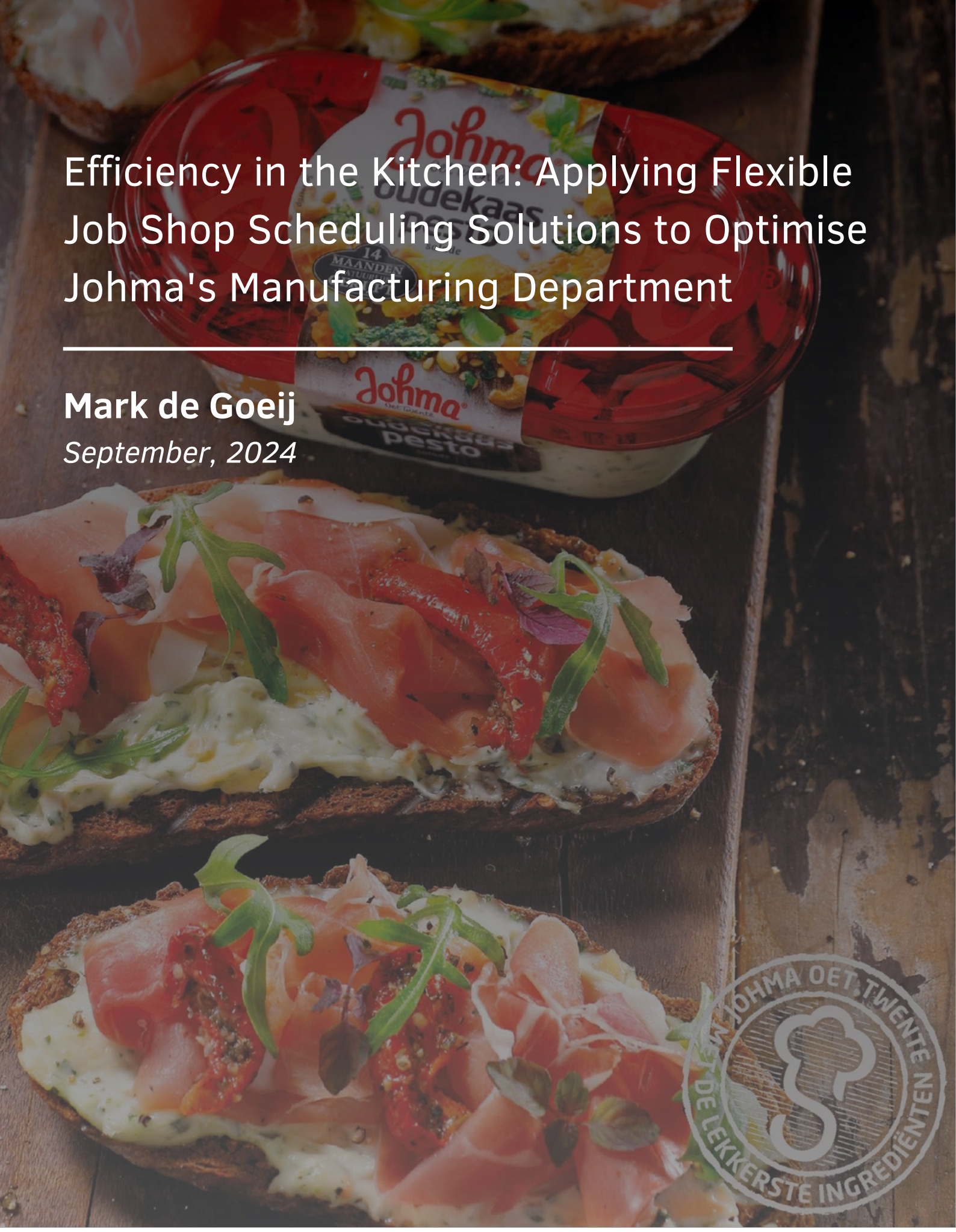


Efficiency in the Kitchen: Applying Flexible Job Shop Scheduling Solutions to Optimise Johma's Manufacturing Department

Mark de Goeij

September, 2024

Johma Oet Twente

UNIVERSITY OF TWENTE.

Master thesis Industrial Engineering and Management

September 2024

Specialisation: Production and Logistics Management

Efficiency in the Kitchen: Applying Flexible Job Shop Scheduling Solutions to Optimise Johma's Manufacturing Department

Author:
M.A.A. de Goeij (Mark)
m.a.a.degoeij@student.utwente.nl

Johma Salades
De Pol 35
7581 CZ Losser
(053) 537 3400

University of Twente
Drienerlolaan 5
7522 NB Enschede
(053) 489 9111

Supervisor Johma Salades
D. van der Woude (Daniël)
Head of Continuous Improvement

Supervisors University of Twente
Dr. E. Topan (Engin)
ir. J. Lok-Visser (Jedidja)

Preface

Dear reader,

You are about to read the master thesis titled "Efficiency in the Kitchen: Applying Flexible Job Shop Scheduling Solutions to Optimise Johma's Manufacturing Department." This research was conducted at Johma Salades in Losser, the Netherlands, as the final assignment for my master's degree in Industrial Engineering and Management at the University of Twente.

During my time at Johma, I learned a lot about the operations of a sizable production company, particularly in the domain of production scheduling. Working on this topic was highly engaging, and I significantly improved my programming, writing, and analytical skills throughout the assignment. The insights and experiences gained within the company revealed the complexities of an organization with diverse departments, each with unique needs and challenges. I discovered that theoretical knowledge alone is insufficient; people management skills and cooperation at all levels are essential. I enjoyed working in such a multifaceted environment and hope to continue in similar roles in my future career. I am grateful to all the employees at Johma who assisted me, as their support was invaluable in understanding the company processes, especially on the shop floor. I also want to thank my company supervisor, Daniel van der Woude, for providing me with this opportunity. Despite the individual nature of the assignment, I could always seek guidance and information when needed.

Additionally, I want to thank my first supervisor, Engin Topan, from the University of Twente. From our first meeting to the last, his constant positivity made our meetings enjoyable. His valuable feedback and personal interest in the assignment kept me motivated and committed to the research. I also appreciate the support from my second supervisor, Jedidja Lok-Visser. Her feedback and insights, drawn from her own recent experience as a master's student, were particularly relatable and helpful.

