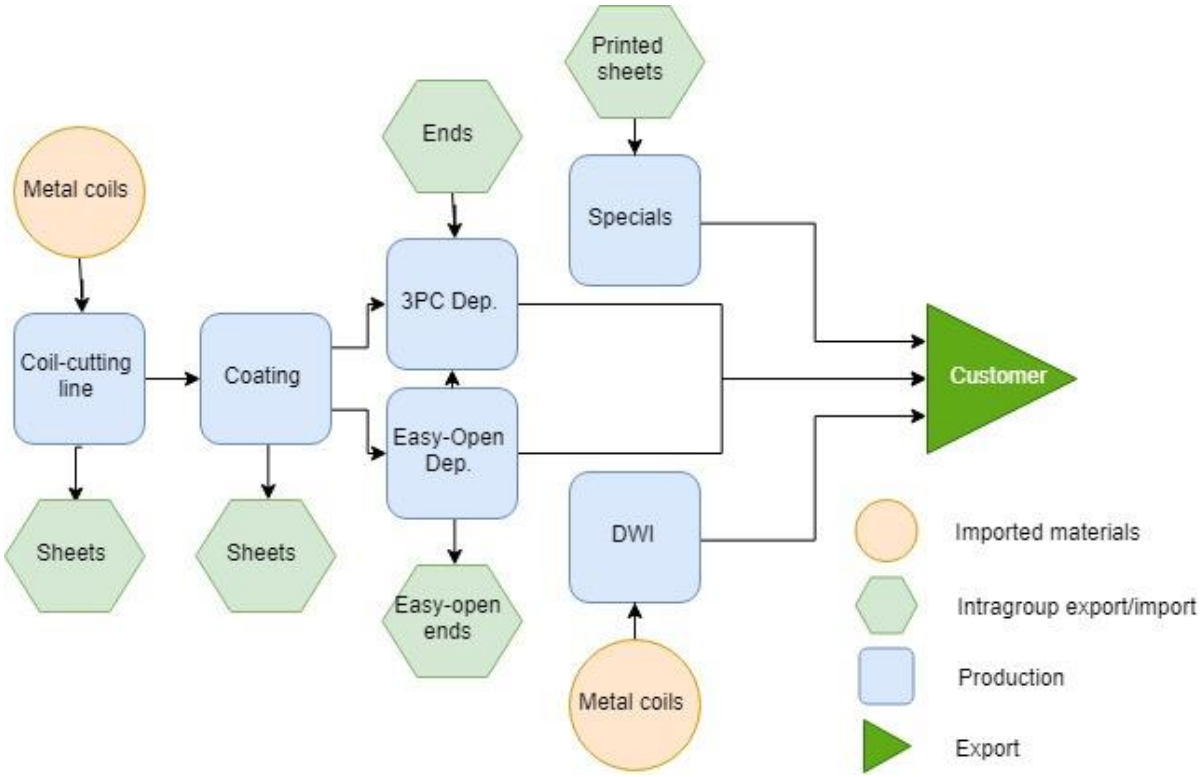


Figure 8: Overview of Deventer departments and connection. Storage is excluded.

Figure 8 shows an overview of the material movements, excluding storage of materials. In-between any of the production steps, the materials might be put into temporary storage. Trivium Deventer is not only dependent on its own customers, but also on customers of other Trivium locations due to the intragroup deliveries. Trivium receives printed sheets and ends from other plants, but at the same time they supply other plants with blank and coated sheets and easy-open ends.

2.2 THE 3PC DEPARTMENT

This research is focused on the 3PC department, which produces three-piece cans and currently is responsible for the production of approximately 50 different products, excluding printed cans. Section 2.2.1 introduces the 3PC department in detail. In Section 2.2.2 we describe the manning situation and in Section 2.2.3 we explain the changeovers between items. Section 2.2.4 explains the dependencies on preceding departments. In Section 2.2.5 we explain the equipment effectiveness of the production lines. Last, Section 2.2.6 describes the issues faced directly within the 3PC department.

2.2.1 Line and item specifications

There are 6 parallel lines that produce the different types of cans, each with their own specification. Each line consists of the same type of machines but is configured for a specific set of specifications.

Figure 9 shows the components of each production line. Each line starts with a sheet feeder (1), where the pallets with metal sheets are placed. Each plate is fed into the double slitter (2), which cuts the metal sheets horizontally and vertically to create blanks that will make up the body of the metal cans. The blanks are fed (3) into a body welder (4), which forms the blanks into a cylindrical body, welds the seams together and covers the seam (5) with a powder to protect the seam from corrosion. The formed body enters the curing zone (6), which is required to cure the powder. When cans with a low height are required, the breaker or wobble cutter (7) breaks the metal body in two parts, creating two bodies with half the height of the original body. A necker (8) compresses the top of the cans inward, which enables stacking of the cans. The flanger (9) creates a ridge on the top and bottom of the can, which is needed when attaching the ends. Before the cans enter the seamer, they might be beaded. The beader (10) adds several ridges to the body, which increases the structural strength of the finished can. After the beader, the seamer (11) connects a top or bottom end to the body and closes the can on one side. At the end of the line, the cans are marked with a code (12), tested (13) and finally palletized (14).

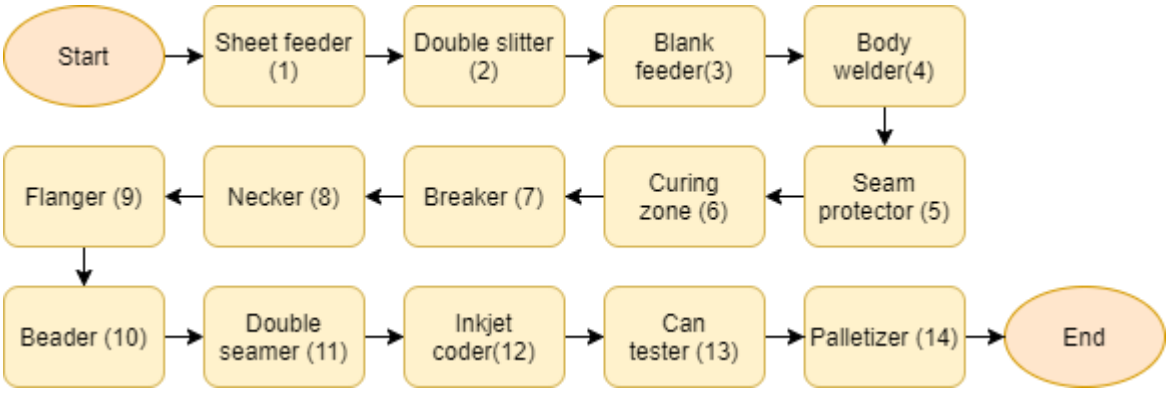


Figure 9: Components of each 3PC production line

Figure 10 shows a metal can at several stages of production. On the bottom is the metal sheet (2), created by the double slitter. Then from left to right, the welded body (4), a necked can (8), necked can with flanged top and bottom (9) and a can with easy-open end attached (11).

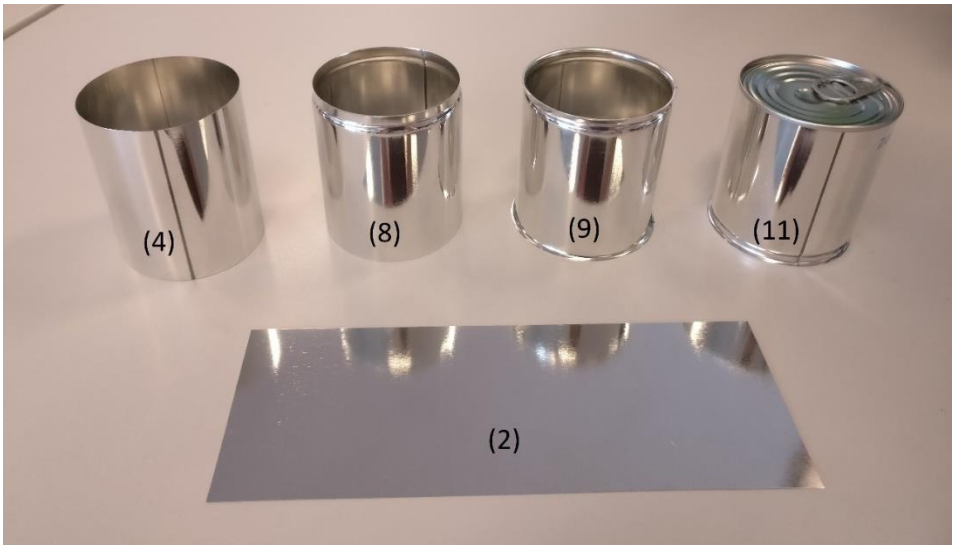


Figure 10: Cans after every stage of the production line. Numbers correspond with the machine they are output from in Figure 9.

Table 1 shows the production line and generic item specifications possible on each production line. Each production line can produce cans with a set diameter, but the height of the final can is variable. A specification is identified by a number for the diameter and height of the can. Specification **XX**s**YY** has a diameter of **XX** mm and a height of **YY** mm. For example, production line 814 produces cans with a height of ***CONFIDENTIAL*** and ***CONFIDENTIAL*** mm. Production line 846 produces the largest number of unique SKUs. The maximum production speed of a line is dependent of the height of the produced cans. Production line 846 can produce ***CONFIDENTIAL*** cans per minute with a height of ***CONFIDENTIAL*** mm, but only ***CONFIDENTIAL*** cans per minute with a height of ***CONFIDENTIAL*** mm.

*Table 1: Production lines and specifications. A * indicates printed items are also created with that specification.*

Line	Diameter	Height	SKUS	Units/min (max)
51DEV807 (807)	CONFIDENTIAL		3	CONFIDENTIAL
51DEV808 (808)			7*	
51DEV810 (810)			11*	
51DEV814 (814)			4	
51DEV828 (828)			10*	
51DEV846 (846)			18	

2.2.2 Manning

On average, it is possible for four production lines to run at the same time. The number of lines available can be determined before scheduling commences, based on the availability of personnel. During vacations and sickness, the availability of personnel is lower and often only three production lines can run at the same time.

Within the 3PC department, the normal working week is from Monday to Friday, 24 hours per day. To ensure 24 hours of production every day, there are three eight-hour shifts every day, which makes for 15 shifts per workweek. When there is a shortage of production capacity, it is possible for production to continue during the weekend, in overtime. During the weekend there are five shifts of overtime available for scheduling, but these must be requested beforehand.

2.2.3 Changeovers

When a line switches from producing one product to another, a changeover is required. Changeovers take up production time and employees that are already high in demand. Due to the difficulty of a changeover, the production department prefers a changeover to take place during the morning shift. The morning shift is considered ideal as during this time there are additional people available to complete a changeover in case of problems. The changeover coordinator prepares the changeovers and can help the employees if needed. We divide the changeovers in two types of changeovers, a big changeover and a small changeover. Examples of small changeovers are changing the pallet type, changing the double cutter or switching the seamer. For a big changeover, several of these components must be changed at the same time. When the generic specification of two products is different, a big changeover is always required. These cans differ in size and form so much, that multiple components must be changed. A big changeover requires four employees for the full duration of a shift.

For products whose specification is the same, there are three possibilities. Either no changeover is needed, only a small changeover is required, or a big changeover is needed. Appendix A identifies the type of changeover between products that have the same specification. The duration and manning cost of a changeover is determined together with the changeover coordinator. It is possible to reduce the time of a changeover by scheduling more employees to complete a changeover, but this leaves less employees available to run the production lines. After a changeover, the production capacity of a line is shortly reduced as the line needs to be started again after a changeover. During start-up additional modifications might be needed to ensure the production line runs smoothly.

2.2.4 Preceding departments

The 3PC department receives its materials from the coil cutting lines, the lacquering lines, the ends department and from other Trivium production plants. The 3PC department cannot produce cans without these materials and is dependent on the preceding departments and lines. The lead time for receiving a coil is approximately six weeks. When the coil is received it must be cut into sheets and coated, before the 3PC department can use the sheets. The lead time of cutting and coating is approximately 2-3 days. During planning, we must take the 6 weeks material lead time into consideration.

2.2.5 Equipment effectiveness

Trivium Deventer uses the overall equipment effectiveness (OEE) to determine the effectiveness of the lines. OEE is a metric first introduced by Nakajima (1988) and is geared towards machines. The following definition of the OEE is used at Trivium Deventer:

$$OEE = \frac{net\ production\ time}{net\ production\ time + unidentified\ stops + identified\ stops}$$

Figure 11 shows the specific components of OEE at Trivium Deventer. The total time used consists of the net time used for production, the non-identified stoppages and identified stoppages. The theoretical production time of a specification is determined by dividing the produced number of cans by the theoretical production capacity, as given in the last column of Table 1.

Changeovers are also part of the OEE, which can have a big effect on the OEE of a production line. Thus, a rise in OEE must be compared with the number of changeovers to get an accurate view of the increase in production efficiency.

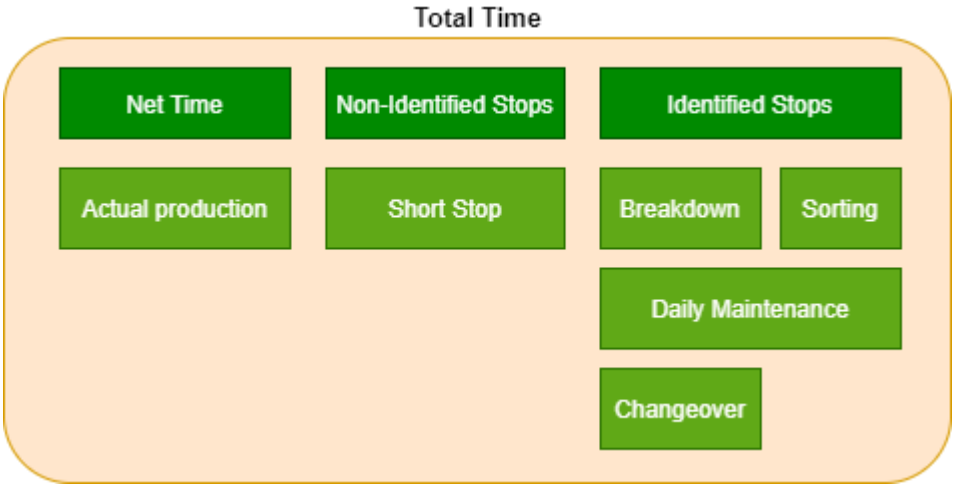


Figure 11: Components of total time in OEE at Trivium Deventer

Table 2 shows the average OEE in 2018 of each general specification. On average, the OEE at Trivium Deventer in the 3PC department is ***CONFIDENTIAL*** %, but there is a strong variation between lines and item specifications. The one specification that can be produced on two lines, the ***CONFIDENTIAL*** mm diameter with a height of ***CONFIDENTIAL*** mm, has an OEE of ***CONFIDENTIAL*** % on line 810 compared to an OEE of ***CONFIDENTIAL*** % on line 814. As the base production speed and efficiency of line 814 is higher than that of line 810, the planner tends to schedule this specification on line 814 when possible.

Table 2: OEE of each item specification in 2018

Line	Specification	OEE 2018
807	CONFIDENTIAL	
808		
810		
814		
828		
846		

2.2.6 Issues

Within the 3PC department several issues are found. The production lines consist of old machines, which are not in perfect state anymore. Breakdowns happen frequently and are difficult to predict by the production department. This affects the production capacity of a production line and makes it difficult to create a schedule that can withstand all unexpected changes. There is limited production capacity due to the limited number of employees available. A changeover requires additional manning, which limits the number of production lines that can produce in parallel.

Changeovers result in set-up time. Directly after a changeover, the line produces at a lower speed. Effective production time and speed is lost when completing changeovers. Moreover, there is the problem of incorrectly produced products, also called Held for Inspection (HFI) pallets. Pallets stay in HFI until inspection takes place and a decision can be made regarding the further processing of the pallets. When pallets are placed in HFI they cannot be transported to customers, which temporarily decreases the production levels. Additionally, HFI pallets take up storage space that could be used for other items.

2.3 WAREHOUSING DEPARTMENT

The storage of finished products takes place in two internal warehouses and two external warehouses. After production, packaging machines palletize the cans, where the pallets are bound with straps and wrapped in plastic, if required. There are two general types of pallets that are created, short packs and tall packs. Tall packs are approximately twice the height of a small pack. Figure 12 shows a tall pack and short pack side-by-side. The palletizers are used by the DWI and 3PC department and have a limited throughput. It is not possible to schedule short packs on all lines, as this will cause capacity problems.

Figure 12: Palletized cans, with on the left side a tall pack and on the right side two short packs stacked together.

Figure 13: Palletized cans waiting at the end of the conveyer belt for pick-up.

When production and palletizing is done, a conveyer belt moves the pallets to the entrance of the warehouse, where they are buffered until an automatic guided vehicle or forklift picks up the pallet. At the entrance of the warehouse, the pallets are picked up and transported to their final location. Figure 13 depicts several pallets waiting for pick-up and in Figure 14 we see the first part of the internal warehouse. The warehouse is separated in 342 small areas, which together have a capacity of approximately 27,000 short packs. The maximum height of a stack of tall packs is three pallets, while the maximum height of a stack of short packs is six pallets. The yellow lines on the floor denote the separate areas used for storage. Theoretically, every area should only house one batch of products,

to ensure first-in-first-out delivery. Multiple batches of the same product are sometimes stored in the same area, due to storage constraints. When this happens, the first-in-first-out delivery is often violated, but this is discussed with the customers to ensure it is not a problem.

The two external locations available to Trivium are Van Opijnen and Kaminsky. The Kaminsky warehouse is only used for DWI cans and only for the customers Saturn Petcare Bremen and Seitz. As Kaminsky is only used for specific customers of the DWI department, we do not consider this further in the research. Van Opijnen is only used for finished cans but can house both DWI and 3PC cans. When pallets are stored at the external location, the pallets are picked up at the conveyer belt and brought to the loading docks. There, the pallets are loaded into transport trucks from Van Opijnen. The exact capacity in pallets at Van Opijnen is unknown, but there is a total area of 8,650m^2 available. Based on previous years data, we estimate the capacity at Van Opijnen to be 8,650 short packs, where Van Opijnen requires 1 M^2 for every pallet.

Figure 14: Finished cans warehouse of Trivium Deventer

Inventory is only stored at Van Opijnen when there is not enough space left at Deventer. According to the warehousing department, not enough space is when the internal warehouse is at approximately 85% of its capacity. The warehousing department indicated that at a storage level of 85%, they start moving pallets to Van Opijnen. Figure 15 shows the number of pallets in storage at Deventer and Van Opijnen from January 2018 until February 2019, compared with the percentual fill rate at Deventer. The fill level of the Deventer warehouse fluctuates between 60% and 93%, with the average being 78%. Data shows that a high number of pallets in Deventer does not indicate a high number of pallets at Van Opijnen. A fill rate of 85% is reached several times in the period January 2018 and April 2019, but no direct increase in stocks in Van Opijnen is found.

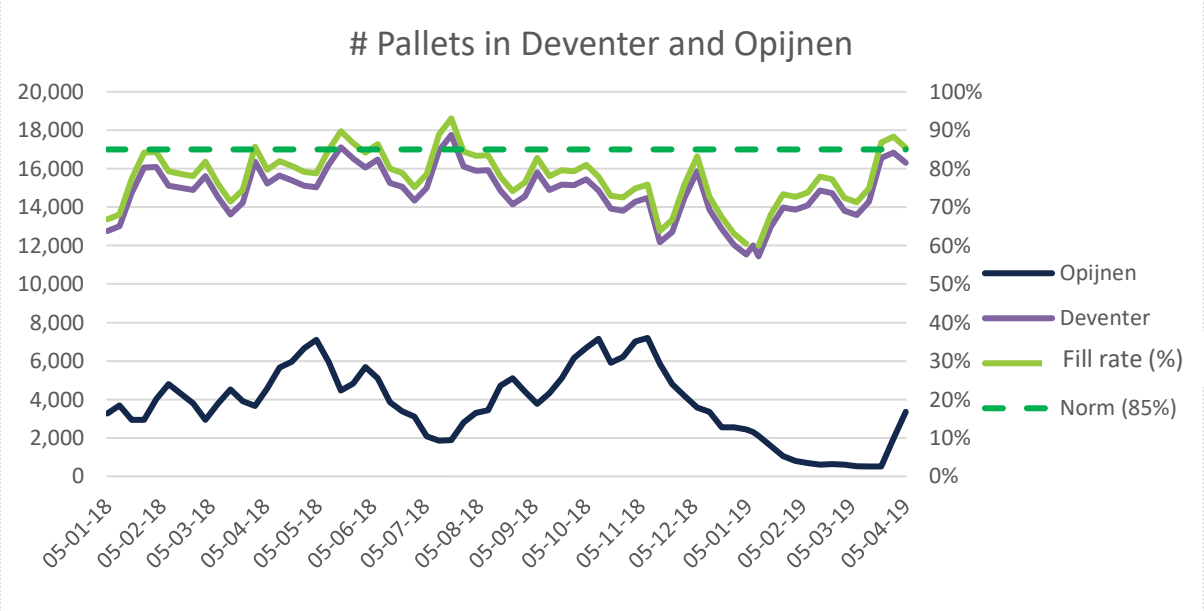


Figure 15: Number of short and tall packs stored in Deventer and Opijnen from January 2018 till April 2019

We expect the absence of a statistical relationship to be caused by a delayed response of the inventory levels at Van Opijnen. Pallets are not moved from Van Opijnen to Deventer when the inventory levels are low. With an average throughput of six weeks, when pallets finally leave Van Opijnen, the inventory level in Deventer has already caused additional pallets to be moved to Van Opijnen. Pallets stay at Van Opijnen until the customer needs the product and are then delivered directly to the customer. Likely, inventory needs to be low for longer periods of time to decrease the average inventory levels at Van Opijnen, but there is no data to support this. The figures above cover the full inventory in the warehouse, not only inventory caused by the 3PC department. When solely focusing on the 3PC department, we find the same results.