This thesis marks the end of my time at the University of Twente, bringing an end to my student life in Enschede. The past six years have been amazing, and I want to thank everyone I met during this period. Special thanks go to my football team, Meisterschaft V2, my fraternity, Directus Calvatus, my former board members at Unipartners Twente, and my friends from Gewoon Gezellig Doen and V.A. Coq, for all the amazing activities and experiences. I am deeply grateful to my family and roommates who always showed interest in my thesis and kept me motivated during challenging times. Special thanks go to my girlfriend who is by far the biggest contributor to my motivation and pushed me through difficult times. Thank you Rozan, I appreciated this a lot and will never forget this.

For the students who are going to start on their thesis, know that it will challenge you both theoretically and mentally. The process may be tough at times, but the reward is well worth it in the end.

I hope you, the reader, enjoy reading this master thesis!

Mark de Goeij

Losser, September 2024

Management summary

Johma Salades is a leading company in the salad manufacturing industry located in Losser, the Netherlands. Johma manufactures all kinds of salads for their brand as well as private labels. To stay competitive, Johma needs to make its production processes as efficient as possible. Johma's main manufacturing department, Kitchen 1, which focuses on bread and toast salads, is not performing efficiently enough. The main reason for this is that the current production scheduling method is too basic or even non-existent in some elements. This causes the salad mixtures to be delivered too late, leading to idle times for the corresponding filling lines. In the previous two production years, Kitchen 1 was responsible for idle times at the filling lines, which accounted for almost 18% of the total production time. Therefore, the core problem identified is the lack of an optimal scheduling method for the jobs and operations in Kitchen 1. This research aims to develop a scheduling approach for Kitchen 1 by answering the following main research question:

'How can Johma near-optimally schedule the required jobs and operations to prepare and mix the salad mixtures in Kitchen 1 to meet the demand of the filling lines, decrease the idle time, and increase flow in the production facility?'

A comprehensive analysis of Johma's production system, focusing on Kitchen 1 and its filling lines, revealed that the company employs both Make-To-Stock (MTS) and Make-To-Order (MTO) processes. Kitchen 1 uses a Pull system that produces salads in two phases: preparation and mixing, driven by demand from the filling lines. The scheduling problem in Kitchen 1 can be classified as a flexible job shop scheduling problem, characterised by multiple salad recipes, unique sequences of preparation and mixing, sequence-dependent setups (cleaning), and constraints on machine and operator eligibility. Currently, the scheduling process is managed based on experience and a basic filling line schedule, leading to inefficiencies and difficulties in ensuring timely delivery to the filling lines.

We performed a literature review to classify Johma's scheduling issue as a flexible job shop scheduling problem. This system is characterised by precedence constraints, sequence-dependent setups, machine and operator eligibility, and working interval limitations. Given the problem's NP-hard nature and its size, heuristic approaches were chosen over exact methods. A mathematical model (MILP) was developed to understand the problem better, and dispatching rules such as Earliest Due Date (EDD), Minimum Slack (MS), and Random were applied to generate initial solutions, tested with both forward and backward scheduling approaches. These dispatching rules were gathered from literature and are the best options for our problem. These solutions were then optimised using improvement heuristics, namely Iterative Local Search (ILS), Simulated Annealing (SA), and Tabu Search (TS), which are also mostly used in literature for comparable problems and because of usability, which is in terms needed for Johma.

The flexible job shop scheduling problem in Kitchen 1 is modelled by developing a mathematical model, and by modelling the heuristics. The input data, which consists of the job list and operation times, is described as well as the problem description, modelling decisions, and several assumptions and simplifications. The mathematical model helps to understand the problem in more detail. A constructive heuristic is made with the dispatching rules Earliest Due Date (EDD), Minimum Slack (MS), and Random to form an initial solution together with other logic for the sequence of operations, operator-machine assignment, setups, forbidden intervals, and scheduling approach. The initial solution is further improved upon with the improvement heuristics Iterative Local Search (ILS), Simulated Annealing (SA), and Tabu Search (TS) together with the neighbourhood strategies normal and random. Combining the constructive heuristics with the improvement heuristics, 18 solution approaches are proposed.

Experiments were conducted using seven problem instances to cover the variety and abundance of jobs, representing real-world job lists at Johma. The EDD dispatching rule as a constructive heuristic outperformed others, particularly when paired with forward scheduling. In terms of improvement heuristics, Tabu Search (TS) delivered the best results in tardiness and makespan with a reasonable runtime. Sensitivity analyses on operator capacity, machine capacity, and buffer times assessed the robustness of the scheduling model. Key findings include:

- **Operator Capacity:** Increasing the number of operators from three to four resulted in optimal tardiness of 0 for all instances, whereas with three operators the largest instances could not reach optimal tardiness. Makespan reductions between 4.18% and 24.43%, ensured schedules fit within employee working hours.
- **Machine Capacity:** Operating with a single mixing machine does not show a clear negative impact on the scheduling performance. Therefore, we cannot conclude if working with two mixers is substantially better.

- **Buffer Time Robustness:** Introducing a higher or lower buffer time between tasks does not reveal a clear pattern for positive or negative scheduling performance.

Overall, Figure 0.1 shows that the forward scheduling approach with the TS improvement heuristic, the EDD dispatching rule, and the Random neighbourhood strategy reduced the tardiness and makespan substantially. This showcases the model's ability to improve performance significantly and reduce idle times to nearly zero in theory for all instances with certain operator settings.

	Basic heuristic Forward - EDD			*Improvement heuristic TS - Forward - EDD - Random*				
Instance	**Tardiness (min)**	**Makespan (min)**	**Run Time (sec)**	**Tardiness (min)**	**Makespan (min)**	**Run Time (sec)**	**Gap T**	**Gap M**
1	44751.26	2360.53	4.45	9828.04	2069.63	2406.93	-78.04%	-12.32%
2	45740.84	2500.87	4.75	11947.50	2287.93	2830.26	-73.88%	-8.51%
3	497.40	1561.33	2.97	48.85	1512.57	1648.15	-90.18%	-3.12%
4	0.00	1132.13	2.74	0.00	1129.63	1540.83	-0.00%	-0.22%
5	0.00	1529.18	2.92	0.00	1526.68	1661.95	-0.00%	-0.16%
6	0.00	1178.82	2.42	0.00	1176.32	1223.25	-0.00%	-0.21%
7	260.06	1382.38	2.44	0.00	1252.23	1211.64	-100.00%	-9.41%

Figure 0.1: Model performance.

A usage and measures plan is provided to ease the transition from the developed model to using it in Kitchen 1. The input is explained and instructions are made to edit or add inputs when needed. Furthermore, a manual is written for Johma to install the required software and use the model accordingly. The output of the model is formatted into a scheduling list of operations with the required information for the operators on the shop floor. In addition, a step-wise plan is described to deal with such a change project to help with the transition. The following steps should be taken; Step 1: Establishing a sense of urgency, Step 2: Creating a guiding coalition, Step 3: Developing a vision and strategy, Step 4: Communicating the change vision, Step 5: Empowering broad-based action, Step 6: Generating short-term wins, Step 7: Consolidating gains and producing more change, and Step 8: Anchoring new approaches in the corporate culture.

The main recommendations of this research are to make the scheduling model operational with the provided settings, transition plan and written manuals and to use the model to check the feasibility of the planned recipes on the filling lines. Furthermore, before operational usage, Johma should focus on having the c-operator give tasks on the shop floor, having the runner communicate early to the cooling cell employees to have ingredients ready, updating the recipes to all be in the same format, further investigating when exactly the salad mixture should be ready at the filling lines, and measure even more processing times to make the model even more accurate. Lastly, storing the actual finish times of the salad mixtures in Kitchen 1 would fill the data gap to be able to analyse the performance of the scheduling method and to further make better data-driven decisions.

This research offers both academic and practical contributions. Academically, it addresses a flexible job shop scheduling problem with real-world constraints, bridging a gap in the literature regarding worker-machine eligibility and sequence-dependent setups with other elements like forbidden intervals and the fact that forward scheduling can be a more beneficial method. Practically, the model provides Johma with a robust solution to improve the efficiency of Kitchen 1, reducing idle time. Furthermore, insights are gained in the activities in Kitchen 1, processing times, operator capacity, finish times of recipes, and schedule feasibility. However, some limitations exist. The model assumes deterministic input, which may not reflect real-world variability such as machine breakdowns and ingredient shortages. Future research could explore dynamic rescheduling methods to handle real-time changes and extend the approach to other departments within Johma. Additionally, research into machine breakdowns and cart/funnel capacity could further refine the model's practical utility.

Contents

Reader's guide

This section provides a guide for the reader to be able to navigate between the chapters of this research.

Chapter 1: Introduction
This chapter starts with an introduction to the company. Furthermore, the main problem is described and a problem cluster is made to identify the core problem of this research. Moreover, the research design is provided and the scope is explained.

Chapter 2: Context analysis
This chapter gives an overview of the production processes within Johma. First, a general overview is given, and following from this Kitchen 1, the main department, is explained in more detail. Furthermore, a short description is given for the filling lines, and the current production planning and scheduling approach is explained in detail. At last, the important factors are listed.

Chapter 3: Literature review
This chapter provides a comprehensive literature review of the problem at hand. First, some concepts are explained. A taxonomy of the scheduling problem is made, and some additional research about NP-hardness and scheduling approaches is given. At last, the different solution approaches are explained.

Chapter 4: Model design
This chapter gives describes the model design. It starts with the input data. A complete mathematical model is given, together with the description, modelling decisions, assumptions and simplifications. In the end, the constructive heuristic is outlined as well as the different improvement heuristics to form the solution alternatives.

Chapter 5: Experiments and results
This chapter analyses experiments to see how the model performs. Several problem instances are tested with the solution approaches. Furthermore, a sensitivity analysis is done to test the impact of changing parameters on the performance. At last, the theoretical model is compared to real-world performance.

Chapter 6: Usage and measures plan
This chapter explains the steps needed to transition from the current situation to the scheduling method recommended in this research. The model usage is explained as well as the interpretation of results. In the end, some general information is given on how to deal with such a transition.

Chapter 7: Conclusions and recommendations
This chapter gives an overview of the main findings of this research. Recommendations are given, limitations are described, and suggestions for future research are provided. Lastly, academic and practical contributions are discussed.

List of abbreviations

ATC Apparent Tardiness Costs.
ATO Assembly-To-Order.

CODP Customer Order Decoupling Point.

EDD Earliest Due Date.
ETO Engineer-To-Order.

FJS Flexible Job Shop.
FJSP Flexible Job Shop Scheduling Problem.
FTEs Full-Time Employees.

GA Genetic Algorithm.

ILS Iterative Local Search.

JAS Johma Advanced Scheduling.

MILP Mixed Integer Linear Program.
MPSM Managerial Problem-Solving Method.
MS Minimum Slack.
MTO Make-To-Order.
MTS Make-To-Stock.

SA Simulated Annealing.

THT Tenminste Houdbaar Tot.
TS Tabu Search.

1 Introduction

This master's thesis is conducted at Johma Salades, focusing on the production processes involved in making bread and toast salads. Section 1.1 describes the company, Section 1.2 delves into the problem identification, and Section 1.3 provides the research design that guides the study.

1.1 Introduction to Johma Salades

The story of Johma Salades (further referred to as Johma) originates in 1968 when a man named Johan started to make salads in his garage in a small town near Losser, The Netherlands. The name Johma combines the names Johan and Martin, and the first customers were small supermarkets and butchers in the neighbourhood. In the seventies, salads became more and more popular and thus demand increased, resulting in the realisation of a larger production facility in Losser.

Currently, Johma is, among other food brands, part of a larger organisation named Signature Foods. Signature Foods consists of facilities in Losser, Oldenzaal, Schiedam, Den Haag, Nieuw Vennep, Almelo, Katwijk and Turnhout (BE) that all manufacture food products. The office in Hilversum acts as the main office, and the office in Osnabruck (GE) focuses on the German market. Figure 1.1 shows the locations of the different companies that are part of Signature Foods (Johma Salades, 2023).