The main issue in warehousing is the limited internal storage available. The finished cans storage is constantly at high levels, which causes inventory to be stored externally. During production planning, the inventory levels of the other departments are not considered. This makes estimating the total finished cans inventory impossible. High costs are associated with external storage, which we elaborate on in Section 2.5. Other factors influencing the storage capacity and inventory costs are stock control and safety stock levels. During our research, we consider the storage space and storage costs.

2.4 Supply chain department

Currently, the planning and scheduling process is completed manually and relies upon the experience and intuition of the planner. To ensure a feasible production plan and schedule, the planners currently complete the following steps:

1. Download sales forecast (Monday)
2. Update production plan and schedule (Monday/Tuesday)
3. Update required materials planning (Tuesday/Wednesday)
4. Implement corrective measures if schedule is infeasible

The planning process starts with the forecasted demand, which is compiled by the Internal Sales department at Trivium Deventer. Most customers submit their own forecasts, but for some the sales manager predicts the forecasted demand.

The forecast is delivered in the form of a rolling forecast, which must be updated on Friday by all Internal Sales employees. The planners download a new forecast every Monday morning, which is then used to update the production plan and schedule. The forecast is updated constantly and generally becomes more precise closer to the actual sales date. There is no deadline for how early demand must be forecasted, which results in differing forecast deadlines between customers. When a customer wants to buy a product, the forecast is changed into an order. At this point it is still possible to change the number of products required, but only if there is enough production capacity and material available before the order deadline.

With the new forecast available, the planner first updates the production plan. The plan specifies the amount of production per week for every item and covers a 12-week horizon. Figure 16 shows part of the week planning file that is used. We see the production line and products on the left side, in this case line 808 and three different products. The right side shows from top to bottom for each product the forecasted orders (D), planned production (P) and expected inventory (I). When the inventory at the end of the week is not enough to meet the forecasted demand, the number turns red, signaling the planner that additional work is needed.

LINE				Month	jan-19					feb-19
				Week	1	2	3	4	5	6
808		Total	73x70x109.2	D	-	-	-	32	-	-
	FLN0973502			P						
				I	412	412	412	380	380	380
		Total	73x110	D	-	-	-	-	20	100
	FLN1173813			P		600				
				I	348	948	948	948	928	828
		Total	73x110	D	31	39	43	43	41	-
	FLN1173814			P		150				
				I	273	384	341	298	257	257

Figure 16: Planning file of 3PC line on line 808 for three different products. D = forecasted demand in that week, P = planned production in that week, I = Inventory at end of week.

This plan is then used to create a weekly schedule. In the schedule we specify the sequence of production and the number of products. The schedule spans approximately 12-weeks, with the first six weeks considered most important. The first six weeks are important because of the ordering of materials and lead time of delivery. The production schedule is updated weekly and becomes more precise closer to the actual week of production. Figure 17 shows a part of the production schedule of production line 808, where we can see that product FBN8270303, FLN1173814 and FLN1173813 are scheduled for production. The top rows show the date and shift where the production is scheduled. The gray colored squares indicate production is planned during that shift, while a yellow colored square indicates a changeover is scheduled. When nothing is scheduled, the square stays white. The number in the gray box indicates the scheduled number of cans to be produced in thousands, which is based upon the expected number that can be produced during one shift. When a production shift is finished, the planner adds the actual number of produced cans in a green or red box, depending on if production was good or bad.

Lines	Monday			Tuesday			Wednesday		
	7 January 2019			8 January 2019			9 January 2019		
	N	M	A	N	M	A	N	M	A
230808	FBN8270303				FLN1173814			FLN1173813	
	130	170	170	170	Change	80	70	80	80
	135	187	157	187		80	123	96	85

Figure 17: Production schedule and results on line 808. N = Night shift, M = Morning shift, A = Afternoon shift

The expected number that can be produced per shift is determined looking at the budgeted OEE of 2019 and the theoretical maximum production capacity of the production lines. The budgeted OEE is based upon the OEE of 2018 and goals set by the Supply Chain department. The planning capacity per shift is then determined by calculating the max capacity per shift and multiplying this with the budgeted OEE. The numbers are discussed with the production team, to ensure they are feasible. Based on this discussion, they may be increased or decreased. Table 3 shows the production capacity per shift for each item specification that is used by the planner.

Table 3: Planning capacity of the lines. Calculated capacity is rounded up.

Line	Item specification	Budget OEE 2019	Max Capacity per shift (x1000)	Used by planner (x1000)
51DEV807 (807)	CONFIDENTIAL			
51DEV808 (808)				
51DEV810 (810)				
51DEV814 (814)				
51DEV828 (828)				
51DEV846 (846)				

After the scheduling is done, a material requirements planning is made. The planner checks the inventory and what is needed for production. The material requirements are communicated to the corresponding departments, which will order materials or schedule production of materials if necessary. Due to the lead time of materials as mentioned before, the planner must be careful to not make big changes in the first six weeks without checking the material availability.

During the week, the schedule is constantly monitored. Due to uncertainty in demand and production changes may be needed in the schedule. This is done by the planner and immediately communicated with the other departments, as changes in the schedule might affect other departments.

2.4.1 Planning and scheduling constraints

When making the production plan and schedule, several constraints must be considered by the planner, which we split into hard and soft constraints. Hard constraints must be fulfilled in the schedule, while soft constraints are preferred but not required. We defined these constraints by looking at data and talking with people from the Supply Chain department.

Hard constraints

- Products must be produced on specific lines
- Sequence of production must be considered as set-ups are sequence dependent
- Limited employee availability
- Maximum inventory capacity
    - Constrained by a limited amount of storage space internally and externally
    - Constrained by management to meet monthly/yearly targets.
- Limited production capacity of the lines
- No late orders in the schedule
- 6-week lead-time of materials
- Maximum capacity of the palletizers

Soft constraints

- Maximum of one big changeover per day. These changeovers only happen during the morning shift
- Minimum production time
    - Completing a changeover for only one or two shifts of production is not preferred.

2.4.2 Issues

An issue at the planning department is the lack of information regarding the inventory and production costs. Without this information, it is difficult to make an informed decision. The production employees want a low number of changeovers, as this makes the workload lower and continuous production is easier. For the planner, smaller lot sizes and more changeovers are sometimes desired, to keep the inventory levels low. These different objectives need to be balanced.

Because of the parallel production lines that are constrained by secondary resources, the planners need to decide which line has priority. Combined with the effect of lot sizing and lot sequencing, the planner cannot determine the effect of scheduling changes efficiently.

2.5 PRODUCTION, INVENTORY AND OUT-OF-STOCK COSTS

This section provides the cost analysis, which uses data from the financial department and production department regarding the different costs. Section 2.5.1 describes the production costs, Section 2.5.2 gives an overview of the inventory costs and in Section 2.5.3 we describe the out-of-stock costs. Figure 18 gives an overview of the production and inventory costs and specifies the components of each cost that we consider.

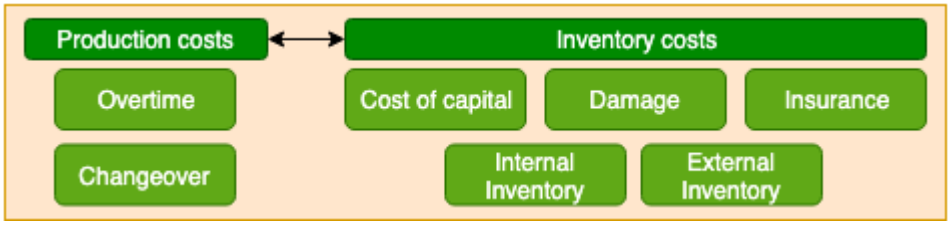


Figure 18: Overview of production and inventory costs and their components

2.5.1 Production costs

The first type of costs that are considered are the production costs. The objective is to find the costs associated with a specific schedule, so only costs that vary with the production schedule are looked at. This means that production costs such as powering the lines, cost of materials and line maintenance are not considered. Figure 19 shows the composition of the production costs at Trivium Deventer, which we consider during our research.

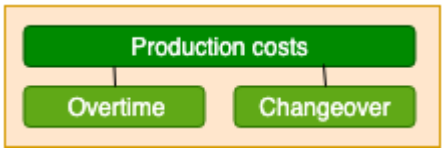


Figure 19: Composition of production costs

Changeover costs are caused by different products requiring different settings on lines. It takes time and extra personnel to complete a changeover. By scheduling bigger lot sizes, the number of changeovers can be decreased, which means that the schedule affects the costs caused by changeovers. Changeover costs are primarily caused by labor costs. Energy costs can also be considered part of the changeover costs, but according to the production department these are negligible in comparison with the changeover costs. The labor costs per hour at Trivium Deventer are approximately $€CONFIDENTIAL$ per hour per employee. This is an approximate value as employees with different levels of experience receive different wages. The exact value was determined in consultation with the finance department, while looking at the wage payments of previous months. Knowing the cost of one hour of labor, we can calculate the cost of the two different changeovers. Every type of changeover always requires a full shift of time, which is eight hours. The big changeover requires four employees, compared to one employee for a small changeover. Table 4 shows the overview of costs for each changeover. When no changeover is needed, no extra costs are accumulated.

Table 4: Overview of changeover costs

Changeover	# employees	Hours/shift	Cost
Big changeover	CONFIDENTIAL		
Small changeover			

The other costs considered are overtime costs. Normal production takes place from Monday to Friday. When production during the week is not enough to meet demand, it is possible to plan overtime during the weekend. The shift workers are paid extra for overtime, which gives rise to extra labor costs. At Trivium, working overtime means employees gets paid an additional 65%, for a total of 165% wage per hour. If an employee works overtime during more than one day in a six-week period, the employee receives an additional 100% on top of the basic wage, instead of 65%, for all hours worked in overtime. The finance department has kept track of labor costs and indicated that on average an

hour of overtime costs €$\boldsymbol{CONFIDENTIAL}$. This combines the cost of working overtime once in six weeks and working overtime multiple times. This is roughly 169.8% of the basic wage received by employees, which is between the percentage of working overtime once in a six-week period and working overtime several times in a six-week period. This implies that when overtime happens, it often happens more than once in the six-week period.

2.5.2 Inventory costs

The inventory costs at Trivium Deventer consists of 5 different components, based on a division by Lambert and Stock (1993). The costs are determined by looking at each of the components and determining their connection to the production schedule.

Cost of Capital

To determine the cost of capital, Trivium Deventer uses the weighted average cost of capital. It balances the cost of debt and cost of equity into one value. As it averages the cost of the different types of capital, this value is used to determine the cost of keeping inventory. All inventory has a monetary value and by keeping products in inventory it means this money is not available for investments or paying off debts. At Trivium Deventer the cost of capital is estimated to be $\boldsymbol{CONFIDENTIAL}$% per year. This is a group estimate and not specific for the Deventer plant. The cost of capital is a percentage of the inventory value, which means that for every product the cost of capital in euros differs. The weighted average cost of capital (WACC) is a measure of capital, which averages the cost of debt and common equity (Leach & Melicher, 2015). We calculate the WACC with the following formula:

$$Weekly\ cost\ of\ capital: WACC_w * \#cans_i * costprice_i$$

$WACC_w = weekly\ WACC$
$\#cans_i = average\ number\ of\ cans\ of\ product\ i\ during\ the\ week$
$costprice_i = costprice\ of\ product\ i\ per\ can$

The WACC of Trivium Deventer is determined by talking to the finance department. This is a value that the plant receives yearly from Trivium Group. For 2019 the WACC is set at $\boldsymbol{CONFIDENTIAL}$% per year, which is a WACC of $\boldsymbol{CONFIDENTIAL}$% every week.

Insurance

Insurance is a form of inventory service costs. Insurance rates are not necessarily dependent on the inventory levels, as it depends on the agreement the company has with the insurance provider. In the long term, the insurance rates are often flexible, as a company can decrease their insurance coverage if they keep constant lower inventories.

At Trivium Deventer, the insurance is not considered a flexible cost on short term. The insurance rate is fixed during the year and is only changed on a yearly basis. Lowering the average inventory during the year may affect the insurance rates on the long term. Thus, it could be beneficial to lower the average inventory levels when possible with regards to the insurance costs.

We do not include the insurance costs in our final model, as the costs are not directly influenced by the 12-week schedule. Nonetheless, it is beneficial for Trivium Deventer to know what costs could be decreased by lowering the long-term inventory levels. Trivium Deventer receives a report from Trivium Group, which includes the total insurance costs. After talking with the finance department, we determined the insurance costs to be €$\boldsymbol{CONFIDENTIAL}$ during the year 2018. This covers the inventory insurance and the consequential damages insurance, of which roughly half of the costs are associated with the inventory insurance. This means the inventory insurance cost in 2018 at Trivium Deventer was roughly €$\boldsymbol{CONFIDENTIAL}$ With a total average inventory value of €$\boldsymbol{CONFIDENTIAL}$ in

Deventer (cans, ends, chemicals and all other parts), the insurance costs in 2018 were 0,8% of the average inventory value.

Damage

Whenever products are stored, it is possible that damages are incurred during handling. The cost of damage should only include those handling costs that are variable with the inventory levels. Damage incurred during shipping to customers is not included, as shipping is not dependent on the inventory level but on the throughput of products (Lambert & Stock, 1993).

The cost of damages as a result of handling consist of two components. There is the cost of lost products, which cannot be shipped anymore due to the damage. Then there is the cost of sorting the damaged pallets. When only a part of a pallet is damaged, it is possible to sort the pallets and remove the damaged cans. The cost of sorting is equal to the hours required to sort a pallet multiplied by the labor costs as determined in Section 2.5.1.

Trivium Deventer has data available regarding the number of accidents from 2017 onwards. Each accident concerns one pallet. A total of 281 accidents were registered between January 2017 and March 2019 within the finished products storage. Using the data, no connection is found between the number of accidents and the number of pallets stored in Deventer. A possible reason for this is the continuous high storage levels, which fluctuate between 65% and 85%.

Internal Inventory

The internal storage costs are composed of handling costs that rise when the inventory increases. The internal warehouse is owned by Trivium Deventer and only used by them. Empty space is not rented to another company and thus not a source of income. Location costs are fixed and do not depend on inventory. A possible flexible cost is the handling, which might increase when more inventory is being held. Relocation costs are an example of extra costs due to high inventory (Lambert & Stock, 1993). At Trivium Deventer inventory handling is done by automated guided vehicles and forklifts. The forklifts are manned by personnel from Trivium Deventer, but in case of shortage it is possible to hire temporary workers. Figure 20 shows the total hours of employees spent in warehousing compared with the average number of pallets in stock in the same week. Both numbers concern all warehouses, as no data was available of only the 3PC warehouse with regards to the working hours. The hours spent on warehousing include temporary workers and Trivium employees.

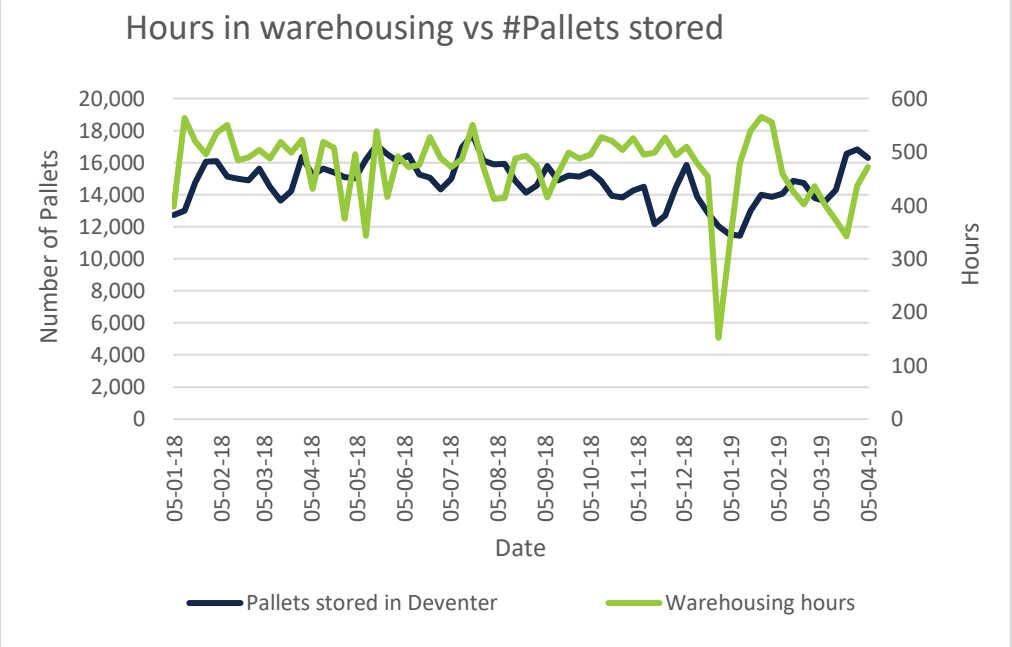


Figure 20: Hours spent on warehousing compared to the number of pallets in storage

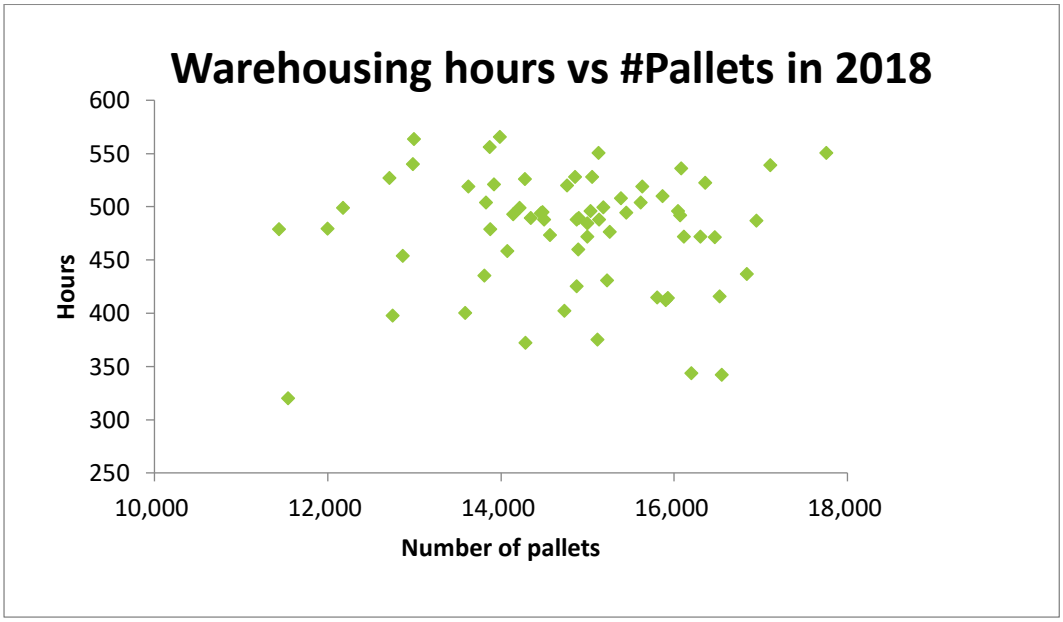


Figure 21: Scatterplot comparing the warehousing hours and the number of pallets

The number of hours in warehousing does not necessarily increase with the number of pallets in stock. Figure 20 shows both the hours spent on warehousing and pallets in storage, which shows no immediate connection between the two. The corresponding scatterplot in Figure 21 shows a spread of data, where no relationship can be identified. Lambert and Stock (1993) remarked that the costs of internal storage tend to be fixed or are variable with the throughput of the facility. Figure 22 shows the hours of warehousing versus the throughput of the warehouse. A slight upward trend is seen in the data, with a correlation between the in/outbound pallets and warehousing hours of 0.59. While the correlation is moderately high, the significance of this result is 0.6. We cannot claim a relationship exists, as the significance of the result is too low to support our hypothesis. While temporary workers

are used sometimes, this cannot be attributed to the level of inventory and thus does not belong to the internal storage costs determined here.

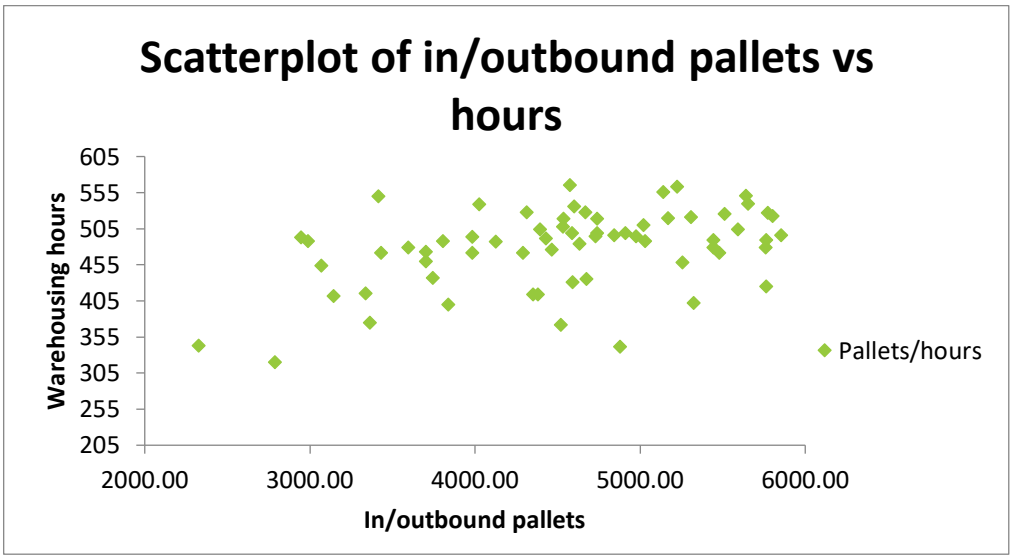


Figure 22: Scatterplot and regression line of in/outbound pallets vs hours in warehousing

Another problem for the internal inventory levels are the HFI or blocked pallets. HFI pallets take up space in the warehouse but are not usable as long as they are blocked. The average number of HFI pallets in the period January 2019 till April 2019 was 866 pallets.

External Inventory

The external inventory of Trivium Deventer is kept at Van Opijnen. Table 5 shows the given and calculated storage costs at Van Opijnen. In 2018 the throughput of pallets at Van Opijnen was determined to be 6 weeks on average. Based on a 6 weeks throughput, the average cost of a pallet stored at Van Opijnen is € $\boldsymbol{CONFIDENTIAL}$. The transport and handling costs are independent of the duration of storage but increase with the number of pallets transported. The storage costs are dependent on the number of pallets and the duration of storage. The transport and handling costs are over half of the average cost of a pallet at Van Opijnen, making the initial decision to store the pallet at Van Opijnen costly. As an example, consider the difference in costs between storing one pallet at Van Opijnen for 8 weeks and storing two pallets for each four weeks. The storage cost in both cases would amount to € $\boldsymbol{CONFIDENTIAL}$, but in the second case the handling and transport costs are doubled. Storing one pallet for 8 weeks incurs a cost of €$\boldsymbol{CONFIDENTIAL}$, while storing two pallets for 4 weeks each, incurs a cost of €$\boldsymbol{CONFIDENTIAL}$.

Table 5: Summarized inventory costs of Van Opijnen

Activity	Costs 2019
Storage space (per pallet per week)	CONFIDENTIAL
Handling costs in- & outbound (per pallet)	
Transport (per pallet for a full truck)	

2.5.3 Out-of-stock costs

Whenever Trivium does not have the capacity to fulfill an order on time, they contact the customer to discuss alternative solutions. Alternative solutions include partial delivery of products or delivery from another Trivium Plant. In both cases higher transport costs might be incurred, to resolve the out-of-

stock situation. When a customer cannot agree to a late delivery, a complaint may be filed. Unsatisfied customers may decide to leave Trivium and move to another packaging supplier. We do not assign out-of-stock costs, but instead require all demand to be fulfilled on time. The production lines have overcapacity and should be able to fulfill demand. Scheduling items late is not permitted and is only done after discussion with the customer.

2.6 Performance measures

To measure the performance of a schedule several indicators are used. Trivium Deventer has as goal to minimize the total costs, while meeting customer demand. Currently, the main performance measure used while scheduling is the inventory value. The planner ensures the inventory level is balanced and below limits set by the company. Another important performance measure is the number of late orders. Late orders are not allowed in the initial schedule, meaning a late order makes the schedule infeasible.

The planning department does not directly calculate the total cost of a schedule, nor do they track the number of backorders. On Trivium Group level the performance is evaluated by looking at the average inventory value during the year. On plant level, currently the inventory levels, OEE and days inventory outstanding (DIO) are considered. At the end of the week, the planner determines how many items were produced versus how many items were planned for production.

The OEE and actual production percentages are measures that we can only calculate after production has taken place. This makes it difficult to use these measures to determine the performance of schedules created by our scheduling algorithm. To determine the performance of our scheduling algorithm, we use the inventory levels and DIO. Additionally, we use the number of changeovers per week as a performance measure and the distribution of production throughout the weeks.

2.7 Problem identification

Recall the problem description in Chapter 1. This section elaborates on that and summarizes the problems related to scheduling. There are two main problems which have the highest impact on the cost of production.

Idle time of the production lines (1)
Idle time of the production lines can have different causes, but only the number of changeovers is directly affected by the schedule. Other causes of idle time are sickness of employees, breakdowns of the production lines and lack of material. Changeovers have a big impact on the idle time, as a changeover causes machine stop of approximately eight hours and is associated with high manning costs.

High inventory levels (2)
The high inventory levels are caused by the fact that more items are produced than that there are shipped. As not all inventory can fit into the internal warehouse, it is stored externally, which gives rise to high costs. There is no insight into the costs associated with inventory and what the optimal lot sizes are. A certain level of inventory is required due to the safety stock of each product, which is needed because of uncertainty in demand and production.

Main causes
The main cause of the problem is that lot sizing and sequencing is not integrated, which makes it impossible to balance the number of changeovers and inventory level adequately. Next to this, there is a lack of knowledge on the tradeoff between inventory and changeover costs. As the inventory

levels and number of changeovers are dependent on each other, these must be balanced to get an optimal schedule. This makes it challenging to create optimal schedules that are robust.

Scope
For the scope of this research, we consider the planning and scheduling of the 3PC department at Trivium Deventer. We aim to integrate scheduling and lot sizing, while keeping in mind the secondary resource constraints of the 3PC department. The focus is to create a method that can solve the planning and scheduling problem. The resulting schedule should cover at least 12 weeks of production. The plan and schedule consider the preferences of the supply chain department and production. The preceding departments are out of our scope, but we do consider the lead times of these departments.

The week plan covers the number of products produced per week for each line but does not immediately cover the manning capacity and sequencing of products. The week schedule indicates the lot sizes and sequences the products. We consider the manning capacity and ensure stock target are met.

Deliverables
As a deliverable we create a proof-of-concept method that supports the production planners in their tasks. The method replicates the process of lot sizing, sequencing and scheduling. A hierarchical and integrated solution is presented. We present several alternative solution methods and compare the results the find the best option. The aim is to use this solution to give new insights in the planning and scheduling processes at Trivium Deventer. Trivium desires insights in the production sizes and stock targets in a close to optimal situation.

2.8 Conclusion

This goal of this chapter is to gather all relevant information required to be able to describe the current situation at Trivium Deventer.

The planning and scheduling process consists of several steps. It starts with a forecast that is made based on customer inputs. Using the forecast, a general production plan is created that gives an overview of which weeks each product is created and the batch sizes of production. Next, the planners determine the production sequence within each week. Each product is assigned specific shifts, while ensuring the scheduling constraints are met.

While scheduling production, the planner must keep in mind the limited availability of manning, the material availability, the production line availability and the costs generated by the schedule. During planning and scheduling, the goal is to ensure all demand is met against minimal costs. We identify 6 different costs that are relevant to the research; changeover costs, overtime costs, external transport, storage and handling costs and the cost of capital. Table 6 shows an overview of the costs that are influenced by the schedule.

Table 6: Summary of costs influence by the schedules.

Type of cost	Calculated cost
Changeover costs	CONFIDENTIAL
Overtime	
External transport costs	
External storage costs	
External handling costs	
Cost of capital	

At this moment, the steps of creating the general production plan and detailed schedules are all completed manually in excel by the planners of Trivium Packaging. There is no immediate feedback on scheduling decisions made with regards to cost effectiveness. The scope of this research is to create a proof-of-concept model that integrates planning and scheduling. The model minimizes the costs influenced by scheduling while ensuring demand is met. We model the scheduling steps and formulate solutions for each step. Alternative solutions are developed and compared, so the best solution can be found.

3 Literature Review

In this chapter, we answer the second research question: "What literature is available to support improvement of the planning and scheduling process at Trivium Deventer?". The goal is to get a deeper understanding of planning and scheduling problems but finding literature related to it.

Section 3.1 gives a general introduction into planning and scheduling. In Section 3.2, we generalize the scheduling problem found at Trivium Deventer and introduce the lot sizing and scheduling problem (LSSP). Additionally, we introduce two different ways to model the LSSP. To solve the LSSP, Section 3.3 introduces and discusses several methods found in literature which can be used. In Section 3.4 we talk about the difficulty of including uncertainty in a scheduling model and methods to overcome uncertainty while solving the problem at Trivium Deventer. We finish this chapter with a conclusion in Section 3.5.

3.1 Planning and Scheduling

Planning and scheduling are two different forms of decision-making which take place at different times in the process. Zijm (2000) created a positioning framework, which shows the three managerial categories of planning on the horizontal axis and the hierarchical planning categories on the vertical axis. The managerial categories are as proposed by Anthony (1965): strategic planning, tactical planning and operational control. Zijm (2000) defined the three managerial categories of planning as technological planning, resource capacity planning and material coordination (Zijm, 2000).

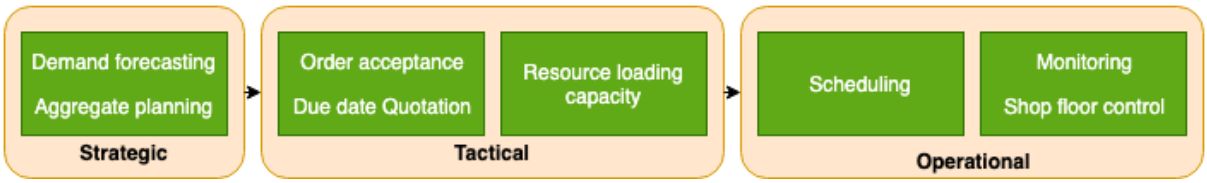


Figure 23: Resource capacity planning category as defined by Anthony (1965).

This research is restricted to the column of resource capacity planning, which focuses on production planning and scheduling. The composition of the resource capacity planning category can be seen in Figure 23. The first part on the strategic level is demand forecasting and aggregate planning. Decisions on this level are capacity planning of the facility, forecasting demand and combining the forecasted demand and capacity restrictions into an aggregate schedule. At the tactical level, job planning, and resource capacity loading must be completed. The decisions are mainly concerned with the effective use of resources. Decisions made on this level are constrained by aggregate planning decisions made on the strategic level and concern the allocation of lines, storage capacity and workforce to a product type. The final level concerns the operational scheduling and shop floor control. At this level, detailed decisions are made with regards to the production schedule. This level is constrained by decisions made on the tactical level, as products are already assigned to a machine and batch size (Zijm, 2000). As mentioned in Section 2.7, we focus on integrating the planning and scheduling and address the scheduling on the operational and tactical level.

3.2 The Lot Sizing and Scheduling Problem

Based on the analysis of the current situation in Chapter 2, we decide to focus on lot sizing and scheduling problems in literature, to help us solve our problem. Lot sizing and scheduling problems (LSSP) are a common type of problem that appear in several forms. The original lot sizing models were

mainly intended to solve the planning problem and did not include a scheduling component. Haase (1994) described lot sizing and scheduling as an activity to obtain simultaneously in which period and in which sequence the products should be produced such that the schedule is feasible and results in minimal costs. Relevant costs are often the holding costs and set-up costs, which should be minimized. Smaller lot sizes increase the number of set-ups required but decrease the holding costs. The same is true in reverse, with bigger lot sizes giving rise to higher holding costs but lower set-up costs. Lot sizing and scheduling models combine the decisions on the tactical level and operational level, which is required to adequately include the sequence dependent set-up times. Section 3.2.1 categorizes the lot sizing and scheduling problem and connects this with the current situation at Trivium Deventer. Section 3.2.2 introduces two general types of LSSP, which can be used to model the problem

3.2.1 Categorizing the lot sizing and scheduling problem

Different lot sizing models can be classified based on several features such as the planning horizon, production levels included, resource constraints, demand type and number of products (Karimi, Fatemi Ghomi, & Wilson, 2003). The planning horizon, levels and number of resources affect the complexity strongly (Gicquel, Minoux, & Dallery, 2008), so we elaborate on them. Within the area of lot sizing and scheduling problems, it is also possible to differentiate between small and large bucket models, which we explain last in this section.

Planning horizon and scheduling period

The planning horizon is the time interval for which the production schedule is made. This time horizon can be finite or infinite. A finite horizon normally is accompanied by dynamic demand, whereas an infinite horizon has stationary demand. While looking at the size of the planning periods, we differentiate between small- and large-bucket models. Small-bucket models are the more traditional models to see for LSSP, while large-bucket models used to be primarily used for lot sizing problems without a scheduling component. Small-bucket models use micro-periods, where one item can be scheduled in every period. Periods are sequenced, which means that when a product is assigned to a period, the products are also sequenced. Large-bucket models use macro-periods, which enable the scheduling of several products within one period. As there are several products placed within one period, there is no immediate sequence. Additional constraints and decision variables must be added to sequence the products.

Levels

Production systems can be single-level or multi-level. A single-level system is relatively simple, where the final product is created directly out of the raw materials. In a multi-level problem, several separate operations are needed to transform the raw materials into the finished product.

The production from metal coil to finished can is multi-level, with several stages for the product to go through. When looking at the 3PC department, the production system is a single-level system where the metal sheets are raw materials which are transformed into finished cans in a continuous operation.

Number of resources

Models are either single-machine or multi-machine. Multiple machines make the problem harder to solve, as we have to determine not only the sequence and lot-size, but also the machine used.

Trivium Deventer is a multi-machine system, as we have several production lines that operate in parallel. The decision we need to make at Trivium Deventer is what line to run, as not all lines can run at the same time. At Trivium Deventer the planning horizon is finite, with a single-level system and

multiple products. There are resource and capacity constraints and the demand is time-varying. Set-ups are sequence dependent, but do not differ between lines.

3.2.2 Models for the lot sizing and scheduling problem

There are many different models of the lot sizing and scheduling problem, with small differences for every problem. Two general models are the capacitated lot-sizing problem (CLSP) and the discrete lot-sizing and scheduling problem (DLSP), which are a small-bucket and large-bucket model respectively (Copil, Wörbelauer, Meyr, & Tempelmeier, 2017). The following basic models of the CLSP and DLSP are given by Fleischmann (1990):

Indices:
i = 1, …, P items
t = 1, …, N periods

Decision Variable:
I_{it}: inventory of item i at the end of period t.
X_{it}: quantity of product i to be produced in period t.
Y_{it}: binary variable indicating if product i is produced in period t.

Parameter:
d_{it}: demand of item i in period t.
c_i: set-up cost of item i
h_i: holding cost of item i per week
$\mathrm{p_i}$: units produced in one period for item i
ss_{it}: safety stock of product i at the end of period t.
$capM_t$: capacity available in period t.

$$(\mathbf{CLSP})\ \min \sum_{i,t} (c_i * Y_{it} + h_i * I_{it})$$

Subject to:

$$I_{i,t-1} + X_{it} - d_{it} = I_{it}\ , \forall i,t$$

$$\sum_i \frac{1}{p_i} * X_{it} \leq capM_t\ , \forall t$$

$$I_{it} \geq ss_{it}\ , \forall i,t$$

$$x_{it} \geq 0\ , \forall i,t$$

$$Y_{it} = \begin{cases} 1, X_{it} > 0 \\ 0, otherwise \end{cases}, \forall i,t$$

$$(\mathbf{DLSP})\ \min \sum_{i,t} (c_i * \max(0, Y_{it} - Y_{it-1}) + h_i * I_{it})$$

Subject to:

$$I_{i,t-1} + p_i * Y_{it} - d_{it} = I_{it}\ , \forall i,t$$

$$\sum_i Y_{it} \leq 1\ , \forall t$$

$$I_{it} \geq ss_{it}\ , \forall i,t$$

$$y_{it} \in \{0,1\}$$

The CLSP in its basic form assigns the production of multiple products to one period. It determines the amount and timing of the production of products in the planning horizon. The result is a production plan per period. Detailed scheduling decisions are not integrated in the CLSP (Gicquel et al., 2008), meaning that the CLSP solves the lot sizing problem, but not the lot sizing and scheduling problem. An approach is to first solve the CLSP and solve the scheduling problem for each period afterwards. The CLSP formulation shows that an item must be set-up in each period that it is produced in. The costs must be sequence independent, as no sequence is determined within the model (Fleischmann, 1990).

The DLSP formulation is similar to the CLSP formulation and uses roughly the same variables. The decision variable X_{it} is replaced by the binary decision variable Y_{it} that only shows if an item is produced within a period, not the amount of production. The objective of the classical DLSP is to determine a production schedule that minimizes the holding and set-up costs. It assumes that the production process always runs full periods, meaning that only one product is produced on a machine in a single period. The model only has a variable (Y_{it}) indicating if a product is produced in a period but does not decide on the amount produced. This amount is a constant in the model. The model given by Fleischmann (1990) assumes an item must be set-up before production, but a set-up can be carried over into the next period. The output of the DLSP is a schedule with the amount and sequence of production in the planning horizon. While the DLSP model does not include sequence-dependent changeovers, it is possible to include these as the sequence is determined within the model (Fleischmann, 1990).

3.3 Solving the LSSP

Section 3.2 introduced the LSSP and two ways to model the problem. In this section we introduce several methods to solve the problem. The LSSP is proven to be an NP-hard problem (Hsu, 1983), making it computationally draining to use an exact method to find a solution (Gicquel, 2008). To solve the LSSP, many other heuristics exist that could be used. Silver (2004) discusses several basic heuristics, which can be applied to a broad range of problems. Within the review, Silver (2004) adopts the following definition: "*The term heuristic means a method which, on the basis of experience or judgement, seems likely to yield a reasonable solution to a problem, but which cannot be guaranteed to produce the mathematically optimal solution*.". A benefit of using heuristics is that they often are able to solve problems that represent the real-world better, because they often require less restrictive assumptions compared to the exact methods. Another reason is that for NP-hard problems, the computational time it takes to find an optimal solution often is too long due to exponential growth. Heuristics aim to find the near-optimal solution, but do not enumerate the solution space, decreasing the computational time (Silver, 2004). This makes a heuristic solution a trade-off between a shorter processing time and a possible non-optimal solution.

3.3.1 Heuristics

Several basic types of heuristics as defined by Silver (2004) are constructive methods, local improvement and problem decomposition. In problem decomposition, a complex problem is split up in several smaller and easier to solve problems. The sub-problems can then be solved independently or sequentially, after which the results are merged into one answer to the complex problem. Constructive methods construct a solution step by step and often a solution is only found when the procedure is done. The greedy method is a specific type of constructive heuristic, where each step the next building block is chosen that gives the best immediate improvement of the objective function. While easy to use, the greedy method can create a bad solution, as initial good choices do not ensure a good final solution. The final basic heuristic is local improvement methods, which requires an initial solution to start. Possible neighbor solutions of the initial solution are evaluated, in the hope of finding a better one. This process is iterated till the end of the method. For this method, a neighborhood must be defined suitable for the problem. In lot sizing and sequencing for example, a neighborhood may be given by exchanging the production of two item, thus changing the sequence. Downside of a local search method is that it can get stuck in a local optimum, without the ability to move to the global optimum (Silver, 2004).

Adaptive search

An example of a constructive heuristic is the adaptive search algorithm (AS) , first proposed by Kolisch and Drexl (1996) and is a combination of a priority rule heuristic and a random search heuristic. In the case of Trivium, the priority rule could be of the form "Earliest Due Date First". This assigns a priority to each job, but instead of directly using these priorities, they are used to calculate a probability. The probability is calculated with Function 3.1.

$$\boldsymbol{Probability_j} = \frac{(\boldsymbol{r_j}+\mathbf{1})^{\boldsymbol{\alpha}}}{\sum_{\boldsymbol{i}}(\boldsymbol{r_i}+\mathbf{1})^{\boldsymbol{\alpha}}} \quad \textit{(3.1)}$$

The r_i is the regret factor of job i, given by the absolute difference between the worst of all priorities and the priority of job i. A higher regret factor means more regret when the job is not scheduled. The alpha is a bias parameter, that affects the randomness of the selection (Kolisch & Drexl, 1996). After a job is selected with help of the probabilities, the job is scheduled and removed from the set of jobs to schedule. New priorities, regret factors and probabilities are determined for the remaining unscheduled jobs. This continues until all jobs are scheduled, after which one iteration of the adaptive search algorithm is finished. Multiple iterations are completed to ensure several solutions are found and evaluated. Due to the randomness of the selection, a new solution is created every iteration.

3.3.2 Metaheuristics

Based on heuristics, a class of methods called metaheuristics have been created. Osman and Kelly (1996) define metaheuristics as follows: "*A meta-heuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions.*".

The subordinate heuristic mentioned in the definition could be a constructive method or local search method. For standard problems, basic heuristics seem to outperform metaheuristics, but for more difficult problems, for example sequence dependent problems, metaheuristics provide good results (Jans & Degraeve, 2007). Fazel Zarandi, Sadat Asl, Sotudian, and Castillo (2018) discuss several metaheuristics that are used in intelligent scheduling, such as simulated annealing, tabu search, variable neighborhood search and genetic algorithms.

Simulated annealing

Simulated annealing (SA) was first introduced as a way to solve combinatorial optimization problems by Kirkpatrick (1984). It uses a local search heuristic as part of the procedure. The algorithm is initiated with an initial feasible solution, of which the total cost is determined. By using shift rules, a neighboring solution is created by modifying the current solution. If the objective value of the neighbor solution is better than or equal to the current solution, the neighbor becomes the new current solution. If the objective value of the neighbor is worse, the neighbor might still be accepted as the new solution, based upon an acceptance probability. This acceptance probability depends on a non-increasing temperature parameter and on the difference between the current objective value and the neighbor objective value. The ability to move to a worse solution means that the SA algorithm can escape from a local optimum (Bertsimas & Tsitsiklis, 1993). Strengths of SA are the ability to move out of local optima, flexibility and the ability to deal with many constraints. Some weaknesses are the difficulty in defining the algorithm parameters, inability to tell if the found solution is the global optimum and an obvious trade-off between computation time and quality of the solution (Fazel Zarandi et al., 2018).

Tabu search

Just like SA, tabu search (TS) is a metaheuristic based on a local search heuristic. TS also has the ability to move out of local optima by accepting a worse solution, but contrary to SA it searches the full neighborhood, instead of only creating one neighbor solution. It then selects the best solution found in the neighborhood and uses this as the base solution for creating a new neighborhood. TS keeps track of the solutions it previously visited in a tabu-list, to avoid getting stuck in a loop by constantly moving to the same solutions (Carvalho, Sofia Pereira, Constantino, & Pedroso, 2001). Same as SA, a strength of TS is the ability to move out of local optima. Additionally, for large and difficult problems, it often finds solutions that are better than previously found solutions by other approaches. Weaknesses are the many parameters that must be determined, no proof of convergence for TS and the computational cost of checking the full neighborhood to find a better non-tabu solution (Fazel Zarandi et al., 2018).

Variable neighborhood search

Variable neighborhood search (VNS) is a metaheuristic first proposed by Mladenović and Hansen (1997) During the optimization process, it changes neighborhood structures to find an optimal solution. This is based on the principle that a local optimum in one neighborhood might not be a local optimum in another neighborhood. On the other hand, a global optimum will be a local optimum in every neighborhood. By using multiple neighborhoods, it is able to escape from a local optimum in one neighborhood, by using another neighborhood structure. If no new solution can be found within a neighborhood, a random solution is created from a broader neighborhood. The procedure stops when the maximum neighborhood size is reached (Hansen, Mladenović, Todosijević, & Hanafi, 2017).