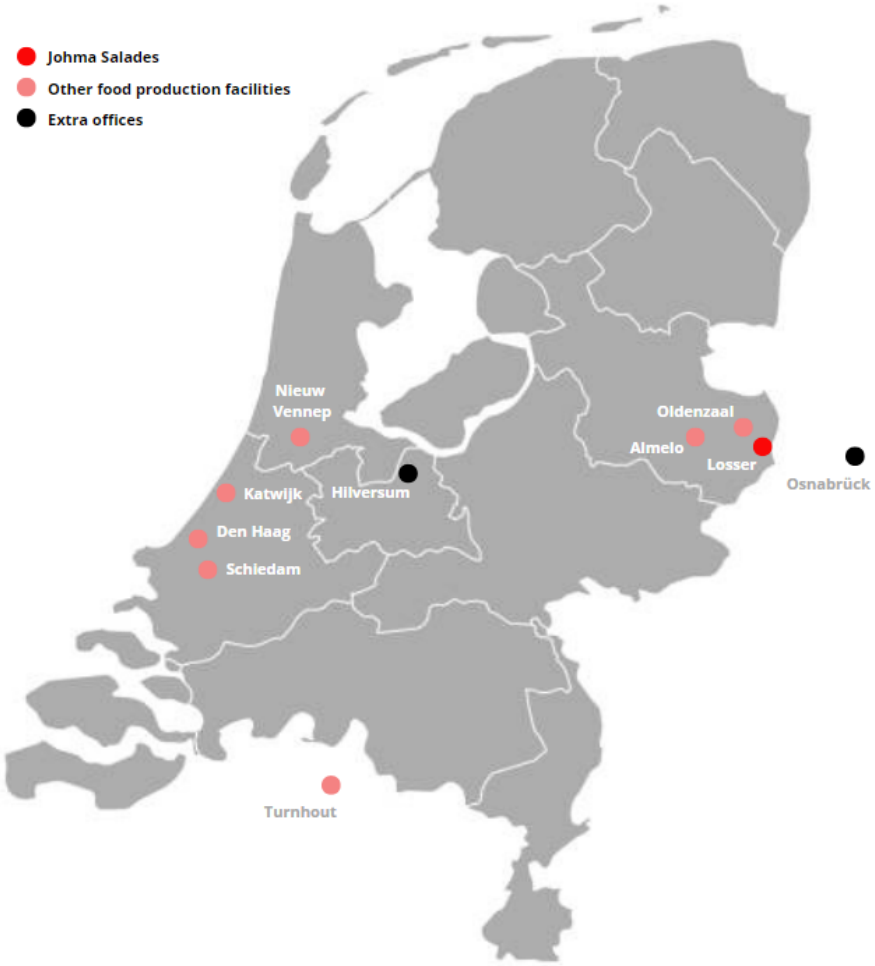


Figure 1.1: Signature Foods companies (Signature Foods, 2023).

Johma's production facility specializes in the manufacturing of salads for A-brands such as Johma and Homann, as well as B-brands for various private labels. The salads are divided into several product groups with different flavours. The main product groups are bread and toast, 100% plant-based, cream-cheese spread, platter salads, portion packs, meal salads, and raw vegetable salads. Furthermore, Johma also produces some of the above-mentioned salads for other private labels. As long as the salad mixtures can be put into certain plastic containers, Johma can manufacture them (Johma Salades, 2023). Figure 1.2 shows some examples of the products that Johma produces.

Figure 1.2: Product examples (Johma Salades, 2023).

The production facility is divided into several kitchens, each dedicated to making a certain type of salad mixture or sub-ingredient that is largely used (pasta or potato). This research focuses on the kitchen that makes the bread and toast salads for Johma and similar B-brands, which is also the largest one.

1.2 Problem identification

Johma, which is a market leader in the salad industry, is committed to maintaining its competitive edge in the market. To secure its future competitive position, the production facility must run smoothly to manufacture salads for the lowest costs possible, as optimising the production processes can substantially reduce production costs. This research aims to improve and optimise the production processes in the main production department (kitchen) that makes the bread and toast products to ensure that the filling lines, following from this department, receive enough salad mixtures to operate efficiently. These filling lines fill the plastic containers with the salad mixture to form the final product (see examples in Figure 1.2). Section 2.3 explains the filling process in more detail.

The following sections explain what problems occur in the main kitchen. Section 1.2.1 defines the action problem, which is the main reason why this research is conducted. Furthermore, Section 1.2.2 describes the problem context in more detail by giving insight into the underlying problems causing the identified action problem. From this overview, this section further clarifies which of the core problems, from the problem context, is selected as the main problem to solve within this research.

1.2.1 Action problem

Anything or any situation that is not how you want it to be, is an action problem (Heerkens & van Winden, 2017). In this situation, the reality deviates from the norm. To solve the problem you have to take action to make sure the reality is equivalent to the norm. Thus an action problem is a discrepancy between the norm and the reality, as perceived by the problem owner.

Johma has eighteen filling lines to process the salad mixtures into a final product (salad mixture into a plastic container as shown in Figure 1.2), and four kitchens to make these salad mixtures. Currently, filling lines 6, 13, 14, 16, and 31 are fully supplied by the main kitchen, and filling lines 2, 4, 11, and 18 are partially supplied by the main kitchen. From now on this main kitchen is referred to as Kitchen 1. All other filling lines are supplied by one or a combination of three of the other kitchens. Figure 1.3 shows that the filling lines supplied by Kitchen 2 have the most idle time. However, Kitchen 2 is currently being improved by the continuous improvement team. Therefore, this research focuses on the filling lines supplied by Kitchen 1, which have the second highest idle time. Idle time is the time that the filling lines are waiting for the salad mixtures from the kitchens, which can also be seen as the starvation of the filling lines. Idle time does not have to be a problem. However, when there is idle time due to starvation, the production process is at a standstill. This could indirectly lead to employees not being used or other costs leading from this. Because of this, idle time is a problem. The other costs cannot

directly be identified as a consequence of idle time and because of this idle time is regarded as the main problem.