Genetic algorithms

A new class of algorithms often used are Genetic Algorithms (GA). A GA attempts to simulate natural evolution, where each individual (or solution) attempts to find adaptions that are beneficial for the individual. During evolution, these adaptions are encoded in chromosomes of each individual. For GA a group of solutions is used to start the algorithm, the starting generation. To evolve the solutions, two distinct solutions are combined by crossover. This is done several times to create multiple solutions for the next generation. Strengths of GA are the random mutation, ensuring a wide range of solutions, simple implementation and a wide search space. Some of the weaknesses are the difficulty of encoding the problem in a chromosome, a high computation cost and difficulty to determine the best parameters (Fazel Zarandi et al., 2018).

3.4 Uncertainty

In Chapter 2 we identified several stochastic factors that cause uncertainty. Most planning and scheduling methods assume that no disruptions happen, but reality is different. Employees may become sick, reducing the workforce or lines can break down, suddenly decreasing the production. Additionally, when using forecasted demand, forecasts may change making production unnecessary, which causes increases in stock if not countered on time. These sudden changes are all sources of uncertainty in the production and scheduling. However, not much attention has been paid to the role of uncertainty in lot-sizing and scheduling (Alem, Curcio, Amorim, & Almada-Lobo, 2018).

Herroelen and Leus (2005) identify five approaches to integrating uncertainty in scheduling: reactive scheduling, stochastic scheduling, scheduling under fuzziness, proactive scheduling and sensitivity analysis. Reactive scheduling is what is currently used at Trivium Deventer. This form of scheduling creates a predictive schedule (or baseline schedule) and adapts it when sudden changes occur. Uncertainty is not considered in the predictive schedule, which means that the planners will always

need to keep track of any sudden changes. Stochastic scheduling incorporates probabilistic distributions in the model to approach the effect of uncertainty. Sel, Bilgen, and Bloemhof-Ruwaard (2017) use this method to incorporate the shelf-life uncertainty of dairy products in scheduling. The problem with using this method at Trivium Deventer is that the probabilistic distribution of the processing times, breakdowns and manning unavailability are not known, which are required for stochastic scheduling. The third option is scheduling under fuzziness, which is advocated by those that argue that probability distributions for activities are often unknown due to lack of data. Fuzzy numbers depend on the experience of human experts, who estimate the expected job duration, availability of resources and breakdown per job/per week. This takes of lot of time, as the planners need to assign to each forecasted demand, each line, each week, their estimates of a positive and negative outcome. The last option is proactive robust scheduling, which aims to create a robust schedule for certain types of uncertainty. According to Davari and Demeulemeester (2019), this method is traditionally combined with reactive scheduling in a two-stage approach. At Trivium Deventer, proactive robust scheduling is done by using the OEE to determine the line capacity. Although this is not on purpose, the OEE estimates lower line capacities as it also includes the changeovers and planned daily maintenances. Another option would be to increase the scheduled lot-sizes based on the average expected demand and variability of the demand.

The last approach as defined by Herroelen and Leus (2005) is sensitivity analysis. Sensitivity analysis does not change the schedule itself but is used to determine how a schedule changes if a parameter changes. This is an interesting method, which can be used at Trivium Deventer to determine the robustness of the schedule. Ideally, we do not want to see big changes in a schedule if a parameter changes slightly during the week.

3.5 Conclusion

In this chapter, we created a theoretical framework relevant for our study based on existing literature. First, literature was used to get a deeper understanding of planning and scheduling problems in general (see Section 3.1). Second, we needed to find a way to generalize the current scheduling problem. In this research, the lot sizing and scheduling problem (LSSP) is used as a generalization. Section 3.3 discussed several theories on how to solve this LSSP. Finally, we discussed how literature deals with uncertainty in optimization of LSSP problems in Section 3.4

Recalling the framework by Zijm (2000), this research focusses on the scheduling on an operational level and the aggregate planning on the tactical level. The goal is to integrate the planning and scheduling to find near-optimal schedules that can be used in practice.

To be able to solve the problem, it is first categorized as a lot sizing and scheduling with sequence dependent set-up costs and times. There are multiple lines to consider, which are constrained in usage by secondary resources. There is only one production level and the planning horizon is finite, with a duration of 12 weeks. Two general models to model the problem are the discrete lot sizing problem (DLSP) and the capacitated lot sizing problem (CLSP). The DLSP uses small buckets and combines planning and scheduling. The CLSP is a large bucket problem, where scheduling can be added by using a sequencing variable within each large bucket.

To solve the problem, Section 3.3 introduced several algorithms that are usable for solving an LSSP. TS is in general capable of generating the best solutions but requires most computational time as all neighbors are checked. GA similarly presents good solutions, but again has high computational cost. Additionally, the initial implementation is considered easy, but fine-tuning of the parameters is difficult due to the number of parameters and the black-box effect. SA creates solutions of lower

quality, but this is coupled with a lower computational cost. Last there is AS, which has the lowest computational cost in general. With its easy implementation and adaptability, AS is the quickest and easiest to use, but tends to generate the lowest quality solutions.

Metaheuristics based on local search methods, such as SA, TS and GA are often used as well as metaheuristics that use an exact algorithm. There seems to be no single heuristic that is singularly preferred over another, the choice of heuristic is mostly dependent on the requirements of the users. GA and TS often create good solutions but have a higher computation time compared to SA and AS for example. SA in turn often creates better solutions than AS but takes longer to complete. Benefit of SA and TS is that they are easy to adapt during implementation, which can be more challenging with GA due to the encoding.

Last, Section 3.4 covered the existence of uncertainty in scheduling. General methods to counter uncertainty are suggested. We identified five different methods of dealing with uncertainty: namely reactive scheduling, stochastic scheduling, scheduling under fuzziness, proactive scheduling and sensitivity analysis.

In summary, we can classify the problem at Trivium Deventer as a LSSP. For modelling the problem, we decide to use the CLSP with the large buckets and a sequencing variable. Because of the macro periods and sequencing variable, the model spans a greater amount of time while having fewer periods. This should reduce computational time while using the model. The downside of using large buckets is that some details regarding sequencing are lost. In the next chapter a model is defined and a method is selected that we use to solve our problem.

4 SOLUTION METHODS

In Chapter 3 we described literature concerning the problem at Trivium Deventer. To gain more knowledge on the quality of the production schedules and possible improvements, we create a scheduling algorithm based on information found in literature. To reach this objective, we answer research question 3 in this chapter. Research question 3 is as follows: "Q3. How can the scheduling problem at Trivium be solved? ".

In this chapter, we present the process of model development and problem solving. Based on the analysis of the current scheduling process and the literature study, we want to identify improvements that contribute to the research goal. Section 4.1 starts with a formulation of the scheduling problem that we intend solve. Then, Section 4.2 describes the model and Section 4.3 describes the solution methods. In Section 4.4 we explain how we deal with uncertainty. Finally, Section 4.5 provides the conclusions.

4.1 PROBLEM TO SOLVE

Recalling Chapter 2, we describe the problem as follows. The production department consists of 6 parallel production lines, that each can produce a specific set of items. Demand is forecasted, which results in production that must be completed. Most items can only be produced on a single machine, but a few items can be produced on multiple machines. Machines need to be set-up before production on an item can begin. The duration and cost of set-up is dependent on the sequence of the items. The lines operate in shifts, every shift only one unique item can be produced on a line. Multiple production lines can run at the same time but are constrained by the number of workers available.

When scheduling, the aim is to minimize the total costs incurred from the schedule. Minimizing costs is only accepted if all orders are fulfilled. It is not allowed to schedule backorders. We consider storage and set-up costs. Storage costs are dependent on the location at which a pallet is stored, which means we must consider where each pallet is stored. The storage costs include the transport costs of moving the pallet to storage and the cost of storage space used by the pallet. The storage space is shared between all production lines. Additionally, we consider the cost incurred because money is tied up in stock, instead of used for investments or debts. Set-up costs depend on the sequencing of items, as switching from one item to another requires the production line to be changed.

When solving the problem, the problem is decomposed in two subproblems. Subproblem one is the determination of capacities for each production line. Subproblem two is the scheduling of items on each production line, while ensuring inventory and production capacity constraints are not violated. By decomposing the problem, we aim create two subproblems that are easier and quicker to solve individually than combined.

Constraints and assumptions

The following constraints must be met, to ensure a feasible schedule is created:

- All demand is fulfilled on time and before the end of the planning horizon.
- At most one item can be produced on a line during a shift.
- A line must be set-up for the specific item, before production can take place.
- Inventory levels may not increase beyond specified capacity limits.
- No more shifts can be scheduled than available in the department and for the machine.
- Overtime can be scheduled, but must be specified beforehand in the parameters.

- The sum of all determined individual capacities of each line cannot be higher than the capacity of the department.

Besides these constraints, we also make the following assumptions.

- All products are stored and moved on pallets. Pallets are considered uniform in size and weight. To account for the difference in tall packs and short packs, we half the number of cans that go on a pallet. A tall pack containing 8000 cans is treated as a short pack of 4000 pallets.
- The warehouse capacity in Deventer and Van Opijnen is discrete and known, where every pallet takes up one space of storage.
- Automated guided vehicles and forklifts do not constrain throughput, there is sufficient buffer at the end of the line to accommodate waiting pallets for a short while. Also, the palletizers do not constrain throughput.
- Pallets are always transported in a full truck, making the cost of transporting one pallet constant.
- Demand is forecasted and known. All demand is fulfilled with full pallets, which means that a customer might receive more cans than forecasted. This is not considered a loss, as the customer pays per pallet.
- Within one warehouse, the transport and storage costs are constant and do not differ between pallets.
- Processing speeds are known for every product on every possible line.
- Inventory levels are measured at the end of the week and costs are only incurred on pallets in storage at the end of the week. During the week, there is enough capacity to store all short-term and long-term pallets.
- Cost of capital is only incurred when a pallet is in storage at the end of the week.

4.2 Mathematical model

Based on the literature review in Section 3.2 and the problem we introduce in Section 4.1, we create a mixed integer programming (MIP) model for planning and scheduling problem. The MIP model serves as validation for the heuristics that we design. This model can create a production schedule for each of the production lines, which includes sequence of production and the production size.

In this section we provide a detailed explanation of the mathematical model and the solver we use to solve our model.

4.2.1 Model description and choices

The output of a model is a feasible schedule, which shows the planners which decisions should be made during a certain planning and scheduling horizon. The decisions are with regards to several options: production sequencing, batch sizing and inventory levels. Each of these options causes a different cost, therefore depending on the decisions made the total cost of the schedule increases. The model's objective is to minimize the total cost. The objective is restricted by several constraints, such as meeting demand of each item and not exceeding the storage and production capacity constraints of the production department.

This model is based on a mathematical formulation created by Almada-Lobo, Klabjan, Carravilla, and Oliveira (2007). It is a capacitated lot sizing problem (CLSP) with sequence dependent set-ups. Lots are sized and sequenced, but not assigned to a specific time slot. Instead of time slots, the model specifies the sequence of products with a new sequencing variable. The model also includes location-dependent transport and storage costs and specifies where items are stored every week.

To decrease the size of the model and ensure it can be solved within reasonable time, we split up the problem and solve the model for each production line separately. To facilitate this, we assume each production line has its own production capacity and its own storage space. At Trivium Packaging, the production lines all use the same storage space in the warehouses. Similarly, production capacity is limited by the number of personnel available, which is also shared between production lines. In the model, we therefore include storage capacity and production capacity parameters, which we determine beforehand for each production line.

4.2.2 The model

Within our model, we define the following indices, parameters and decision variables.

Indices

t: denotes a week number. The total number of weeks is given by T.

i, j: Denotes an item number, which will be assigned to each product. This is used to keep track of production, demand, storage and costs. The total number of products is given by P.

ℓ: Denotes a warehouse location, in our case the internal or external warehouse. Specific costs are associated with a specific warehouse.

Parameters

cp_i	Cost price of product i
u_i	Units of a pallet of product i
h_ℓ	Holding cost of a pallet in location ℓ per week
$trans_\ell$	Cost of transporting a pallet to location ℓ
coc	Cost of capital for a pallet per week
d_{it}	Demand in pallets for product i at the end of period t
$capM_t$	Machine capacity in period t in number of shifts
$capW_\ell$	Storage capacity at location ℓ in pallets
p_i	Pallet produced per shift of product i
s_{ij}	Set-up time in shifts from product i to product j
c_{ij}	Set-up cost from product i to product j
$InitInv_{i\ell}$	Number of pallets of product i in stock initially in location ℓ
P	Total number of products
T	Total number of weeks
M	Big number used by constraints 5, 6 and 9

Decision variables

X_{it}	Number of shifts product i is produced in week t.
$I_{it\ell}$	Number of pallets of product i in stock at the end of week t in location ℓ
T_{ijt}	Binary variable that denotes if a set-up from product i to product j occurs in week t.
α_{it}	Binary variable that denotes if a set-up of product i is carried into week t.
α_{final}	Item set-up after the last week of production
V_{it}	Variable that denotes the order that product i is produced in week t.
$m_{it\ell}$	Number of pallets of product i moved to location ℓ in week t.
$sd_{it\ell}$	Demand in pallets of product i satisfied from location ℓ in week t.
K_t	Binary decision variable for inventory constraints.

Objective Function

The objective function minimizes the costs that depend on the schedule, which we identified in Section 2.5. As explained previously, to find a good solution to the problem, we minimize the total costs while balancing the number of changeovers and the inventory level. This aim is reflected by the objective function. The objective function consists of four components, the set-up costs, the holding costs, the transport costs and the cost of capital.

$$Setup\ cost = \sum_i \sum_j \sum_t c_{ij} * T_{ijt}$$

The set-up cost is calculated by multiplying the cost of each set-up with the number of set-ups. As the cost of set-up is sequence-dependent, we check for each individual combination. Depending on the products, the set-up costs are zero, small or big, as explained in Section 2.5. This component indirectly minimizes the number of set-ups, as costs are assigned to each set-up.

$$Holding\ cost = \sum_i \sum_t \sum_\ell h_\ell * I_{it\ell}$$

The holding cost is calculated by multiplying the end-of-week inventory at one location with the holding cost at that location. The holding cost only considers the end-of-week inventory and disregards any inventory fluctuations within the week.

$$Transport\ cost = \sum_i \sum_t \sum_\ell m_{it\ell} * trans_\ell$$

The third component is the transport cost. We consider the transport cost from the production line to the storage warehouse, where the cost only depends on the warehouse and not on the specific location where the item is stored at the warehouse. Pallets can also move from production to the customer immediately, in which case the pallet is transported through the internal warehouse. In this case, the pallet incurs the cost of the internal warehouse. As we showed in Section 2.5 that there was no immediate connection between the inventory costs and warehousing actions, we do not include movements within one warehouse in the transport costs.

$$Capital\ cost = \sum_i \sum_t \sum_\ell coc * I_{it\ell} * u_i * cp_i$$

The final component calculates the cost of storing inventory by using the cost of capital. As the inventory is stored in pallets, we calculate the cost price of a pallet and multiply this with the cost of capital. The same as with the holding costs, we only calculate the costs of inventory at the end of the week. If a pallet is stored from Monday to Friday, no cost of capital is incurred. This component ensures the inventory levels are minimized on average and goes hand in hand with the holding cost.

The final objective function is as follows:

$$\sum_i \sum_j \sum_t c_{ij} * T_{ijt} + \sum_i \sum_t \sum_\ell (coc * I_{it\ell} * u_i * cp_i + m_{it\ell} * trans_\ell + h_\ell * I_{it\ell})$$

As each component minimizes another part of the schedule, this objective function can result in high set-up costs, but have low costs for the other components. The goal is total minimization.

Constraints

Constraints 1-4 are all required for the inventory control in the model. Constraint 1 ensures inventory is balanced between the weeks t and t-1, where the initial inventory in week 1 is given. The end inventory in any week is not allowed to be negative, which ensures that all demand will be fulfilled due to this constraint. Constraint 1B is the same as Constraint 1, except that is only ensures the balance between the initial inventory and the inventory at the end of week 1. Constraint 2 limits the inventory at each of the available storage location to the set capacity. Constraint 3 ensures that all pallets that are produced are also moved to one of the storage locations, while constraint 4 ensures

that all demanded pallets are satisfied from one of the available locations. These constraints are required so that the model can track the inventory levels at each location and the movement to each location, which is required for the objective function.

$$I_{it\ell} = I_{i(t-1)\ell} + m_{it\ell} - sd_{it\ell} \qquad \forall i, t > 1, \ell \qquad (1)$$

$$I_{i,1,\ell} = InitInv_{i\ell} + m_{i,1,\ell} - sd_{i,1,\ell} \qquad \forall\, i, \ell \qquad (1B)$$

$$\sum_i I_{it\ell} \leq CapW_\ell \qquad \forall t, \ell \qquad (2)$$

$$\sum_\ell m_{it\ell} = X_{it} * p_i \qquad \forall i, t \qquad (3)$$

$$\sum_\ell sd_{it\ell} = d_{it} \qquad \forall\, i, t \qquad (4)$$

Two additional constraints are required to ensure the model creates correct schedules. Initially, we want inventory to be stored in the primary storage location, unless this location is full. As a result, the model may incur higher costs on the short term, while reducing the long-term costs. The current warehousing department does not look at future demand or expected inventory levels. The internal warehouse is filled first, after which pallets are moved to Van Opijnen. To include this decision process in our model, we use the following set of constraints.

$$capW_\ell - \sum_i I_{it\ell} \leq M * (1 - K_t) \qquad for\ \ell = Deventer, \forall t \qquad (5)$$

$$\sum_i m_{it\ell} \leq M * K_t \qquad for\ \ell = Opijnen, \forall t \qquad (6)$$

Constraints (5) and (6) ensure that pallets are only moved to Van Opijnen when the Deventer storage location is full. K_t is a binary decision variable that can take the value of 1 or 0 each week. Depending on the value that K_t takes in a week, the constraints become as follows:

- $K_t = 1$:
  $capW_{dev} - \sum_i I_{it,dev} \leq 0$ -> ; $capW_{dev} \leq \sum_i I_{it,dev}$ $\forall t$
  $\sum_i m_{it,op} \leq M$ $\forall t$
  When K_t is 1, the capacity in the internal warehouse must be lower than or equal to the inventory level in Deventer. As the inventory can never be higher than the capacity, this means the inventory in Deventer is full. At the same time, the number of pallets moved to Van Opijnen can take any value lower than or equal to M, which is a arbitrarily big number. Consequently, when $K_t = 1$ pallets can be moved to Van Opijnen.
- $K_t = 0$:
  $capW_{dev} - \sum_i I_{it,dev} \leq M$ $\forall t$
  $\sum_i m_{it,op} \leq 0$ $\forall t$
  When K_t is 0, the capacity in the internal warehouse must be lower than or equal to the inventory level plus an arbitrary big number. With M greater than the warehouse capacity, it implies that the inventory levels in Deventer may take any value. At the same time, we see that the number of pallets moved to Van Opijnen must be smaller than or equal to zero. As it is limited to being positive, the number of pallets moved to Van Opijnen must be zero.

It is still possible that pallets are stored at Van Opijnen when there is space in Deventer, when these pallets where previously stored there. Only moving pallets to Van Opijnen is not possible when there is leftover storage space. Limiting the pallet movement may increase the cost of the best solution found, as sometimes it is beneficial to move pallets to Van Opijnen when Deventer is not full. We include these constraints, as this is the decision strategy currently in use and the strategy we will use within our heuristics.

$$\sum_i X_{it} + \sum_i \sum_j s_{ij} * T_{ijt} \leq CapM_t \quad (7)$$

Constraint 7 is needed to restrict the capacity usage of the production line. It calculates the total usage of shifts by summing the production and set-ups that require time.

$$\sum_i \alpha_{it} = 1 \quad \forall t \quad (8)$$

$$X_{it} \leq M * \left(\sum_j T_{ijt} + \alpha_{it}\right) \quad \forall i,t \quad (9)$$

$$\alpha_{it} + \sum_j T_{jit} = \alpha_{i(t+1)} + \sum_j T_{ijt} \quad \forall i, t < T \quad (10)$$

$$\alpha_{i,T} + \sum_j T_{ji,T} = \alpha_{final} + \sum_j T_{ij,T} \quad \forall i,\ t = T \quad (10B)$$

$$V_{it} + P * T_{ijt} - (P-1) - P * \alpha_{it} \leq V_{jt} \quad \forall i,j,t \quad (11)$$

Constraints 8-11 all concern the set-ups and sequencing of products. Constraint 8 ensures that the set-up of one item is carried over. A set-up must be carried over, to ensure the weeks stay linked. Constraint 9 says that production may only take place when the item is set-up in week t or the item is carried in from the previous week. The big M parameter ensures that whenever the item is set-up or carried in, any number of shifts can be produced. Constraint 10 balances the set-ups between succeeding weeks. When an item is set-up in a week, it must be carried over or changed into another item. Constraint 10B is the same as Constraint 10, except that is balances the set-ups in the final week. Constraint 11 ensures that products are sequenced, and the correct order of set-ups is completed. The item that is carried in into week t can take any value, with all succeeding items produced in week t getting a higher value assigned.

$$T_{ijt}, \alpha_{it}, K_t \in \{0,1\} \quad \forall i,j,t \quad (12)$$

$$X_{it}, I_{it\ell}, m_{it\ell}, sd_{it\ell}, V_{it} \in \{0, inf\} \quad \forall i,j,\ell,t \quad (13)$$

Constraints 12 and 13 are basic constraints that ensure that the variables are binary respectively positive but continuous.

4.2.3 CPLEX

To solve the problem exactly we employ IBM ILOG CPLEX Optimization Studio (IBM Corp, 2019). The MIP model previously is used as input for CPLEX. We solve the MIP model using CPLEX so we can use the results as a benchmark for our heuristics. As CPLEX is not our main method in this research, we do not extensively test the different settings available to us.

4.3 PROPOSED SOLVING METHODS

In this section we propose multiple algorithms to solve the scheduling problem. As explained in Section 4.1, we decompose the model in two subproblems. Subproblem one uses the department constraints and distributes the available resources, production and storage capacity, to each of the production lines. Subproblem two schedules the production lines while ensuring the previously assigned constraints are met.

To solve subproblem one, we take the production capacity and storage capacity of the whole production department and distribute it between all production lines. We divide the production capacity based on the moving average of the demand for each of the production lines. By using the moving average, the production capacity fluctuates based on the fluctuations of the forecasted demand. The storage capacity is divided based on the average forecasted demand during the scheduling period. The storage capacity of a production line is constant during the scheduling period.

After the storage and production capacity is determined for each production line, we use a heuristic to create a feasible schedule for each of the production lines. Based on the literature review, we decide to use simulated annealing and adaptive search to solve subproblem two, our scheduling problem. We choose simulated annealing because of the good solution strength, combined with a low computational time. Tabu search creates better solutions but also requires more computational time to get to this solution. Due to the big number of neighbors each solution has, it would take a long time to evaluate all of them with tabu search. Table 7 gives a summary of pros and cons for each method. SA and TS rank the same, but we decide to use SA because of earlier experience with implementation, making implementation and further adaptation easier. Additionally, we use AS to compare a heuristic to a metaheuristic. AS has the benefit of being a fast algorithm, which is flexible and easy to implement.

Table 7: Summary of algorithms elaborated on in Section 3.3

Algorithm	Solution strength	Computational time	Adaptability	Implementation
Adaptive search	--	+	++	++
Simulated annealing	+	-	+	+
Tabu search	++	--	+	+
Genetic algorithm	++	--	-	-+

Implementation of the algorithms is done in IntelliJ, a Java IDE. Appendix 2 shows the pseudocode for the algorithms. Both algorithms we create to solve subproblem two follow the basic idea introduced for our MIP model in Section 4.2. Section 4.3.1 explains how we decide to solve subproblem one. In Section 4.3.2 and Section 4.3.3 we explain how to solve subproblem 2, by using SA and AS respectively.

4.3.1 Production and inventory capacity allocation

The first step is to divide the production and inventory capacity of the production department among the six production lines. We distribute them according to the following rules:

- Production capacity: Each production line is assigned the production capacity which is required to meet the forecasted demand for its items. Any leftover production capacity within the department is distributed according to the moving average of the demand. The assigned production capacity for a production line can change per week.
- Inventory capacity: Each production line is assigned storage space according to the average demand of its items throughout the planning period. The storage capacity is constant during the weeks.

The production line capacity is set for each week. The minimum production capacity required to meet actual demand is used, to ensure a feasible schedule can be created while solving the second subproblem. When there is more production capacity in the department than needed to meet demand, the leftover production capacity of the department is divided based on the moving average of the demand. We choose to use the moving average to ensure capacity is balanced between weeks. Due to the possibility of having a high demand in one week versus no demand in the other weeks, using the moving average of demand ensures the production capacity required to meet the high demand is distributed over multiple weeks.

Additionally, inventory space is allocated to each production line. The allocated inventory space is based on the initial inventory and expected demand. Each production line gets at least the space that

is required to store the initial inventory. The leftover inventory space is divided based on the expected demand of all the weeks. We use a constant inventory capacity, as using a fluctuating inventory capacity causes mostly infeasible schedules. Products can be kept in stock for many weeks, especially for the items which are in constant demand, but are produced in batches. For example, decreasing the inventory capacity between weeks caused the solution to be infeasible, as the inventory capacity would be decreased below the starting inventory of the week.

The calculations

The individual line production capacity is based upon the forecasted demand and time required to fulfill this demand. To start, we calculate the number of shifts that are required to be able to meet the forecasted demand.

$$Cumulative\ shifts_t = \sum_{\ell=1}^{L} \sum_{i} ((\sum_{tt=1}^{tt=t} \frac{d_{i,tt}}{p_i}) - I_{i,0,\ell}) \ , \forall t$$

The cumulative shifts required in week t sums up all demand in weeks 1 to T and subtracts the initial inventory. With the cumulative required shifts known, we calculate the average demand spanning four weeks, where later weeks have a lower impact on the moving average. The formula is as follows:

$$Moving\ average_t = \frac{\sum_{k=0}^{3} \sum_{i=1}^{P} (4-k) * \frac{d_{t+k,i}}{p_i}}{10}, \quad \forall t$$

When the moving average requires a week that is beyond our scope of demand, we use the average demand over all weeks instead. Based on the moving average demand and required demand, we determine the line capacity on each line during every week. Initially each line gets assigned the capacity that is required to meet demand. When there is leftover capacity after the first step, this leftover capacity is divided over all production lines. We do this based on the ratio of the moving average of every line versus the total sum of the moving average. Each line gets their ratio of the available number of shifts, unless this will put their capacity above the max capacity. In that case, the max capacity is used, and the leftover shifts get distributed again.

Below an example of the line capacity determination.

Example line capacity determination

There are two production lines which can produce one type of product each. The production lines share resources and together cannot produce more than 8 items per period. Additionally, the max capacity, initial inventory and average demand per line is given below.

Line	Max Production Capacity per line	Initial Inventory	Average Demand
1	6	0	3.17
2	5	2	3

The forecasted demand of the products produced on line 1 and 2 is as follows:

Line\Week	1	2	3	4	5	6
Line 1	3	1	0	7	3	5
Line 2	0	6	3	4	2	3

Below we find the cumulative needed shifts and moving average demand for each week and production line. The cumulative needed shifts for line 2 in week 1 is zero, as there is no demand in week 1 for this line. In week 2 this need becomes 4 shifts, as there is a demand of 6 products, but an initial inventory of 2 products. Subtracting the initial inventory of 2 means that a total of 4 shifts are needed to meet demand, as one product can be created per shifts.

Cumulative shifts	1	2	3	4	5	6
Line 1	3	4	4	11	14	19
Line 2	0	4	7	11	13	16
Moving Average						
Line 1	2.2	2.1	3.2	5.0	3.7	3.9
Line 2	2.8	4.3	3.1	3.1	2.6	3

The final step is to assign capacities to each line. First capacity is assigned to ensure demand can be met. Line 1 receives three capacity units to ensure the cumulative number of shifts is met. The leftover 5 capacity units are divided over line 1 and 2 according to the ratio of their moving average. With the ratio of 2.2/2.8, line 1 receives 2 additional capacity units and line 2 receives 3 additional capacity units. In the table below, the red numbers give the shifts assigned to ensure demand can be met. The green numbers are assigned afterward, by dividing the leftover production capacity according to ratio. Following this method, production capacities are assigned to each line every week.

Assigned Shifts	1	2	3	4	5	6
Line 1	3+2=5	0+2=2	0+4=4	0+5=5	0+5=5	0+5=5
Line 2	0+3=3	1+5=6	0+4=4	0+3=3	0+3=3	0+3=3

4.3.2 Simulated Annealing

The first algorithm we propose is a simulated annealing (SA) algorithm. It follows the same basic idea as the MIP model introduced in Section 4.2. The goal is to create a schedule that minimizes the costs as much as possible, while ensuring all demand is met.

We start the algorithm by creating an initial schedule. To get the initial schedule, we assign all demand a production slot in the week it is required in. Then we start at the last week and move any production that violates the production capacity constraint to an earlier week. We keep doing this, until we are at the first week and the initial schedule is done. In this step, we focus on getting a feasible initial schedule and do not try to optimize the schedule further beyond feasibility. If there is not enough production capacity to create a feasible schedule at all, the algorithm terminates without continuing to simulated annealing.

Once the initial schedule is created, we initiate the SA component of the algorithm. The initial schedule is set as the current schedule and we set the start temperature. Next, the neighbor schedule is created by using a set of neighboring strategies. We use two possible neighborhood ratio sets, a traditional one and a new one. The traditional strategies are the swap and move strategy, while the new set creates neighbors that reduce holding costs, reduce set-up costs or swap production between weeks. For both sets, we also include the strategy of relinking the weeks, which means that the items that are set-up at the end of a week get changed.

With the neighbor schedule created, we calculate the objective value of the current and neighbor schedule. When the schedule is infeasible the objective value is penalized, which we elaborate on later in this section. We need a penalty to ensure that the algorithm does not always stay in the infeasible zone, as infeasible solutions are often cheaper than feasible solutions. The neighbor schedule is accepted based on (i) the difference in the objective values and (ii) the temperature. We repeat the creation of a neighbor schedule several times, to see if we find improvements. When all iterations are done, the temperature is lowered with the cooling rate. After lowering the temperature, we check if the current schedule is feasible, and reset it to the last found feasible schedule if not. This is done to ensure the algorithm does get stuck in an infeasible solution zone. The algorithm terminates when the stop temperature is met.

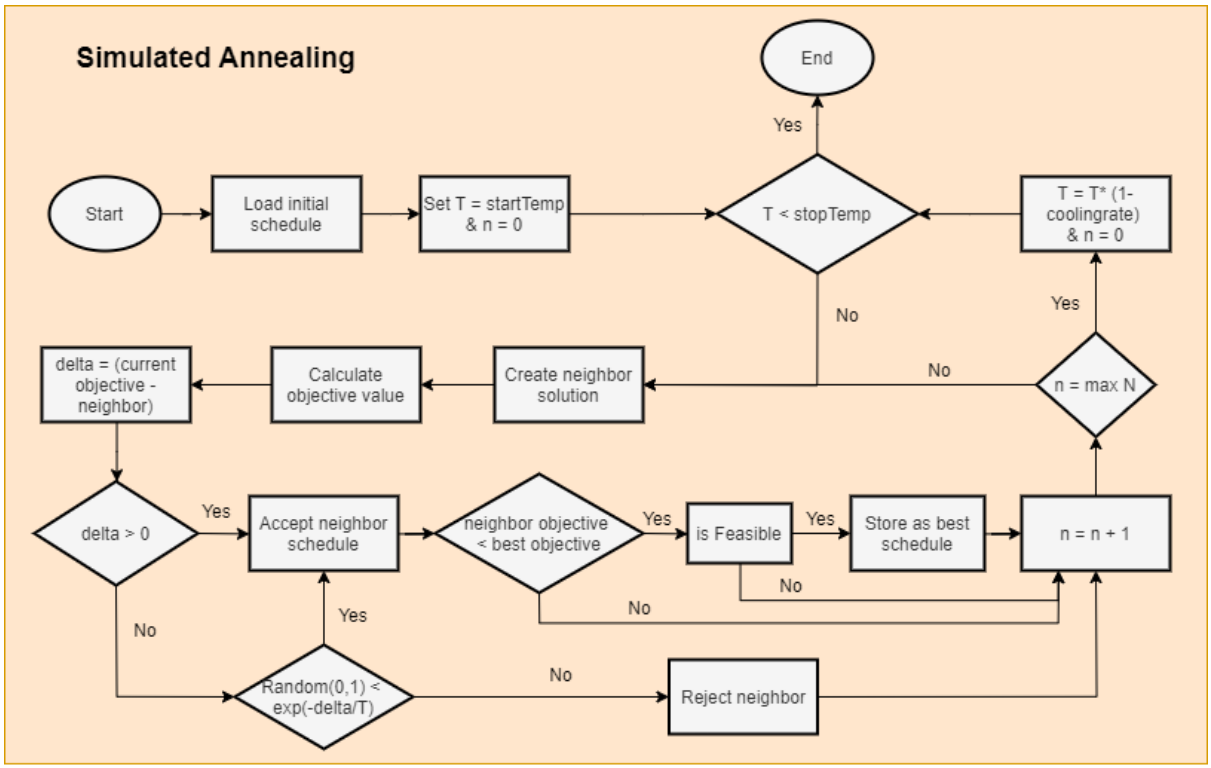


Figure 24: Simulated Annealing heuristic structure

In summary, the algorithm we design consists of the following steps (see Figure 24):

1. Create initial schedule and set as current schedule, set start temperature
2. Create neighboring schedule
3. Calculate score of neighbor schedule
4. Accept neighbor schedule based on difference in score and temperature. Increase n with one.
    a. $n < N$; Return to step 2
    b. $n = N$; Continue to step 5
5. Lower temperature t.
    a. $t > StopTemperature$; Return to step 2
    b. $t <= StopTemperature$; Algorithm done.

Next, we explain each of the different steps in detail. We explain the creation of the initial schedule and SA algorithm in detail. We explore the benefit of including infeasible solutions versus only accepting feasible solutions and explain how we determine the parameters.

Initial schedule

To create an initial schedule, we propose a method where we create an initial schedule in multiple steps, where each step moves the solution towards feasibility. This method is based on the first two steps of the heuristic developed by Almada-Lobo et al. (2007). The research shows good results for using a five-step heuristic to solve a CLSP, where the first two steps are used to create a feasible solution. To create the initial schedule, we complete the following steps:

1. Assign item production to week it is demanded in for all weeks
2. Sequence the production in all weeks
3. Remove production capacity and inventory capacity violations from week.
4. Sequence products

By assigning the item production to the week the item is demanded in, we ensure no late orders are possible in the schedule. We now have a schedule were each product is produced just-in-time, but with possible capacity violations. To determine production capacity violations, we first determine the sequence of production in a week. Production is sequenced with the aim to reduce the set-up costs as much as possible. With the sequence of production known, we calculate the production and inventory capacity violations in each week. If the production capacity constraint is violated, the algorithm moves the production of one or multiple items to an earlier week. While our aim is to have a feasible schedule at the end of the algorithm, any move that we must do to reach feasibility also reduces cost if possible. Production is moved according to the following priority:

1. Moving production removes overtime completely and item is produced in previous week
2. Moving production removes overtime partially and item is produced in previous week
3. Moving production removes overtime completely
4. Moving production removes overtime partially

No more production than needed is moved to the previous week, to reduce the possibility of creating overtime in the previous week. Effectively, less pallets are kept in storage in the preceding weeks. After the capacity constraints are met, the products are sequenced again within the week, to create an initial schedule. As production is moved forward according to a set priority, this method is deterministic and produces the same initial schedule every time.

Neighborhood structure

Within the SA algorithm, we evaluate two different neighborhood creation strategies. In the first option, we use the traditional neighborhood methods of swapping and moving production. When shifting production, we pick an item and move its production or a part of it to another week in the current schedule. When swapping production, we pick two items and swap the production with each other. It is possible for these two items to be the same, so that essentially only the number of shifts a product is produced is swapped, while the items stay in their corresponding weeks. We call this version SA – Classic Strategy, or SA – CS in short.

The second strategy creates neighbor solutions by prioritizing moves that lower the objective value. We categorize the strategies as move, swap and other tactics. Reducing the number of set-ups should reduce the set-up costs, while we attempt to reduce the holding costs by moving production to a later week. With relinking we attempt to optimally link weeks to each other. We call this version the SA – Optimal Neighborhood Structure, or SA – ONS in short.

Table 8 shows the different adaption possibilities for both the SA-CS and SA-ONS.

Table 8: Neighborhood strategies of SA-CS and SA-ONS

SA-CN neighbor tactics	SA – ONS neighbor tactics
Move production	Reduce set-ups (move production while reducing set-up)
	Reduce holding costs (move production to later week)
Swap production	Swap production
Relink production weeks	Relink production weeks

After the creation of a neighbor, its objective value is calculated. If the new objective value is better than or equal to that of the current schedule, the new schedule is accepted. If the objective value is worse, the schedule is accepted with a chance equal to: $e^{\frac{objective_{current} - objective_{neighbor}}{temperature}}$. The smaller the difference between the new and current objective value, the bigger the chance the neighbor solution is accepted.

Cooling schedule

To run the algorithm, we must first determine the cooling schedule, which is determined by the start temperature, the stop temperature, the length of the Markov chain and the cooling rate. The first parameter is the start temperature, which is the starting point of the algorithm. The temperature decreases during the algorithm, until the stop temperature is reached, and the algorithm is done. The temperature is decreased with the cooling rate, a bigger cooling rate means bigger steps are taken and the stop temperature is reached quicker. The length of the Markov chain gives the number of iterations at each temperature step. Together, these parameters determine the duration of the SA algorithm.

We determine the start temperature by looking at the acceptance ratio of the algorithm for several temperatures. Initially we want to have an acceptance ratio close to 0.8, which ensures that roughly 80% of the generated neighbors are accepted. We do not use an acceptance ratio of 1.0, because we start with a solution that is feasible, instead of a completely random solution. Lowering the start temperature reduces the computational time. For the SA-ONS algorithm we start with a temperature of 30000. The SA-CS algorithm starts with a temperature of 40000 to ensure an acceptance ratio of roughly 0.8 for all production lines. Table 9 shows all acceptance ratios for each production line for the given start temperature.

Table 9 Overview of acceptance ratio for each of the production lines with the given start temperature.

Production line	SA – ONS(T = 30000)	SA – CS (T = 40000)
807	0.85	0.88
808	0.83	0.89
810	0.87	0.90
814	0.78	0.90
828	0.89	0.90
846	0.8	0.71

The stop criterium, or stop temperature, is based on the argument that the algorithm can be terminated when the improvement in cost, to be expected when continuing the algorithm, is small (Laarhoven & Aarts, 1987). Because of the acceptance probability function as described before, a lower temperature implies a lower acceptance probability. We choose a stop temperature of 500 for the SA-ONS algorithm and a stop temperature of 50 for the SA – CS algorithm. At each of these stop temperatures, we find an acceptance ratio close to zero.

Last, we determine the length of the Markov chain and the cooling factor. The cooling factor and Markov chain length are related, as the cooling factor determines the difference between each temperature chain and the chain length determines the length of each temperature chain. A bigger cooling factor requires a longer Markov chain length, to ensure the system is stable at the end of each chain (2010). These parameters also strongly affect the duration of the algorithm. The length of the Markov chain is calculated by: $\frac{number\ of\ iterations\ total}{\left(\log_{decreasing\ factor}\frac{end\ temperature}{start\ temperature}\right)}$. The duration of one iteration is on average 1.5 ms for both algorithms. We set the time limit for the algorithm at 6 minutes for all production lines together. This gives us 240000 iterations in total, which we distribute evenly over all production lines. Using the formula above, we find a Markov chain length of 120 for the SA-CS algorithm and a chain length of 203 for the SA-ONS algorithm.

	SA – ONS	SA – CS
Start temperature	30000	40000
Stop temperature	500	50
Cooling rate	0.98	0.98
Chain length	203	120

Figure 25 Overview of simulated annealing parameters.

Improving the algorithm

During the initial tests, we change the SA algorithm at several points. Initially, no feasible solution was considered as a possible solution. This makes it possible for the algorithm to miss a better feasible solution, if it can only be reached by moving through an infeasible solution. We include infeasible solutions as possible solutions but penalize infeasible solutions with a multiplier based on the current temperature. The lower the temperature, the higher the penalty becomes, ensuring that infeasible solutions have less change to be accepted at lower temperatures. Infeasible solutions are penalized according to the following equation:

$$Penalty = \frac{4 * start\ temperature}{current\ temperature}$$

During the first Markov chain the start and current temperature are the same, which means the penalty equals 4.We choose to use a penalty of at least 4 to ensure that at the start of the algorithm, feasible solutions are still preferred over infeasible solutions. Infeasible solutions have a much lower cost compared feasible solutions, which means that the algorithm often gets stuck in an infeasible zone. By penalizing infeasible solutions with at least four, we attempt to balance the cost difference between feasible and infeasible solution.

Additionally, we add a reset point, where the algorithm is reset to the last feasible solution found up until then if it does not have a feasible solution at the end of a Markov chain. By doing this, we ensure the algorithm does not get stuck in the infeasible solution zone.

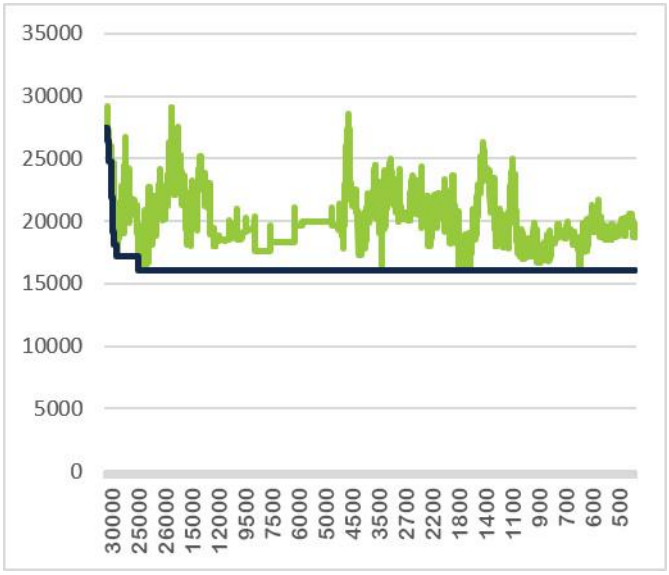


Figure 26: Objective value of production line 810 without allowing infeasible solutions.

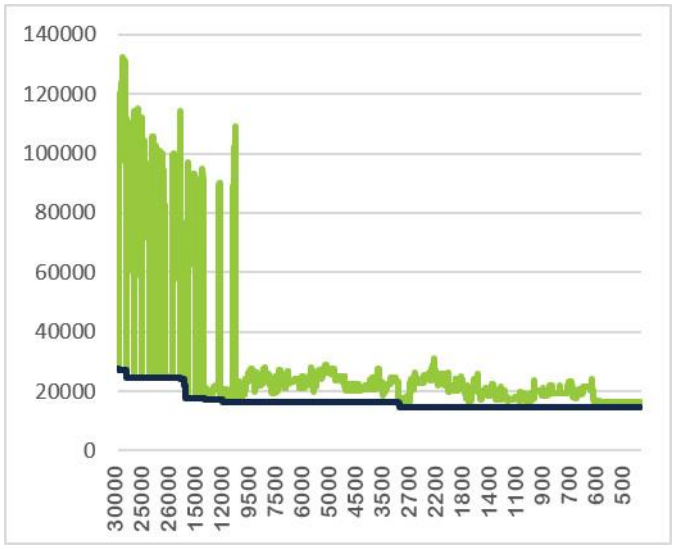


Figure 27: Objective value of production line 810 with infeasible solutions and reset.

Figure 26 and Figure 27 show the development of the objective value when infeasible solutions are not allowed and when infeasible solutions are allowed. The best solution found in the first instance is €16055, while the best solution found in the second instance is €14750. A small improvement is found when allowing infeasible solutions and using a reset.

Table 10: Overview of feasible schedules and improvements found for each of the neighbor strategies in SA-ONS.