Kitchen 1 is thus unable to keep up with the demand of the filling lines, causing the filling lines to wait for salad mixtures before they can continue manufacturing. This is a problem that has already been ongoing for a long time, but Johma did not have time yet to solve this. The reality and the norm are expressed in the number of hours the filling lines are waiting for the salad mixtures, which is the idle time. Figure 1.3 shows the reality from Johma's most recent production years, 2022 and 2023.

filling line	idle time	production time	idle percentage	department(s)	idle kitchen 1	idle kitchen 2	idle kitchen 3	idle kitchen 4
2	593.33	1,164.24	51%	kitchen 1 + 2	**59.33**	534.00		
4	756.27	1,115.09	68%	kitchen 1 + 2 + 4	**264.69**	378.14		75.63
5	910.28	1,817.35	50%	kitchen 2 + 4		728.22		182.06
6	616.34	3,100.62	20%	kitchen 1	**616.34**			
7	361.93	1,530.01	24%	kitchen 3			361.93	
8	204.77	1,022.98	20%	kitchen 2		204.77		
10	696.58	1,995.51	35%	kitchen 4				696.58
11	713.14	514.54	139%	kitchen 1 + 2	**570.51**	142.63		
12	327.46	965.11	34%	kitchen 3			327.46	
13	411.28	2,722.74	15%	kitchen 1	**411.28**			
14	285.54	3,539.44	8%	kitchen 1	**285.54**			
16	312.19	3,527.88	9%	kitchen 1	**312.19**			
18	847.27	1,586.87	53%	kitchen 1 + 2	**423.64**	423.64		
21	85.67	2,762.26	3%	kitchen 3			85.67	
30	223.31	5,240.49	4%	kitchen 3			223.31	
31	319.96	3,932.2	8%	kitchen 1	**319.96**			
32	1,291.72	1,570.24	82%	kitchen 2 + 4		775.03		516.69
33	664.01	3,571.94	19%	kitchen 2 + 4		398.41		265.60
				total idle time:	3,263.48	3,584.83	998.37	1,736.56

Figure 1.3: Total idle time (starvation) per kitchen for corresponding filling lines.

The filling lines supplied by Kitchen 1 together have been idle for 3,264 hours in total in the last two production years because there is no salad mixture available (starvation of the filling lines) compared to the 18,535 hours of total production time. Thus almost 18% of the time the filling lines are idle because of a lack of salad mixtures from Kitchen 1. Therefore, there is a discrepancy between the norm and reality. The problem owner in this research is Johma or even in more detail the head of the continuous improvement team. The action problem at Johma is the following:

'The filling lines that are supplied by Kitchen 1 are idle for 18% of the total production time, and Johma wants this to be 15% of the total production time.'

The reduction of 17%, which is reducing the current 18% to 15%, is determined by talking with the head of continuous improvement and is based on his experience, and on the wishes of the company.

1.2.2 Problem cluster and core problem selection

Problem identification is crucial to get to the root cause of the action problem. By conducting interviews with staff, observing the processes, and analysing data and existing written information the problems, causing the action problem, are identified. A problem cluster is made to show the relationships between the action problem and the underlying problems. According to Heerkens and van Winden (2017, p. 22), this is needed to show the connections and to communicate these to the problem owner, which in this case is Johma. Figure 1.4 shows the problem cluster.

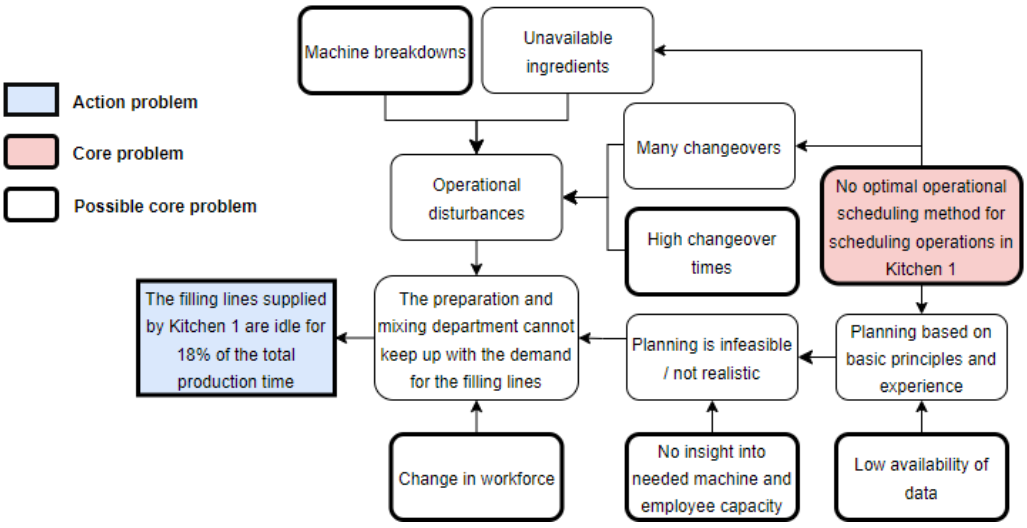


Figure 1.4: Problem cluster.

The following possible core problems can be considered in Figure 1.4:

- **Change in workforce:** Johma hires flex workers at a staffing agency because the production of salad mixtures is variable over the year. For example, during Christmas, Easter, and the summer the demand for salads is higher than in other periods of the year. Therefore, a flexible workforce is essential to meet this fluctuating demand. The downside of flex workers is that they often do not know how to operate the machines or how to execute manual activities. Full-Time Employees (FTEs) at the department explain and show how the work is done and this takes time in which they cannot do work themselves. In addition, every week the temporary workers can be different, and thus the whole cycle starts again. It is a problem that cannot be influenced within this research as the company that provides flex workers is currently unable to provide the same workers over and over. This is a problem that should be considered by Johma to potentially solve in the future.

- **Machine breakdowns:** During the production process several machines are used for mixing and preparation of salads. The three mixers in the mixing department have very few breakdowns. Most machine breakdowns occur in the preparation department. There are eight machines in the preparation department; a pilling machine, four cutting machines (a small one, a big one, the Holac, and the Urschel), a can opener, an egg machine and a vibrating sieve. From these machines the Holac has the most breakdowns according to the operators and the pilling machine is next in line. A breakdown often leads to not being able to prepare the ingredients for the salad mixtures. Besides a production stoppage, the employee(s) operating the machine are also idle when a breakdown occurs. While these operators may occasionally perform other operations, the majority of their time is dedicated to operating the machine, and thus machine breakdowns affect the production output. However, this possible core problem is not chosen to solve as no data is available on the breakdowns, and breakdowns do not happen frequently. Even the machines with the most breakdowns do not break down that often.

- **High setup times:** There are two types of setups; cleaning and machine equipment setups. Cleaning occurs when a different recipe than the one before is processed to prevent cross-contamination of allergens. Cleaning the mixers and the machines is time-consuming, and is done by several operators. Besides cleaning, machine equipment setups also happen. Machine equipment setups occur at the machines in the preparation department when a different ingredient than the one before has to be edited. The Holac for example, has three different knife sets that depend on the type of ingredient that has to be cut. The equipment setups are not that frequent and also do not take much time. The cleaning, however, takes a lot of time and happens more frequently. Although it is hard to influence the high cleaning time, as cleaning needs to be done properly to prevent cross-contamination, it is taken into account in the model in Chapter 4, because it is an important operation between different recipes.