	Setups	Relink	Switch	Holding
# of tries	12048	12156	11830	12206
Feasible	7315	7306	5284	11705
Improvements	277	1379	2039	5

Another possible improvement is changing the probability of choosing a certain neighbor strategy. Table 10 shows the number of feasible solutions and improvements found by each neighbor strategy. When looking at the neighbor strategies for the SA-ONS algorithm, the relinking strategy produces feasible schedules most often, but it does not result in an improvement as often as the move and switch strategies. The switch strategy leads to the most improvements, then reducing setups, then relinking and last reducing holding costs. The relinking and reducing holding costs strategies create more often feasible schedules, compared to the move and switch strategies. Based on these findings, we decide to use the probabilities outlined in Table 11.

Table 11: Probability of choosing neighbor strategies for ONS

Strategy	Probability ONS
Reduce Setups	0.3
Reduce holding costs	0.15
Relinking	0.2
Switch	0.35

We use the same improvements for the SA-CS algorithm. Adding the usage of infeasible solutions to the SA-CS algorithm does not create improved solutions. Table 12 shows the attempts and improvements for SA-CS. The move and switch strategies are most effective, with the relinking strategy coming in last. Based on these findings, we decide to use the probabilities as outlined in Table 13. The probability to move or switch production is bigger than the probability to relink weeks, which is in line with the findings of the best neighbor strategy.

Table 12: Overview of feasible schedules and improvements found for each of the neighbor strategies in SA-CS.

	Move	Relink	Switch
# of tries	7706	7634	7620
Feasible	3332	5481	3923
Improvements	60	18	50

Table 13: Probability of choosing neighbor strategies for SA - CS

Strategy	Probability CS
Move	0.4
Relink	0.2
Switch	0.4

4.3.3 Adaptive search

The second algorithm we propose is adaptive search (AS), which is a constructive heuristic. It creates a schedule by sequentially adding a product to the production schedule. Which product is added to the production schedule is based on probabilities decided by using a priority rule. Additionally, all

production is assigned a storage location. The adaptive search algorithm does not necessarily create a feasible schedule and can thus result in not finding a schedule. Next, we explain the algorithm in general. After that, we explain the priority rule that is used and the parameters that we choose to use.

The algorithm

The full algorithm is structured as follows: We begin with an empty schedule and the full set of demanded products. From now on we call the demand of an item in a specific week a job. We then create the initial job set by removing all jobs from the set that can be fulfilled by using the initial inventory levels. We take the initial job set and start scheduling in the first week. The first step is to assign all jobs that have their deadline in the current week to the production schedule. We do this to increase the number of feasible schedules created, as late orders are not allowed. After the jobs of the current week are assigned, we reach the first decision point. Here the decision is made to move to the next week or stay in the current week, which results in the following actions:

- The algorithm stays in the current week: The next decision is which job to add to the schedule next. After the job is added to the schedule, we return to the decision point where we might move on to the next week.
- The algorithm moves to the next week: We return to the first step as described before, where we schedule the jobs with deadlines in the current week first.

The probability to move to the next week is based on the spare capacity within the week. This adds the possibility for schedules where the first weeks are not filled up with production. Lower spare capacity increases the possibility of moving to the next week.

The second probabilistic decision is when we decide what job should be assigned to the schedule next, which follows the standard adaptive search method. Each job i is assigned a priority (ϕ_i), which is based on the number of weeks the deadline of the job is away, the size of the job and if the machine is already set-up for the job. In turn, the priority is used to determine a regret (R_i) and probability (P_i) for each item. Using this probability, we randomly choose the job to assign next. It is possible for one item to have multiple jobs in the upcoming week, as demand can be spread over several weeks. In that case, only the first job of an item is considered for scheduling. Demand must be satisfied in sequence, so producing demand of a later week earlier is not possible, as it will still be used to satisfy the first demand that is found. Considering only the first job of an item then reduces the number of options available.

The constructive part stops when there is no demand left or we are past the final week. We then sequence the production of all jobs within all weeks, without moving them between weeks. If the schedule is feasible, the objective value is calculated. The objective value is calculated in the same way as in the SA algorithm. If the objective value is an improvement, we store the solution as our best solution. The current solution is then cleared, and the next iteration of the algorithm starts. Figure 28 shows an overview of the adaptive search algorithm. As adaptive search is a randomized search method, each iteration generates a different solution. The best solution found during all iterations is the final solution of the adaptive search algorithm.

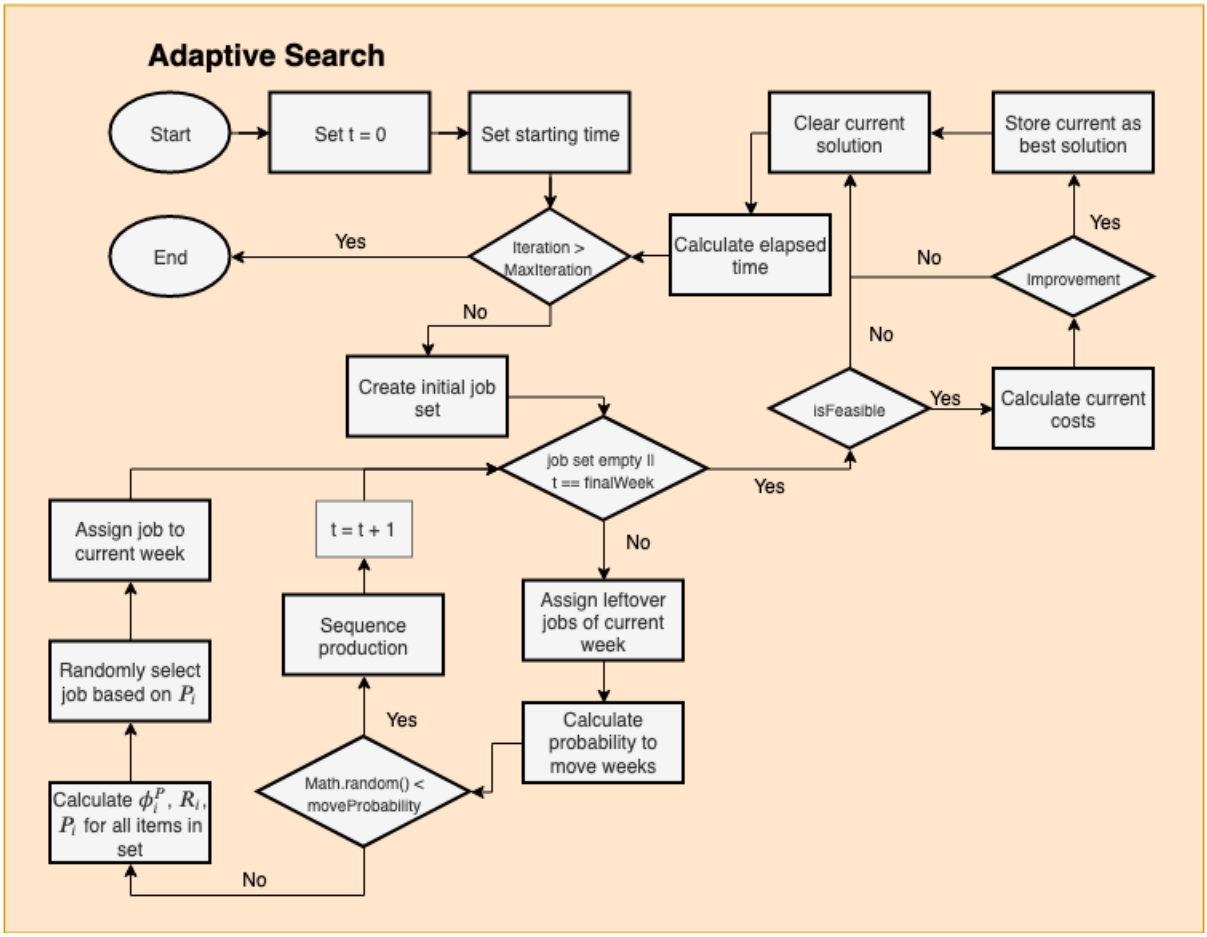


Figure 28 Graphical representation of the adaptive search algorithm

The algorithm terminates when the maximum time is elapsed. Additionally, when it is not possible to assign all leftover jobs with its deadline in the current week to production in that week, the algorithm stops the iteration and restarts with an empty schedule and the initial job set.

Priority rule

As mentioned previously, the priority rule is based on three factors. We look at the deadline of the job, the size of the job and if a similar job is already set-up in the week. Each of these parameters aims to promote optimality or feasibility of the final solution. Using the set-up while determining the regret ensures that jobs that do not require a set-up are chosen preferably, lowering set-up costs in the final solution. By including the deadline, we aim to prioritize jobs that must be completed earlier. By scheduling the closer deadlines first, the time that finished products are kept into storage also decreases. Last, we consider the size of a job, where we balance between feasibility and optimality. We assign a higher priority to bigger jobs, unless these jobs to not fit into the current week. By giving a higher priority to bigger jobs that still fit in the week, we increase the change of creating a feasible schedule.

The priority rule is as follows, where the size of job and deadline of job are combined in the ShiftPriority component:

$$\phi_i = \text{WeightSetup} * \text{IsSetup} + \text{ShiftPriority}$$

The value of IsSetup is dependent on the lowest set-up cost within the week for a certain job. If no set-up is needed, because the item is already set-up or a similar item is set-up, the set-up value is 1.

When only a small set-up is required (see Section 2.2.3), the set-up value is 0.5. When a big set-up is required, the set-up value is 0, as there is no priority to schedule a job that requires a big set-up.

The ShiftPriority combines the deadline of the job and the jobsize into one parameter. Jobs whose demand is further away, receive a lower priority. Similarly, smaller jobs have a lower priority, as these are easier to fit in the schedule. The ShiftPriority is calculated as follows:

$$ShiftPriority = \frac{Jobsize}{(Week_{demanded} - Week_{production}) * WeightWeek}$$

$Week_{demanded}$ is the week that a job must be finished, while $Week_{production}$ is the week the algorithm is currently scheduling production in. The $WeightWeek$ parameter gives us control over the importance of the job size versus the deadline of the job.

Parameters

We run experiments to find the best parameter values for the parameters $WeightWeek$, $WeightSetup$ and α. The alpha is a parameter used in the calculation of the item probability, which we explain in Section 3.3. An alpha of zero makes the adaptive search algorithm random, while an alpha approaching infinity makes the algorithm deterministic.

We determine the parameters in the following order:

1. We determine the values of $WeightWeek$ and $WeightSetup$. To do this we use an arbitrary set of alphas ({250, 150, 50, 1, 0}) and 1000 replications per alpha.
2. We determine the best alpha scheme and number of iterations while using the values of $WeightWeek$ and $WeightSetup$ determined in step 1.

The parameters are determined sequentially, as we find them to be dependent on each other. Changing the parameters for the priority rule affects the best alpha scheme. To solve this problem, we first determine the $WeightWeek$ and $WeightSetup$ with several alphas and multiple iterations. With the $WeightWeek$ and $WeightSetup$ determined, we find the best alpha scheme to use.

Table 14 and Table 15 show the average objective value found by running the algorithm 20 times for production line 810 and 846 respectively. The alpha scheme used is as described above. A feasible solution was found during all 20 executions of the algorithm.

Table 14: Parameter sets {WeightWeek , WeightSetup} and found solutions for production line 810

Parameters	**{1,2}**	**{3,2}**	**{5,2}**	**{7,2}**
Avg. Objective value	28093	30199	31956	32719
Parameters	**{1,4}**	**{3,4}**	**{5,4}**	**{7,4}**
Avg. Objective value	26941	30121	31005	34077

Table 15: Parameter sets {WeightWeek , WeightSetup} and found solutions for production line 846

Parameters	**{1,2}**	**{3,2}**	**{5,2}**	**{7,2}**
Avg. Objective value	68473	50641	53274	57227
Parameters	**{1,4}**	**{3,4}**	**{5,4}**	**{7,4}**
Avg. Objective value	66002	67145	48030	52349

We choose to use the parameter set {3,2}, so that WeightWeek is 3 and WeightSetup is 2. Set {3,2} can find feasible solutions most often among all production lines. For production line 846 it results in

the second highest objective value found, where only set {5,4} scores higher. The results of production line 810 are closer, with {1,4} performing best. The lower parameter sets show better results than the higher parameter sets. Over all lines set {3,2} shows some of the best results combined with the ability to find a solution most often.

With the $WeightWeek$ and $WeightSetup$ determined, we find our final values of alpha. We look for an alpha scheme that gives the best feasible solution, but that is also able to find a feasible solution reliably. Table 16 and Table 17 show the objective value for several combinations of alpha and number of iterations. Each combination is repeated 20 times and the average objective value is used in the tables. Red numbers denote combinations where adaptive search was not able to find any feasible solutions in any of the repetitions. This is most common at the lower alphas of 0 and 1 and at the high alpha of 250. We also tried alphas above 250, but this almost never gave a feasible solution.

Table 16: Performance for different combinations of alpha and sample size for production line 846

alpha\sample	50	200	1000	2500	5000
250	**79703**	76336	70948	68801	68184
150	80290	76293	72193	**68390**	**66850**
50	80143	**73909**	**70764**	68752	67717
1	88500	84281	80654	80367	79807
0	91967	89815	86077	84389	83257

Table 17: Performance for different combinations of alpha and sample size for production line 807

alpha\sample	50	200	1000	2500	5000
250	33799	32589	30400	29160	28758
150	33264	32119	**29625**	29338	**28592**
50	**32680**	**31818**	30373	**28914**	28923
1	0	0	42556	0	41879
0	0	39993	0	0	44537

Overall in the results we find that the higher alphas have better results than the alphas of 0 and 1. What exactly gives the best result is dependent on the production line, which is expected as the different problem sizes for each production line create different sized solution spaces. The smaller problems (807, 814, 828) benefit from the higher alphas, as the priority rule focusses on feasibility. It seems the smaller problems have a relatively small feasible space, which makes it more difficult to find feasible solutions randomly.

Noticeable in the alphas is that while the results improve with higher replications, the best solution changes between alpha 150 and 50. Exception to this rule is production line 814, which has the best results with an alpha of 1. Important to note is that while we found the best objective value using this alpha, AS did have problems finding feasible solutions. Using 1000 iterations, AS finds a feasible solution 8 out of 20 repetitions, while using 5000 iterations resulted in a feasible solution in 15 out of 20 repetitions. Normally, we expect better results for lower alphas with higher replications, but here the best results are found with alphas of 150 and 50. An explanation could be that the algorithm requires the priority rules to find feasible solutions reliable, but that while the priority rules focus on feasibility and improving cost at the same time, they do not necessarily point toward the best solution, to ensure feasibility.

For the final alpha scheme, we decide to use the alpha set of {250, 150, 50, 1} with {200, 1000, 2500, 5000} iterations. This makes for a total of 8700 iterations, where the alpha of 250 is iterated 200 times and the alpha of 1 is iterated 5000 times. We exclude alpha is 0, because often it does not find feasible solutions. Alpha 1 is includes because the randomness is beneficial for some production lines, most notably production line 814. By using multiple alphas, we can find a good result for the different production lines and datasets.

4.4 Uncertainty of the process

The process we deal with has several factors of uncertainty, e.g. machines might break down, work may go slower as expected, the forecast can change, etc. In Section 3.4 we discussed several ways of dealing with uncertainty and identified possible methods to use at Trivium. The method interesting to use at Trivium is proactive robust scheduling. Stochastic and fuzzy scheduling are both difficult to implement at Trivium, as stochastic scheduling requires the probabilistic distributions of production, breakdowns, unavailability and fuzzy numbers are dependent on the judgment of humans. Due to the number of items, it would be too much work for the planners to assign to each demand, production and line their positive and negative view. The model can only create reliable schedules when the input data is correct and reliable. To ensure the model can also create usable schedules when the input data is unreliable, we include a measure of robustness by determining the pallet production per shift based on the OEE of each of the lines.

As the OEE contains the time spent on changeovers and we schedule the changeovers separately in our method, the OEE is lower than it should be if changeovers would be excluded. The result of this is that we might end up using too low production speeds in planning, as we compensate for changeovers twice. We decide to use the OEE including changeovers, as the variability in production and demand is so high, that we find the additional buffer is justified. When using this method, it is important to monitor the OEE constantly to adapt to any changes in production speed and the production process. Additionally, the OEE is based on data collected during production, thus the effectiveness of this method depends on the quality of the data. When the OEE is based on accurate data, basing the production speeds on them is an effective way to make the schedule robust for changes in production.

This robustness mainly focusses on uncertainty in production and not the demand, which is something we will have to keep into account when analyzing the results. To ensure the schedule can handle all changes in demand, we would have to analyze the forecast used at Trivium Deventer, but this is outside the scope of this research. When sudden changes in demand happen, we use reactive scheduling to adapt the schedule to the new circumstances.

4.5 Conclusion

We model the problem as a capacitated lot-sizing and scheduling problem with sequence dependent set-ups and secondary resources. The objective function consists of the holding costs, transport costs, set-up costs and cost of capital costs. Based on this objective function, we can compare multiple schedules and determine the schedule which generates the lowest costs.

The problem is split up into two subproblems. The first subproblem determines the production and storage capacity for each production line based on the forecasted demand and initial values. The second problem solves the planning and scheduling problem for each of the production lines.

We have chosen to implement a simulated annealing algorithm and an adaptive search algorithm for solving the second subproblem. Simulated annealing is a relatively short algorithm compared to tabu

search, that can find good solutions without getting stuck in local optima. Tabu search is also an option but will be computationally demanding due to the high number of neighbors that exist in this problem. Evaluating every single one of them each iteration would cost too much time. We do not use a genetic algorithm, as there are many parameters to configure to get a good solution. Adaptive search chosen as it is fast and easy to implement and adapt. By using a heuristic and meta-heuristic we can compare the ability of two strongly different algorithm.

Within the SA algorithm, we use two neighborhood structures, one based on the random selection of an item to produce and the other based on neighbor that optimize on a certain parameter, e.g. number of changeovers or holding costs. We allow infeasible solutions in the SA algorithm, but penalize them, to decrease the acceptance of infeasible solutions. Additionally, we reset the SA algorithm every temperature iteration to a feasible solution if needed. This is done to avoid getting stuck in the infeasible solution space.

The AS algorithm makes use of adaptive probabilities, which are based on the job size, deadline of the job and if the job or a similar one is already setup. To be able to find good solutions in varying situations, we use an alpha scheme with several alphas and iterations scaled to the alphas.

5 ANALYSIS OF ALGORITHMS

In Chapter 4 we formulated several algorithms to solve the scheduling problem. In this chapter we present the results of our research and answer the fourth research question "Q4. What is the best solution for the scheduling problem at Trivium Deventer?". We consider the SA and AS scheduling algorithm and compare these to the results from CPLEX.

In Section 5.1, we discuss the experimental setup. In Section 5.2 we compare the different heuristics and find the best alternative. Then, in Section 5.3 we compare the algorithm with the schedules created by the planner, to find points of improvement. Finally, in Section 5.4 we conclude the chapter.

5.1 EXPERIMENTAL DESIGN

The first step in our experiments is to determine the best algorithm to use for scheduling. We run each of the algorithms several times for all datasets and compare the results to each other.

Ideally, we want to compare the schedules created by the algorithms to the actual demand and production levels, including the loss of production that happened in reality. During the modelling process, we discovered this is too complex. The production levels are influenced by many factors, which we did not include in our model. This makes comparing the algorithm results to the actual production levels inaccurate. Instead of comparing our algorithms to the actual production, we compare the results of the algorithms to the initial schedule created by the planner. We use the comparison to determine on which points it is possible to improve the current schedule method, if applicable.

Data Sets

The experiments are completed using datasets gathered during the research. We decide to collect datasets instead of creating data, so that we can compare the results with the schedules created by the planners. We use 4 datasets that contain 24 weeks of data each. Every dataset contains 24 weeks of forecasted demand and the initial inventory. Additionally, for each dataset, we have the schedule created by the planner during that week, which covers a maximum of 12 weeks. Table 18 shows the date on which each dataset was gathered. Due to the short time period in which data collection took place, there is overlap in the datasets. We could not prevent this, as datasets were only collected between May and September of 2019. We consider the effect of this overlap in the conclusion.

Table 18: Datasets and date of gathering

Set 1	01-05-2019
Set 2	08-05-2019
Set 3	24-06-2019
Set 4	01-07-2019

Each dataset is used to create a schedule which minimizes the total costs over 24 weeks of production. When comparing the algorithms to each other, we use results found by analyzing 24 weeks of data. We also compare the schedules to the schedule created by the planner. As the planner only schedules up to 12 weeks of demand, for this comparison we only look at the first 12 weeks of the results given by the algorithm. We still create the schedule using 24 weeks of data, to avoid what we call the cooling-down period. When using 12 weeks of data, we see that the expected inventory levels decrease after roughly week 8. As no demand is found after week 12, the model aims to decrease the inventory level

as much as possible, to decrease costs. This is undesirable, as it gives an inaccurate view of the production to be scheduled and the expected inventory levels.

Experiments

Initially, we experiment to determine the best algorithm available. We compare the results of each algorithm to each other and to the results of CPLEX. The CPLEX solver is limited to 8 hours of computation time, to ensure we have enough time to get results.

We complete the experiments for each of the available datasets. As the algorithms contain a degree of randomness, we replicate each experiment 5 times. This enables us to determine the variability of results and check if the results are consistent or differ strongly between runs. We replicate experiments by running the algorithms again with the same datasets and parameters, while ensuring the random number generators are generating different numbers each run.

We also test the heuristics under unconstrained and constrained conditions. With unconstrained conditions, we assume and ensure the inventory space is ample enough to store all pallets. We expect to see no transport costs in this situation and no storage costs, except for those incurred by the cost of capital. As explained in Chapter 2, storing pallets internally incurs no costs except for the cost op capital. With constrained conditions we limit the internal storage space to force the model to use external storage. We do this to see how each heuristic uses external storage and minimizes the total costs in constrained circumstances.

Last, we compare the results from the best heuristic with the schedule created by the planner. For this we use parameters that resemble reality. The inventory space and production capacity are in line with what is available to the planner.

5.2 BEST SOLUTION METHOD

When used in practice, the algorithms are used on a regular basis. Every week, new schedules are created based on new demand data. Because of this, we determine the best algorithm by looking at the results gained from 5 different datasets, which we describe in Section 5.1. Here, we present the results by using one dataset and Appendix 5 shows the results using the other datasets. These initial experiments are done with ample internal inventory space, that no transport costs should be incurred.

Table 19: Results of all production lines for data of 01-05-2019.

	807	808	810	814	828	846	Total
CPLEX	€ 5,922	€ 8,909	€ 9,916	€ 15,639	€ 11,934	€ 25,317	€ 77,637
AS	€ 18,355	€ 23,576	€ 36,655	€ 40,584	€ 34,081	€ 62,781	€ 216,032
SA – ONS	€ 14,649	€ 18,699	€ 14,429	€ 28,548	€ 17,326	€ 37,403	€ 131,054
SA - CS	€ 7,388	€ 11,915	€ 11,842	€ 20,908	€ 14,608	€ 29,524	€ 96,185

Table 19 shows the results for one dataset for all algorithms. The results from dataset 01-05-2019 show that the SA-CS algorithm creates the best schedules in terms of objective cost compared to the SA-ONS and AS algorithm. We find the same results when using the other datasets (see Appendix 5). Averaged over 5 replications, SA-CS performs best for all production lines. SA-ONS has the second-best performance and AS is worst by far.

Regarding the results from CPLEX, the optimal value was only found for production line 807. For the other production lines, the found objective value is the best value found within 8 hours. For production lines 808, 810, 814 and 828 the best objective value was found after approximately 5 minutes, but

CPLEX was not able to determine within 8 hours if the found value is optimal. Thus, for these production lines further improvements might be possible. CPLEX encountered the greatest challenge while solving production line 846, requiring 8 minutes to find a feasible solution of €80,614 for production line 846. CPLEX terminated after 8 hours with the best-found objective value of €25,317.

Table 20: Percentual increase in objective value compared to the CPLEX results

	807	808	810	814	828	846	Total
AS	210%	165%	270%	160%	186%	148%	190%
SA -ONS	147%	110%	46%	83%	45%	48%	80%
SA - CS	25%	34%	19%	34%	22%	17%	25%

Table 20 shows the objective value increase for each production line. For all production lines the lowest increase in objective value compared to the CPLEX solution is given by the SA – CS algorithm. Relatively seen, the difference between the performance levels of each of the algorithms is highly noticeable. Yet, here we must also consider the structure of the cost. One changeover can cost up to €1376, which is almost equal to the cost difference between CPLEX and SA-CS for production line 807.

Aside from the total costs incurred, we consider several other KPIs. The shifts of production are equal between all algorithms, but the total number of changeovers decreases from highest for AS to lowest for CPLEX, see Table 21.

Table 21: Additional KPIs for data 01-05-2019 with ample inventory space.

	CPLEX	AS	SA – ONS	SA - CS
Changeovers	23	71	46	34
Shifts production	771	771	771	771
Inventory value	€1,335,760	€2,025,801	€1,852,097	€1,612,115

For all KPIs, CPLEX has the best values, followed by SA – CS, SA – ONS and AS. While AS has the highest number of changeovers, it does not have the lowest average inventory value. This shows that while there are many runs in AS, they are not linked together optimally to reduce inventory value. SA – CS has the lowest number of changeovers, but the runs are linked together in such a way that nothing more is produced than required.

The majority of the costs is made up of changeover costs, see Figure 29. Comparing the different algorithms, we see that a decrease in changeover costs is the biggest cost driver to reduce the objective value. As we used parameters with ample inventory space, no storage and transport costs were incurred, which is as expected.

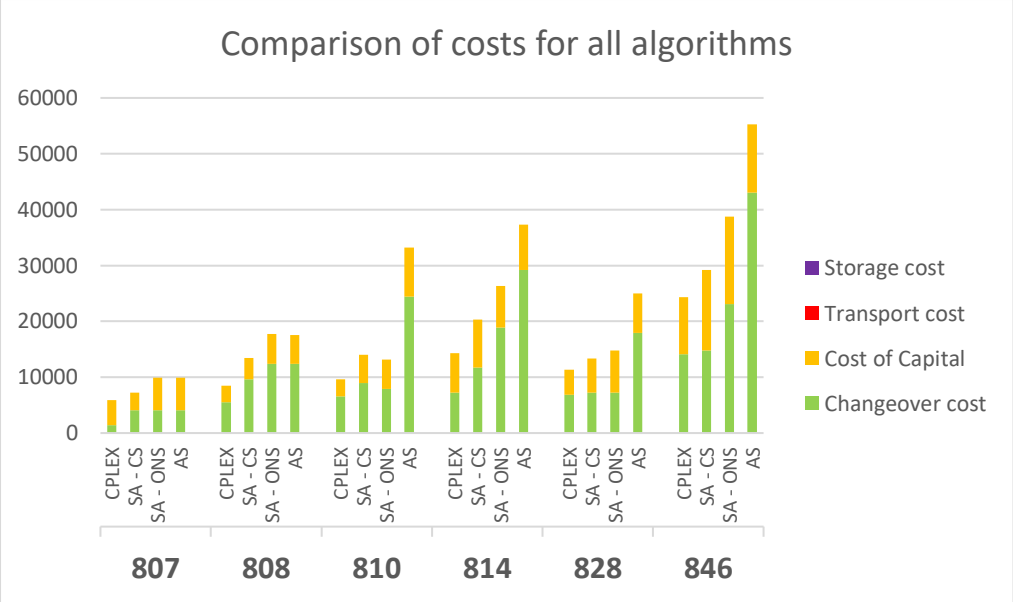


Figure 29: Cost comparison of all algorithms. None of the algorithms had storage or transport costs.

We also used a parameter set that constrained the inventory level, to force the heuristics to move items to external storage. Table 22 shows the KPIs when inventory space is limited. Compared to the results presented in Table 19, the average inventory value is lower but an increase in changeovers is found. The number of changeovers is increased to be able to cope with the limited amount of storage space. AS is not able to find feasible solutions when the inventory space is constrained. This could be due to the adaptive search parameters, which focus on reducing the number of setups. More setups are needed to handle lower inventory space, which could be the reason AS does not find results with constrained inventory space.

Table 22: Results of algorithms with constrained inventory space.

	AS	SA – ONS	SA - CS
Changeovers	No complete results	124	85
Shifts production	No complete results	771	771
Inventory value	No complete results	€1,179,371	€1,105,847

The distribution of the different costs stays roughly the same, but due to the constrained inventory space, holding and transport costs are incurred. The cost of capital is lower compared to the non-constrained dataset, which is as expected due to the lower inventory levels. Despite the lower cost of capital, the total cost is higher due to the added storage and transport costs. Additionally, higher changeover costs are incurred at all production lines, as more changeovers are needed to keep low inventory levels.

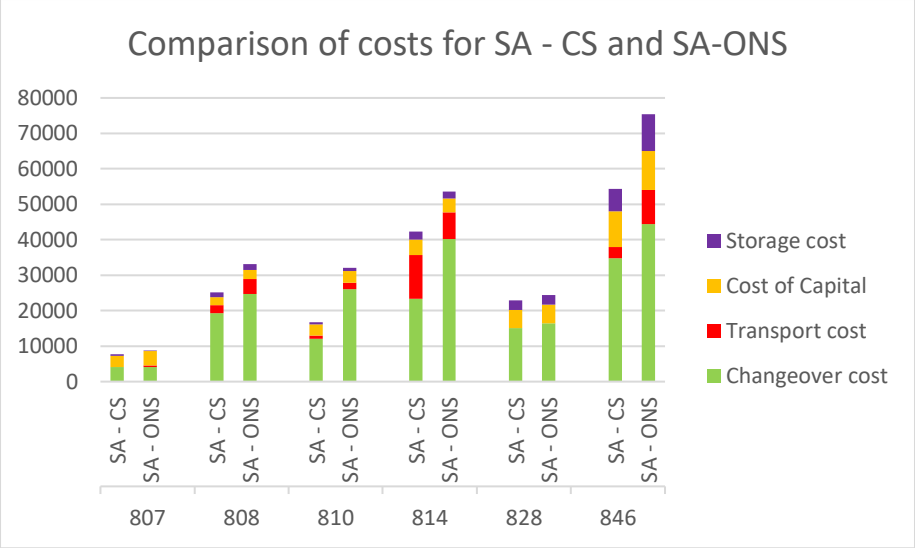


Figure 30: Incurred costs for SA the algorithms with constrained inventory levels.

This decrease in inventory value is supported by the distribution of production throughout the weeks. Figure 31 shows the shifts of production in weeks 1-8 (Q1), weeks 9-16 (Q2) and weeks 17-24 (Q3). With the inventory constrained, the production is distributed much more evenly over the three quarters. By spreading production, it increases the number of changeovers, but reduces the items completed several weeks ahead of time. This in turn reduces the average inventory value and the level of inventory that must be moved to Van Opijnen.

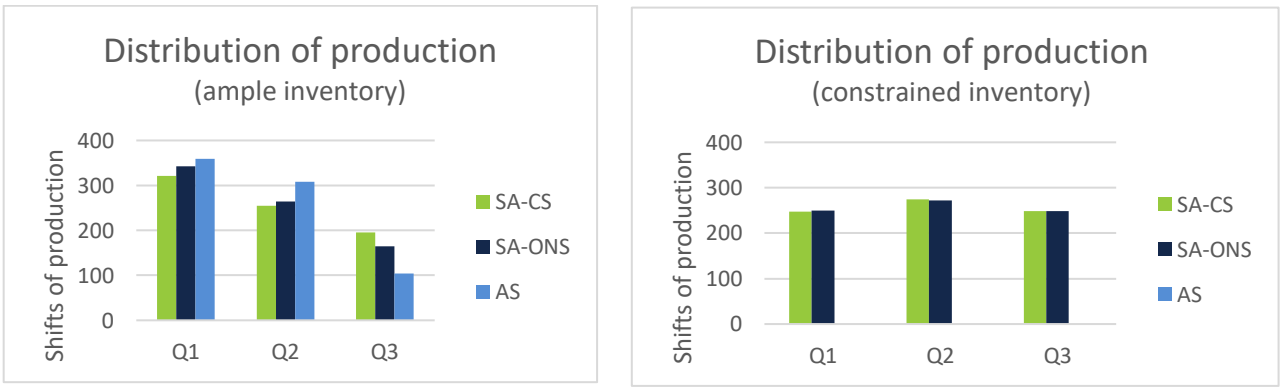


Figure 31: Distribution of production with ample (left) and constrained (right) inventory space

Conclusion

In summary, we find that the SA-CS algorithm produces the best results of all heuristics. CPLEX outperforms all heuristics for all instances, which is as expected. For more complex models, SA-CS performs better than CPLEX in terms of computational time versus solution strength. This becomes clear when looking at production line 846, whose complexity made it difficult for CPLEX to find a good solution within reasonable time. It takes CPLEX 121 minutes to find a solution that is equal to the solution found by the SA-CS algorithm within 5 minutes.

SA-CS finds schedules with less changeovers and production that is distributed more evenly over the weeks compared to SA-ONS and AS. The greatest decrease of costs is in the changeover costs, which are directly linked to the number of changeovers. The decrease in changeovers is not compensated

by higher inventory levels, instead production is distributed more evenly over the weeks and average inventory levels are lower in the better performing algorithms.

5.3 MODEL VERSUS REALITY

In this section, we compare the results of the best algorithm with the schedule created by the planner. For this, we only look at the first 12 weeks of the created schedule. Figure 32 shows the expected inventory value of all production lines in the scheduled weeks.

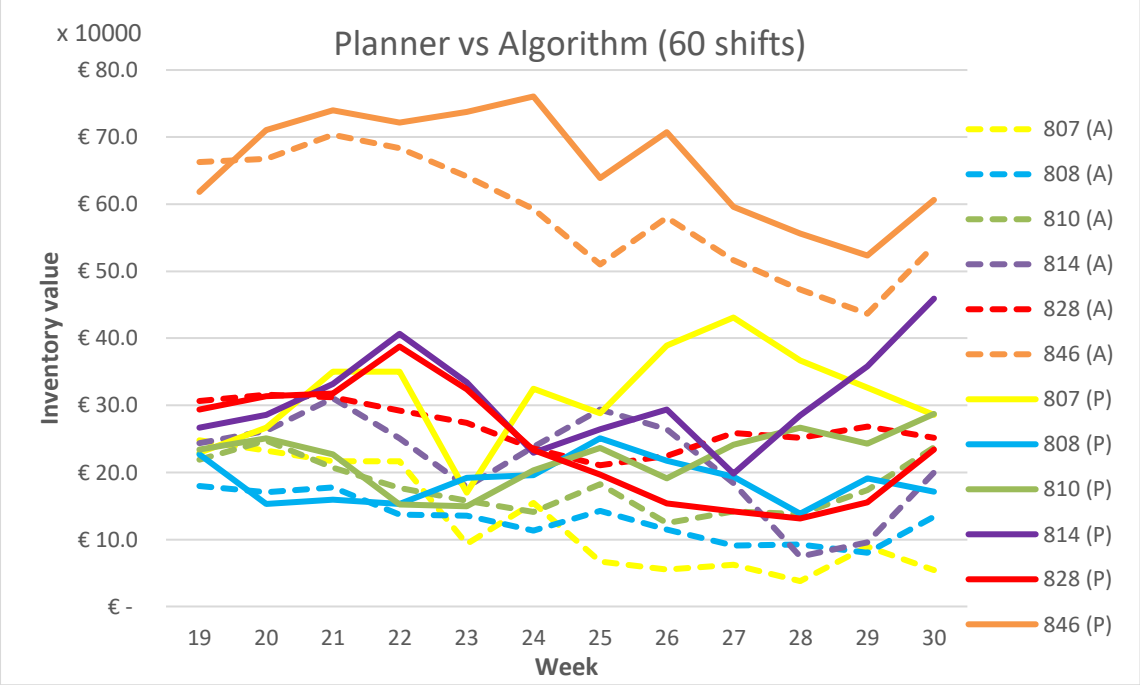


Figure 32: Comparison of SA - CS schedule and schedule created by planner

The SA-CS algorithm has on average a 32.1% lower inventory value during all weeks. In the last six weeks, this increases to a 47.5% lower inventory value. In general, the algorithm keeps less inventory compared the what the planner schedules. Especially for production line 846 the planner keeps more inventory, compared to what the algorithm schedules.

Production is distributed over the weeks, with the planner scheduling most production in the first six weeks. The algorithm schedules more production in the last six weeks of the schedule, which is likely caused by the high initial inventory levels. The algorithm schedules less production to quickly lower the inventory levels in the first weeks.

Table 23: Production shifts and changeovers in part 1 (Q1) and part 2 (Q2).

	Q1		Q2	
	Production	**Changeovers**	**Production**	**Changeovers**
SA – CS	194 shifts	20 shifts	214 shifts	15 shifts
Planner	272 shifts	23 shifts	236 shifts	24 shifts

Additionally, less changeovers are scheduled by the SA – CS algorithm (see Table 24). While the planner uses on average 3.9 changeovers per week, the SA – CS algorithm uses 2.8 changeovers per week. The algorithm also uses shorter runs, by distributing runs over multiple weeks. Instead of scheduling 1.5 million cans of production in one week, the SA – CS algorithm schedules two runs of

750,000 cans in two different weeks. The production line stays set-up for that item, but the run is split up to decrease average inventory levels.

Table 24: Overview of performance measures

	Production	Changeovers	Changeovers (avg. per week
SA – CS	408	35	2.8
Planner	508	47	3.9

We also looked at the results per production line. We find the following results for each of the production lines:

Production line 807:
The planner keeps a lot of spare inventory, approximately €285K versus €122K recommended by the algorithm. In the algorithm, we see that each item is set-up for roughly six weeks in a row. During this time, the item set-up is produced when required. In the week of a changeover, we notice a jump in inventory and production. The algorithm ensures enough product is stocked to cover the weeks that the item is not set-up and thus cannot be produced.

Production line 808:
Both the planner and algorithm schedule items together that require no or minimal setup. Similarly, to production line 807, the average inventory level is higher. An obvious difference is the distribution of production, as Figure 33 shows. By distributing production over several weeks, the average inventory value is lowered, as more items are produced just-in-time.

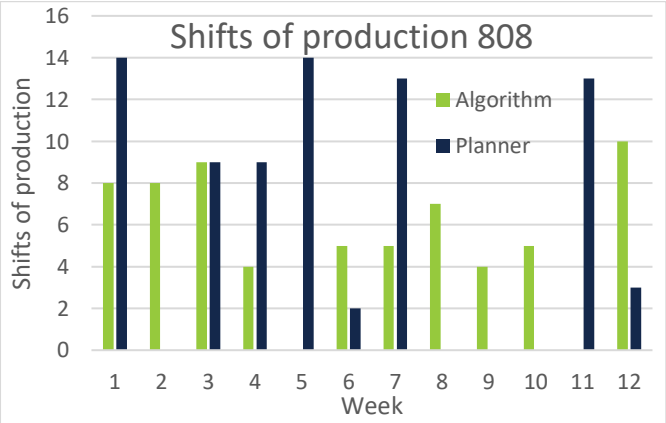


Figure 33: Shifts of production scheduled in week 1-12.

Production line 810:
Production line 810 produces one item with high demand throughout the weeks for a major customer. The algorithm schedules this item to be setup the majority of the weeks, producing the other demanded items in batches and placing them in inventory. Just as with production line 808, production is distributed more evenly throughout the weeks.

Production line 814:
Two major items are produced on this line, which require a small changeover. For the line, the schedule created by the planner and the algorithm shows the most overlap. An item is produced two or three weeks in a row, after which the switch is made to the other major item. In between two major items, the others are created in batches to cover up to 10 weeks of demand and placed in stock.

Production line 828:
For production line 828 we again notice a big gap between the inventory levels given by the algorithm and those planned by the planner. The planner keeps on average 1201 pallets in stock versus 872 pallets kept in stock by the algorithm.

Production line 846:
Production line 846 produces the greatest number of items of all production lines. The gap between the algorithm and planner is the biggest out of all production lines, which can be expected due to the high number of items. Surprisingly, no big differences are found between the schedules, with an equal number of changeovers and similar order of production. Again, the main difference is found within the inventory levels, where the planner schedules high levels of spare inventory throughout the weeks.

5.4 Conclusion

This section gives a short recap of the results found in previous experiments. We evaluate the heuristics proposed in Chapter 4 by means of several experiments. We compare the heuristics to our model which we solve by using CPLEX.

We compared a total of three heuristics, using four different datasets. For all datasets, the SA-CS heuristic has the best results in terms of objective costs. SA-ONS comes in second and the worst schedules are created by AS. We found that the number of changeovers has a significant effect on the total costs found by each algorithm. SA-CS created schedules with the lowest number of changeovers, while AS schedules have the highest number of changeovers.

When comparing the schedule created by SA-CS to the schedule created by the planner, we find that the currently used schedules can be improved upon. In general, the planner schedules more production and keeps more product on stock. This is likely a form of safety stock, despite there being no agreements to keep safety stock for these items. Additionally, we see that the planner schedules more changeovers, which amount to a significant part of the costs incurred, as seen in Section 5.2 above.

6 CONCLUSIONS AND RECOMMENDATIONS

In this chapter we conclude our research and provide recommendation with regards to the implementation of our results and in terms of future work that could improve the research done. In Section 6.1, we discuss our concluding remarks related to the steps performed during this research and the outcomes that we found. Section 6.2 discusses the limitation of our research. Section 6.3 provides recommendations for Trivium and future work with regards to our research.

6.1 CONCLUSION

The question we aimed to answer during this research is as follows:

> ***What can Trivium do to structurally improve their planning process to minimize the production- and inventory costs, while meeting all demand?***

This need arose due to uncertainty about the optimality of their current planning process and no clear overview of the costs directly associated with planning and scheduling. To structure our research, we define and introduce several research questions in Chapter 1. Here we present a summary of the most relevant remarks, gained by answering the research questions throughout our research.

To answer our main question, we first complete a detailed analysis of the current situation, especially the planning and scheduling process currently in place and the costs that are influenced by this scheduling process. With regards to the planning process, we find that most is done manually in Microsoft Excel. This simple planning strategy makes it difficult to view the effect of scheduling decisions on the total cost incurred. Regarding the costs, we identify the following five major costs: cost of capital, external storage cost (storage, handling & transport costs), changeover costs and cost of overtime.

Aside from analyzing the current situation, we also perform a literature study focused on the scheduling problem found at Trivium. We find an MIP model that matches the Trivium production process, based on the lot-sizing and scheduling problem. We formulate the mathematical model in Section 4.2 and use it for benchmarking the results we obtain from our heuristics. During our literature study, we discuss the most compatible algorithms, found in literature, which can be used for designing such a heuristic. As a result of the literature study, we decide to use simulated annealing and adaptive search for designing our heuristics.

When designing our heuristics, we decompose the problem, which creates two subproblems that are more easily solved separately compared to the full problem. Subproblem one concerns the storage and production capacity of the 3PC department and distributes this among the production lines. When solving subproblem two, we create a production schedule for each of the production lines, according to the capacity constraints given by subproblem one.

When comparing our heuristics and the MIP model, we find large differences. Of the three heuristics, simulated annealing with the classic neighborhood strategies performs the best, followed by simulated annealing with our own created neighborhood strategies. Adaptive search performs worst in all situations, the problems being too complex for adaptive search to find a good feasible solution. The SA-ONS algorithm likely constrains the movement to neighbors too much, hindering its ability to find good solutions. Following expectations, the MIP model, solved in CPLEX, finds the best results for all production lines and scenarios. While SA-CS does not outperform the MIP model, it is particularly

useful for bigger problems, in this research production line 846, due to the quick solution time and quality of solution to go with it.

The results from SA-CS also outperforms the schedules created by the planner in all cases. The biggest differences found are within the average inventory levels, as the planner tends to keep safety stock for items, without existing agreements regarding this. Smaller improvements are found within the number of changeovers and distribution of production. Despite this, even more improvement is possible when considering the results found by using the MIP model.