- **No insight into required machine and operator capacity:** Several operators work in Kitchen 1. Next to operating the machines, the operators also execute several manual operations such as depositing raw ingredients into carts. The mix of the number of operators and needed machines to operate efficiently is unknown to Johma. This leads to inefficient use of the operators and machines and in the end an

inefficient supply to the filling lines. In Chapter 5 a sensitivity analysis is done to see what impact operator and machine capacity have on the performance of the model.

- **Low availability of data:** Currently production is scheduled based on experience, gut feeling, and a basic list of recipes. A reason for this is the low availability of data in Kitchen 1. To start with, the processing, setup, and transport times are not accurately known, which makes it very hard to indicate when to start manufacturing a certain salad mixture. Furthermore, the only indication the head operator (C-operator) has is the end time at which the salad mixtures should be available for the filling lines. For example, the list indicates that four salad mixtures should be made of the same recipe for filling line 13, and that this should be finished at 9:00. It thus only indicates the start time at the filling line of the first salad mixture. The other three salad mixtures do not have to be finished at this time, but the time at which it does have to be finished is unknown. With the lack of this information, it is difficult, and in most cases impossible, to know when the manufacturing of a certain recipe has to start within the preparation department, as there is no time indication for preparation and mixing operations. This leads to a sub-optimal sequence of operations being executed and time losses as the C-operator has to continuously look at the basic schedule on paper, estimate when to do what job and its operations, and then distribute the operations accordingly. Within this research, the data is collected by measuring the process times. Furthermore, a tool is made to calculate the different finish times for the single salad mixtures for the filling lines. With this information known, the start times can be obtained for the different jobs and operations.

- **No optimal operational scheduling method:** The current operational scheduling method for Kitchen 1 is very basic as mentioned in the previous problem. Currently, the C-operator of Kitchen 1 tells the employees what to do and when to do this. This is based on a sequence of recipes for the whole day that the C-operator receives at the start of the shift. Besides the low availability of data on when to execute what operation, the visual representation of the work that has to be done is not intuitive. It is a basic list of all recipes with amounts that have to be made for all the corresponding filling lines in a sequence of ascending completion times. With this information, the C-operator knows which group of the same recipe has to be finished first. However, every recipe takes a different processing time, and thus a recipe that has to be completed earlier based on the list is sometimes not a recipe the employees at the preparation department should start with. For example, when the schedule on paper indicates that recipe 1 should be finished at 9:10 for filling line 6 and this recipe takes 10 minutes, and recipe 2 should be finished at 9:15 for filling line 11 and this recipe takes 20 minutes then you should start with recipe 2, but because there is no indication of processing times the employees start with recipe 1 as this is first on the list. There is thus no thought-through method of scheduling the operations of the recipes in Kitchen 1. Thus the operational scheduling method of jobs within Kitchen 1 is too basic or even non-existing. The aim of this research is to develop a method to schedule the operations in Kitchen 1 more optimally to reduce idle times at the filling lines.

According to Heerkens and van Winden (2017, p. 43-44) a problem is only a problem if you are sufficiently sure it occurs. Furthermore, a core problem does not have a direct cause itself. It is thus at the end of the problem chain in the problem cluster. Another thing to consider is that if you cannot influence the problem then it cannot be a core problem, because you would not be able to solve it. In the end, if you still have more core problems left, you should pick the one that has the greatest impact effect at the lowest cost. This can be executed based on an educated guess and with the help of the problem owner. Figure 1.4 shows that the last possible core problem is selected as the core problem of this research. This particular problem is the biggest because when solved it can substantially reduce the idle time at the filling lines. The challenge lies in the limited available data on the execution of activities within Kitchen 1, including the time required for these operations. The variability of recipes, each requiring different processing steps, adds to the complexity. Furthermore, the capability to produce several salad mixtures parallel to each other complicates the situation even further. The scheduling of the process activities must be adjusted towards the availability and speed of the different filling lines to create optimal flow within the production facility. All these points have to be taken into account when solving this scheduling problem. The following core problem is selected for this research:

'No optimal production scheduling method for the jobs and operations in Kitchen 1 to supply the corresponding filling lines properly'

1.3 Research design

According to Heerkens and van Winden (2017, p. 36), research is structured following the seven phases of the Managerial Problem-Solving Method (MPSM). This research also uses this approach. The MPSM consists of the following seven phases: (1) defining the problem, (2) formulating the approach, (3) analysing the problem, (4) formulating (alternative) solutions, (5) choosing a solution, (6) implementing the solution, and (7) evaluating the solution.

1.3.1 Research questions

This section outlines the research questions that are answered in this study. The main research question, derived from the problem statement, is as follows:

'How can Johma near-optimally schedule the required jobs and operations to prepare and mix the salad mixtures in Kitchen 1 to meet the demand of the filling lines, decrease the idle time, and increase flow in the production facility?'

To answer the main research question, several sub-research questions are formulated. These questions serve as the framework for the subsequent chapters in this study. The following sub-research questions are answered in the following chapters:

- **Research question 1: What production activities take place at Kitchen 1 and the corresponding filling lines, and how are these activities scheduled?**
  - *What do the current production activities, and production flow look like?*
  - *What is the role of the machines and operators in the production process and what is their performance?*
  - *What does the current method of scheduling the activities look like?*
  - *What does the current production planning procedure look like?*
  - *What are important factors, such as constraints, to be taken into account when making such a schedule?*

  Chapter 2 answers research question 1 to obtain a comprehensive understanding of the current situation at Johma. To achieve this comprehensive understanding, the current production activities need to be thoroughly mapped out and elaborated upon. Methods like observation, interviewing, and analysis of available data are used for this. With this, a more in-depth performance assessment of the machines and operators to identify areas for improvement needs to be done. Additionally, the existing production planning and scheduling approach is outlined to enhance understanding of this procedure. Ultimately, key constraints are identified as important factors in this context analysis. These constraints are about the relationships between certain resources, processes, and the different recipes.

- **Research question 2: What is known in the literature about production scheduling to increase production output and flow, and decrease idle times at the filling lines?**
  - *What scheduling models can be found in the literature that apply to Johma's situation?*
  - *How to model the production scheduling problem at Johma?*
  - *What solution approaches can be used to improve the production schedule at Johma?*

  Chapter 3 answers the second research question by executing an extensive literature review. A detailed overview needs to be made of the production scheduling methods that exist and how these can be used in the situation at Johma. Following this is a framework of possible solution approaches to improve the production scheduling problem at Johma.

- **Research question 3: How can the flexible job shop scheduling problem in Kitchen 1 be modelled?**
  - *What data is needed and how should the data be formatted and managed?*
  - *What model assumptions, simplifications, and decisions need to be made?*
  - *How can the scheduling problem be formulated into a mathematical model?*

  - *How can we make a feasible schedule with a constructive heuristic and improve it using an improvement heuristic?*

  Chapter 4 tackles research question 3 by formulating a mathematical model to address the problem at hand. The chapter delves into the selection and management of data, the way the model should make decisions, as well as simplifications and assumptions that are needed for the mathematical formulation. The mathematical model is used for understanding the problem, and heuristics are used as the problem is too difficult to solve with an exact approach. By integrating these components, several solution approaches are determined to experiment with in Chapter 5.

- **Research question 4: Which solution approach performs best, and what experiments can be conducted using the developed model to explore this performance further?**
  - What is the performance of the model?
  - Which solution approach performs best?
  - Which experiments can be performed to evaluate the model performance?

  Chapter 5 explores the performance of the model by executing several experiments, and by providing an extensive analysis of the results. The experiments provide insights into the effect of certain decisions by changing parameters in the model, enhancing informed future decision-making.

- **Research question 5: What measures can Johma adopt to ease the transition to the proposed scheduling approach?**
  - How can the model be used?
  - How can the results be interpreted?
  - What measures are needed to be able to use the model properly?

  Chapter 6 explains how to transition from the current operational scheduling approach to the proposed one. The chapter describes the use of the model as well as how to interpret the results. Furthermore, the measures that are needed before implementation are explained as the model can only be used properly in certain circumstances to be used in daily operations.

- **Research question 6: What conclusions and recommendations can be drawn for Johma based on the findings from this research?**
  - What are the key conclusions?
  - What are the main recommendations?
  - What practical and theoretical contributions does this research provide?
  - What limitations occur in this research?
  - What are the proposed directions for further research?

  The last chapter gives the main conclusions and recommendations drawn from this research to Johma. Besides this, contributions to practice and theory are given to state the relevance of this research. Limitations and future directions for research are also explained to elaborate on what would be beneficial to research next.

1.3.2 Scope

The scope of this research is on Kitchen 1, which makes the salad mixtures for the bread and toast products. Kitchen 1 comprises a preparation and mixing department and provides the corresponding filling lines with the salad mixtures. While Johma consists of several interconnected departments, Kitchen 1 holds the greatest potential for improvement, as Johma is already improving Kitchen 2. In addition, the corresponding filling lines have the second largest idle times as shown in Figure 1.3, and thus it is logical to solve the problems occurring in Kitchen 1. Processes occurring downstream and upstream of Kitchen 1's supply chain are not considered in this study. The filling lines supplied by Kitchen 1 are only considered when testing the model by using the production planning of the filling lines as input for the model. Besides this, data collection is needed to handle the low availability of data. The setup times cannot be influenced but will be regarded in the model and insight into required machine and operator capacity is gained within the sensitivity analysis. To conclude, this research focuses on the development of a scheduling method to properly schedule jobs and operations within Kitchen 1, as this department directly causes idle time at the filling lines.

2 Context analysis

This chapter provides a context analysis of the current situation at Johma and aims to answer the following research question:

'What is the current situation at Kitchen 1 and the corresponding filling lines?'

In the scope of this research, the focus is on bread and toast products, denoted as salads in the rest of the report. Section 2.1 explains the general salad production processes from beginning to end. Section 2.2 focuses on Kitchen 1, which consists of a preparation and mixing department, and Section 2.3 focuses on the corresponding filling lines. Section 2.4 explains the current production planning and scheduling procedure. Lastly, Section 2.5 elaborates on the important factors that need to be considered for the development of the model in Chapter 4.

2.1 General process overview

The production at the filling department is planned, based on forecasts, historical data, sales information, and client orders. The finished products (salads in plastic containers) are stored in a warehouse, and customers can order a certain amount of these salads, which are picked from stock and transported to the customers. This type of production system is called a Make-To-Stock (MTS) system (Olhager, 2010). However, Johma also produces based on a Make-To-Order (MTO) system as some clients order the precise amount and the production planning is adjusted with this information. Kitchen 1 makes the salad mixtures based on the planning from the filling lines. This means that the filling lines use all salad mixtures manufactured by Kitchen 1 and thus no stock is needed between these departments. The filling lines only use a buffer space in which the salad mixtures that are finished earlier by Kitchen 1, can be placed. Although salad mixtures can be placed earlier in this zone, it is not recommended to have much work in process because of the shelf life of the salad mixtures. This internal production system is called a Pull system as the demand is pulled by the filling lines from Kitchen 1 (Benton, 2011). This is done by the list of recipes Kitchen 1 receives and by oral communication between the operators of the filling lines and Kitchen 1. Salad mixtures are only made when demanded by the filling lines.

This section further explains the processes required to make a salad from beginning to end to better understand how Kitchen 1 is located within the internal supply chain. Figure 2.1 shows the general process overview where the white boxes, preparing and mixing, refer to the processes in Kitchen 1.

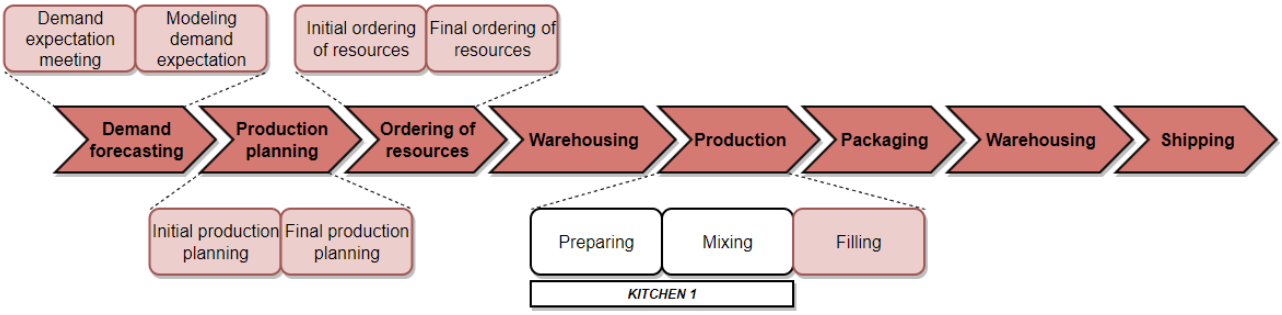


Figure 2.1: General process overview of salads.