In general, a planning and scheduling program based on a simulated annealing heuristic is the best option when the problems get more complex. When scheduling is limited to a few items or multiple items with very low demand, an MIP model with the CPLEX solver is best. While there is still improvement possible, as shown by our MIP model, the heuristic can improve on the schedule which is currently used.

6.2 LIMITATIONS

Within our research there are several limitations. Data was collected manually and only during the duration of the research. The size of the datasets means there is overlap between the different datasets, ranging from 16 weeks to 23 weeks of overlap. On the other hand, part of the dataset is based on forecasted data, which varies as the weeks progress. Especially for the datasets with 16 weeks of overlap, the fluctuations in the forecasted demand reduces the similarities between the datasets. While we used the OEE as a way of including robustness in the model, this is mostly focused on fluctuations in the production levels. Robustness could be increased further by analyzing the forecast quality and adjusting our production and stock levels to be able to handle forecast fluctuations, but analyzing the forecast was outside the scope of our research.

Within our model, we also make several assumptions that affect how good the model reflects the real-life situation. We assume each production line has a set storage capacity, while in reality it is first-come, first-serve and each production line uses whatever capacity is available when needed. Additionally, we removed some flexibility from the schedule by assuming all changeovers require 8 hours, when changeovers can be reduced by assigning more personnel or vice versa. It is possible that better schedules can be made when changeovers are flexible, but this would also increase the complexity of the model significantly.

The adaptive search algorithm often has problem finding a feasible solution. By selecting the adaptive search parameters, we put too much focus on the optimality of the solution and not the feasibility. This could be solved by using variable parameters that initially focus on feasibility but later shift towards optimality. This would increase the chance of a feasible solution found, while also focusing on finding an optimal solution. Downside to this is the increase in the algorithm length, as the algorithm is effectively split up in two separate phases.

Furthermore, we program our algorithm in Java, but do not use a graphical user interface. This makes it difficult for the company to use the algorithm in its current state. Additionally, data is input manually, as the available tools do not allow connecting to the database of Trivium Packaging.

6.3 RECOMMENDATIONS

Based on the results of this research, we present several recommendations.

First, we recommend a strict evaluation of the inventory levels throughout the company. The results show large improvements can be made regarding the stock levels. When the company decides to take steps in this direction, they should make sure they consider the fluctuations in the forecasted demand. While the model distributes production and ensures there is spare capacity in most weeks, due to the low stock levels it might not be able to compensate high increases in demand on short term. Currently, there are no agreements on deviations that are accepted with regards to orders versus demands, which makes creating a good and robust schedule difficult.

Second, during scheduling the planner should look towards distributing the production over multiple weeks instead of producing everything within one week. Distributing production ensures a lower average inventory and keeps the schedule flexible. When production is distributed, a delay in production does not automatically mean the whole production schedule must be moved further ahead. A balance must be found in this, as shortening production runs can increase the difficulty for production.

With regards to future work, we recommend more research into the usage of simulated annealing. Based on the MIP model, there is still room for improvement, which could be found in finetuning the parameters further. Improvements for simulated annealing could be found by using variable penalties, instead of a fixed penalty which we currently use. The penalty should vary based on the level of infeasibility, which means we must include a measure of infeasibility in the model. Currently this is not present, making it not possible to use a variable penalty. Using a penalty that varies with the measure of infeasibility would increase the change of chooses infeasible solutions that are close to being feasible. It would reduce the chance of getting stuck in an infeasible zone and possibly remove the need of resetting the solution at the end of a Markov chain.

For future work with regards to the MIP model, we recommend the usage of the output of simulated annealing as the initial solution. By using such an initial solution, the computation time of the MIP model could be reduced, making it a viable solution strategy for also the big problems.

Additionally, we would recommend the addition of forecast reliability in future research. During our time at Trivium, we noticed that the forecast can be very unreliable, which makes scheduling an everchanging task. Research into the forecast reliability and methods to improve this, would be of great assistance towards efficient scheduling. Without a reliable forecast, it might prove difficult to decrease the inventory levels as the schedule will not be able to cope with all demand fluctuations.

BIBLIOGRAPHY

Alem, D., Curcio, E., Amorim, P., & Almada-Lobo, B. (2018). A computational study of the general lot-sizing and scheduling model under demand uncertainty via robust and stochastic approaches. *Computers and Operations Research, 90*, 125-141. doi:10.1016/j.cor.2017.09.005

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

Anthony, R. N. (1965). *Planning and Control Systems: A Framework for Analysis*. Harvard University. Graduate School of Business Administration: Division of Research, Graduate School of Business Administration, Harvard University.

Ardagh Group. (2019). Ardagh Group. Retrieved from https://www.ardaghgroup.com/

Bertsimas, D., & Tsitsiklis, J. (1993). Simulated Annealing. *Statist. Sci., 8*(1), 10-15. Retrieved from doi:10.1214/ss/1177011077

Carvalho, F., Sofia Pereira, A., Constantino, M., & Pedroso, J. P. (2001). *Tabu Search for a Discrete Lot Sizing and Scheduling Problem*.

Copil, K., Wörbelauer, M., Meyr, H., & Tempelmeier, H. (2017). Simultaneous lotsizing and scheduling problems: a classification and review of models. *OR Spectrum, 39*(1), 1-64. doi:10.1007/s00291-015-0429-4

Davari, M., & Demeulemeester, E. (2019). The proactive and reactive resource-constrained project scheduling problem. *Journal of Scheduling, 22*(2), 211-237. doi:10.1007/s10951-017-0553-x

Fazel Zarandi, M. H., Sadat Asl, A. A., Sotudian, S., & Castillo, O. (2018). A state of the art review of intelligent scheduling. *Artificial Intelligence Review*. doi:10.1007/s10462-018-9667-6

Fleischmann, B. (1990). The discrete lot-sizing and scheduling problem. *European Journal of Operational Research, 44*(3), 337-348. doi:https://doi.org/10.1016/0377-2217(90)90245-7

Gicquel, C. (2008). MIP models and exact methods for the Discrete Lot-sizing and Scheduling Problem with sequence-dependent changeover costs and times.

Gicquel, C., Minoux, M., & Dallery, Y. (2008). Capacitated Lot Sizing models: a literature review.

Haase, K. (1994). *Lotsizing and scheduling for production planning*. Berlin ;: Springer-Verlag.

Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization, 5*(3), 423-454. doi:10.1007/s13675-016-0075-x

Herman, M. (2010). *Simulated Annealing & the Metropolis Algorithm : A Parameter Search Method for Models of Arbitrary Complexity*.

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research, 165*(2), 289-306. doi:https://doi.org/10.1016/j.ejor.2004.04.002

Hsu, W.-L. (1983). On the General Feasibility Test of Scheduling Lot Sizes for Several Products on One Machine. *Management Science, 29*(1), 93-105. Retrieved from doi:10.1287/mnsc.29.1.93

IBM Corp. (2019). IBM ILOG CPLEX Optimization Studio (Version 12.9.0.0). Retrieved from https://www.ibm.com/products/ilog-cplex-optimization-studio

Jans, R., & Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research, 177*(3), 1855-1875.

Karimi, B., Fatemi Ghomi, S. M. T., & Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega, 31*(5), 365-378. doi:https://doi.org/10.1016/S0305-0483(03)00059-8

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics, 34*(5-6), 975-986. doi:10.1007/BF01009452

Kolisch, R., & Drexl, A. (1996). *Adaptive Search for Solving Hard Project Scheduling Problems* (Vol. 43).

Laarhoven, P. J. M. v., & Aarts, E. H. L. (1987). *Simulated annealing : theory and applications*. Dordrecht.

Lambert, D. M., & Stock, J. R. (1993). *Strategic logistics management* (3rd ed. ed.). Homewood, IL: Irwin.

Leach, J. C., & Melicher, R. W. (2015). *Entrepreneurial finance* (Fifth edition. ed.). Stamford, CT, USA: Cengage Learning.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), 1097-1100. doi:https://doi.org/10.1016/S0305-0548(97)00031-2

Nakajima, S. (1988). *Introduction to TPM : total productive maintenance*. Cambridge, Mass.: Productivity Press.

Osman, I. H., & Kelly, J. P. (1996). Meta-Heuristics: An Overview. In I. H. Osman & J. P. Kelly (Eds.), *Meta-Heuristics: Theory and Applications* (pp. 1-21). Boston, MA: Springer US.

Sel, C. a. r., Bilgen, B., & Bloemhof-Ruwaard, J. (2017). Planning and scheduling of the make-and-pack dairy production under lifetime uncertainty. *Applied mathematical modelling, 51*, 129-144.

Silver, E. A. (2004). An Overview of Heuristic Solution Methods. *The Journal of the Operational Research Society, 55*(9), 936-956.

Zijm, W. H. M. (2000). Towards intelligent manufacturing planning and control systems. *OR-Spektrum, 22*(3), 313-345. doi:10.1007/s002919900032

Appendix A. CURRENT CHANGEOVERS

For each line the changeovers are described between identical sub-specifications. When the sub-specification is not identical, a big changeover is needed. The tables below only specify changeovers between products whose sub-specification is the same. All changeovers between items with different sub-specifications are big, which is why we do not specify them.

THE REST OF THIS SECTION IS CONFIDENTIAL.

Appendix B. PSEUDOCODE

Algorithm 1: Simulated Annealing

```
1   Create initial solution          >> Algorithm 2: Initial Solution
2   Set algorithm parameters
3   Set best solution to initial solution
4   While temperature > Stop temperature && NoImprovement < 30 do
5           Copy current solution to new solution
6           While n < Markov Chain Length do
7                   Create neighbor solution          >> Algorithm 4: Random neighbor strategy
8                   Set NoImprovement = NoImprovement + 1
9                   If neighbor solution is feasible then
10                          Calculate cost difference
11                          Calculate acceptance probability          >> Algorithm 4: Acceptance probability
12                          If acceptance probability > threshold then
13                                  Accept neighbor solution as current solution
14                                  Set NoImprovement = 0;
15                          End if
16                          If Obj(current solution) < Obj(best solution) then
17                                  Set current solution as best solution
18                          End
19                  End if
20                  n = n + 1
21          End
22          Decrease temperature with cooling rate
23  End
```

Algorithm 2: Initial Solution

```
1   Load input data of demand to be scheduled
2   For Week = 0 to MaxWeek do
3           Assign demand of week to production
4   End For
5
6   For Week = MaxWeek to 0 do
7           Calculate overtime currently in week.
8           If Overtime greater than zero Then
9                   Move production equal to overtime one week earlier
10          End if
11          Sequence production in week
12  End for
```

Algorithm 3: Random neighbor strategy

```
1  Create random number from 0 to NumberOfNeighbor strategies
2  Switch random number
3          Case 0:
4                  Use Neighbor strategy 1
5          Case 1:
6                  Use Neighbor strategy 2
7          ….
8
9  End Switch
```

Algorithm 4: Acceptance probability

```
1  If NeighborCost <= CurrentCost Then
2          Return 1.0
3  Else
4          Return exp((CurrentCost - NeighborCost)/Temperature)
5  End if
```

Algorithm 5: Adaptive search

```
1  Load machine and demand data
2  Set bestObjective value to maximum integer
3  Do
4          Create new solution                    >> Algorithm 6: Adaptive iteration
5          If Solution is feasible AND newObjective < bestObjective Then
6                  bestSolution = newSolution
7                  bestObjective = newObjective
8
9          End if
10 While Stop requirement not met
```

Algorithm 6: Adaptive iteration

```
1  Load machine and demand data
2  Remove demand that can be fulfilled from inventory from demand data
3  While not in FinalWeek AND demand left to assign Do
4          If First iteration in week Then
5                  Assigned demand of current week to production
6          End if
7          Calculate spare capacity in week
8          Calculate probability to move to next week
9          If move to next week Then
10                 Sequence production in week
11                 Week = week + 1
12         Else
13                 Choose item to produce
14                 Find first demand of item to schedule for production
15         End if
16 End while
```

Appendix C. CPLEX MODEL

This appendix shows all the code used during the creation of the CPLEX model.

```
/*********************************************
 * OPL 12.9.0.0 Model
 * Author: Anne
 * Creation Date: 18 Aug 2019 at 11:13:16
 *********************************************/
//{string} Weeks = ...;
int NbWeeks = ...;
range Weeks = 1..NbWeeks;
int NbLocations = ...;
range Locations = 1..NbLocations;

Products = ...;

tuple productData {
           string name;
           string specification;
           float costprice;
           int unitsperpallet;
           int productionspeed;
           int safetystock;

}

productData Product[Products] = ...;

           string locname[Locations] = ...;
           float transportcost[Locations] = ...;
           float holdingcost[Locations] = ...;
           int WarehouseCap[Locations] = ...;


// table that stores demand (in pallets)
int Demand[Products][Weeks] = ...;
int ChangeCosts[Products][Products] = ...;

// Parameters
float CostofCapital = ...;
int ShiftsAvailable[Weeks] = ...;
int ChangeTime[Products][Products] = ...;
int InventoryStart[Locations][Products] = ...;
string InitItem = ...;
int N = ...;
int M = ...;
int W = ...;

// decision variables
dvar int+ ShiftProduction[Products][Weeks];
dvar int+ Inventory[Locations][Products][Weeks];
dvar int+ Set-up[Products][Products][Weeks] in 0..1;
dvar int+ ProduceProduct[Products][Weeks];
dvar int+ MovingPallet[Locations][Products][Weeks];
dvar int+ SubDemand[Locations][Products][Weeks];
dvar int+ CarriedSet-up[Products][1..NbWeeks+1] in 0..1;

// costs var
dvar float+ totalchangeover;
dvar float+ totaltransport;
dvar float+ totalcoc;
dvar float+ totalholding;

dvar int Y[Weeks] in 0..1;

minimize totalchangeover + totaltransport + totalcoc + totalholding;


subject to {

  // inventory balance
  forall( i in Products, l in Locations, t in Weeks) {
           InvBal:
           if(t == 1) {
                     Inventory[l][i][t] == InventoryStart[l][i] + MovingPallet[l][i][t] - SubDemand[l][i][t];
           } else {
                     Inventory[l][i][t] == Inventory[l][i][t-1] + MovingPallet[l][i][t] - SubDemand[l][i][t];
                     }
           }

  forall(l in Locations, t in Weeks) {
    MaxInv:
    sum(i in Products)(Inventory[l][i][t]) <= WarehouseCap[l];
}

 forall(i in Products, t in Weeks) {
 sum(l in Locations)(MovingPallet[l][i][t]) == ShiftProduction[i][t]*Product[i].productionspeed;
}
  forall( i in Products , t in Weeks ) {
    ctDemand:
      Demand[i][t] == sum(l in Locations)(SubDemand[l][i][t]);
    }

  forall(t in Weeks) {
    capacity:
           sum(i in Products)(ShiftProduction[i][t])+sum(i in Products, j in Products) (Set-up[i][j][t]) * ChangeTime[i][j] <=
ShiftsAvailable[t];
}
```

```
   carryset-up2:
          sum(i in Products)(CarriedSet-up[i][NbWeeks + 1]) <= 1;

   forall(t in Weeks) {
    carryset-up:
          sum(i in Products)(CarriedSet-up[i][t]) == 1;
   }


   forall(i in Products, t in Weeks) {
     Requireset-up:
          (sum(j in Products)(Set-up[j][i][t]) + CarriedSet-up[i][t])*M >= ShiftProduction[i][t];
          }

   forall(i in Products, t in Weeks) {
     BalanceSet-up:
          sum(j in Products) (Set-up[j][i][t]) + CarriedSet-up[i][t] == CarriedSet-up[i][t+1] + sum(j in Products)(Set-
up[i][j][t]);
          }

   forall(i,j in Products : j != i, t in Weeks) {
     SequenceReq:
          ProduceProduct[i][t] + N*Set-up[i][j][t] - (N-1) - N*CarriedSet-up[i][t] <= ProduceProduct[j][t];
    }

   forall(i, j in Products, t in Weeks : i == j  ) {
     DiagSet-ups:
          Set-up[i][j][t] == 0;
    }

   if (InitItem != "") {
            forall(i in Products: Product[i].name == InitItem, t in Weeks: t == 1) {
              InitialSet-up:
                   CarriedSet-up[i][t] == 1;
             }
  }

    totalchangeover == sum(i,j in Products, t in Weeks)(ChangeCosts[i][j] * Set-up[i][j][t]);
          totaltransport == sum(i in Products, l in Locations, t in Weeks)(MovingPallet[l][i][t]*transportcost[l]);
          totalcoc        ==        sum(i        in        Products,        l        in        Locations,        t        in
Weeks)(Inventory[l][i][t]*Product[i].unitsperpallet*Product[i].costprice*CostofCapital);
          totalholding == sum(i in Products, l in Locations, t in Weeks)(Inventory[l][i][t]*holdingcost[l]);


   forall(t in Weeks, l in Locations: l == 2) {
     Warehouse1:
          sum(i in Products) (MovingPallet[l][i][t]) <=  W*Y[t];
}
   forall(t in Weeks, l in Locations: l == 1) {
     Warehouse2:
          WarehouseCap[l] - sum(i in Products) (Inventory[l][i][t]) <= (1-Y[t])*W;
}

}

execute DISPLAY {
for(var t in Weeks)
  for(var i in Products)
          for (var j in Products)
                    if(Set-up[i][j][t] == 1)
                    writeln("Set-up[", i, "][", j, "][", t, "] = ", Set-up[i][j][t]);

  writeln();

          writeln("changecosts " + totalchangeover);
          writeln("transportcosts" + totaltransport);
          writeln("costofcapital" + totalcoc);
          writeln("holdingcost" + totalholding);

          for(var l in Locations) {
                    var avginventory = 0;
                    var avgvalue = 0;
                    for(var t in Weeks) {
                              for(var i in Products) {
                                        avginventory += Inventory[l][i][t] * Product[i].unitsperpallet;
                                        avgvalue += Inventory[l][i][t] * Product[i].unitsperpallet * Product[i].costprice;
                              }
                    }
                    writeln("Location " + l);
                    writeln("Inventory " + avginventory/NbWeeks);
                    writeln("Inventory value  " + avgvalue/NbWeeks);
                    writeln("");
          }

          for (var t in Weeks) {
          var capacityused = 0;
                    for(var i in Products) {
                              capacityused += ShiftProduction[i][t];
                              for(var j in Products) {
                                        if(ChangeCosts[i][j] != 0) {
                                                  capacityused += Set-up[i][j][t]*ChangeTime[i][j];
                              }
                              }
                    }
          writeln("Week(c)" + t +": " + capacityused);
          }

}
```

Appendix D. ADAPTIVE SEARCH PARAMETER TUNING

Below are the results for the different alphas and iterations for all production lines.

807	50	200	1000	2500	5000
250	33799.16	32589.05	30400.64	29160.56	28758.45
150	33264.08	32119.14	**29625.06**	29338.62	**28592.29**
50	**32680.02**	**31818.32**	30373.87	**28914.71**	28923.97
1	0	0	42556.34	0	41879.72
0	0	39993.29	0	0	44537.85

808	50	200	1000	2500	5000
250	41630.29	37997.64	32299.13	30112.27	29589.87
150	**39047.22**	**35743.23**	33098.83	**29798.13**	**28743.01**
50	40907.66	38042.93	**32128.1**	30044.76	29364.51
1	60342.95	50325.09	43004.11	38808.46	34991.94
0	52022.04	50721.9	40322.65	38135.48	36299.52

810	50	200	1000	2500	5000
250	47343.47	51053.81	43451.16	40055.65	**38929.55**
150	55981.02	**47365.13**	43410.15	40533.69	39435.87
50	**42604.5**	47429.22	**42447.2**	**39893.35**	39997.42
1	62633.03	60230.01	54490.76	51815.07	48538.24
0	64881.13	59163.72	50621.41	46947.27	45472.7

814	50	200	1000	2500	5000
250	0	0	0	0	0
150	0	53017.64	55018.68	53267.7	52861.59
50	**54525.96**	53394.82	51251.82	52874.9	50715.82
1	0	**52095.39**	**50613.35**	**50776.44**	**49330.2**
0	0	0	59531.01	56208.05	53034.29

828	50	200	1000	2500	5000
250	0	0	0	0	0
150	0	0	**32787.5**	0	0
50	0	0	0	**41486.6**	**36298.58**
1	0	0	0	0	0
0	0	0	0	0	0

846	50	200	1000	2500	5000
250	**79703.92**	76336.69	70948.29	68801.92	68184.02
150	80290.28	76293.73	72193.11	**68390.06**	**66850.84**
50	80143.37	**73909.9**	**70764.96**	68752.85	67717.31
1	88500.14	84281.66	80654.51	80367.42	79807.77
0	91967.8	89815.74	86077.57	84389.12	83257.12

Appendix E. EXPERIMENTAL RESULTS

Here we include the experimental results for the datasets not covered in the main text. For all datasets, the best results are found by the SA – CS algorithm. This is followed by SA-ONS and last in performance is AS.

Table 25: Results of all production lines for data of 08-05-2019

	230807	230808	230810	230814	230828	230846
AS Cost	€ 34,276	€ 25,618	€ 36,899	€ 43,575	€ 33,307	€ 70,466
ONS Cost	€ 14,478	€ 14,580	€ 16,730	€ 24,577	€ 14,199	€ 37,917
CS Cost	€ 7,758	€ 11,578	€ 12,974	€ 19,403	€ 11,220	€ 28,927

Table 26: Results of all production lines for data of 07-01-2019

	230807	230808	230810	230814	230828	230846
AS Cost	€ 30,405	€ 23,089	€ 42,714	€ 32,631	€ 29,432	€ 60,328
ONS Cost	€ 12,824	€ 12,496	€ 13,520	€ 25,548	€ 15,577	€ 33,784
CS Cost	€ 7,658	€ 10,680	€ 11,053	€ 20,086	€ 12,628	€ 23,117

Table 27: Results of all production lines for dataset 06-24-2019

	230807	230808	230810	230814	230828	230846
AS Cost	€ 31,056	€ 24,740	€ 46,859	€ 33,694	€ 26,853	€ 71,056
ONS Cost	€ 15,245	€ 12,643	€ 13,059	€ 27,592	€ 16,338	€ 36,832
CS Cost	€ 6,728	€ 10,527	€ 10,821	€ 17,518	€ 13,759	€ 28,171