Figure 2.1 shows that several actions are taken before the production of the salads can start. First, the sales team discusses with the customers how many salads the customer expects to buy. With this information, the forecasting department can make a forecast with their software system and historical data. Another software system provides the quantities that have to be made from the forecast information that is given. The planning department makes the production planning with the help of the information from the software system and some standard rules that they have. Ordering of ingredients can be done when the planning for production is released. Production starts in several kitchens depending on the type of salad that has to be made. Each kitchen prepares and mixes the ingredients into a salad mixture. These mixtures are transported in large funnels towards the corresponding filling lines. The filling lines fill up the plastic containers and the packaging department makes sure the salads (salad mixture in a plastic container) are palatalised. The pallets are stored in the warehouse and shipped to the customer when demanded.

Figure 2.2 shows what the bread and toast production process flow looks like. Kitchen 1, consisting of the preparation and mixing departments, and the corresponding filling lines are depicted within the grey rectangles. The six hexagons depict the other departments that are involved. The diagram shows the flow of ingredients

(coming from the cooling cell, warm kitchen or pasta kitchen) that either need editing by one or several machines or no editing at all. Following this, wet spices are added to the sub-mixture of ingredients and this mixture is mixed by one of the three mixers. The salad mixture produced by these processes is then transported to one of the corresponding filling lines to be further processed into a final product.

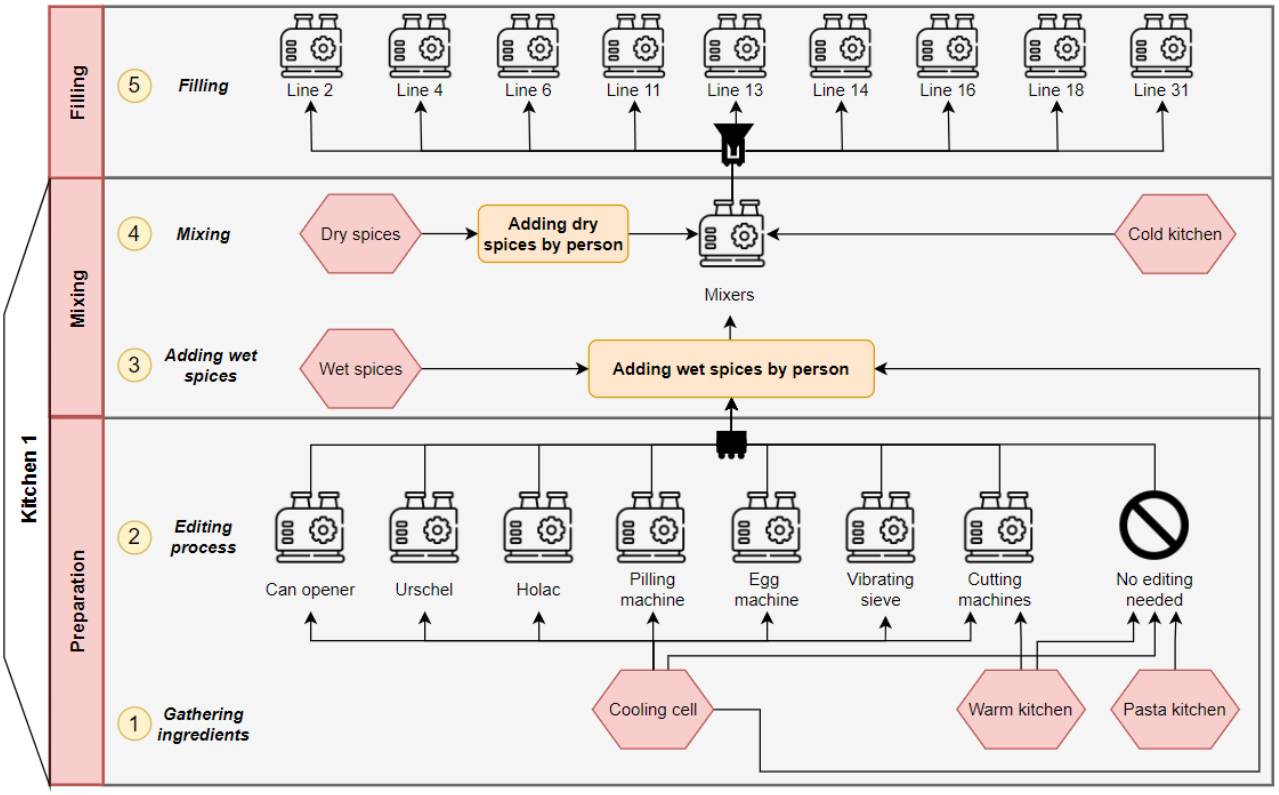


Figure 2.2: Production process of bread and toast products.

2.2 Kitchen 1

This section dives deeper into the preparation and mixing departments from Kitchen 1. Section 2.2.1 explains the recipes to give an impression of the variety and complexity that is needed to make these. Section 2.2.2 and 2.2.3 explain the preparation and mixing departments in more detail, respectively.

2.2.1 Recipes

There are many salad flavours that all have a different recipe for all the different brands (Johma and other private labels). In total, there are 91 recipes, and each recipe requires a unique set of ingredients, preparation and mixing steps. For instance, salads from Johma require more luxurious, and more abundant ingredients, such as higher quality tuna and salmon, which are used more generously in salads for Johma than those for other brands. Furthermore, there are 16 kinds of chicken ingredients, and some even need a different preparation activity. In total 114 unique ingredients are used in all the different recipes, and this does not even include all the different spice mixtures. Because of this, the production process activities are highly diverse, making the scheduling of jobs highly complex. Figure 2.3 shows the process flow of two different recipes to give an impression of the complexity. The numbers on the arrow lines depict the amount of ingredients. The production activities before adding wet spices are different per recipe and thus require different production steps.

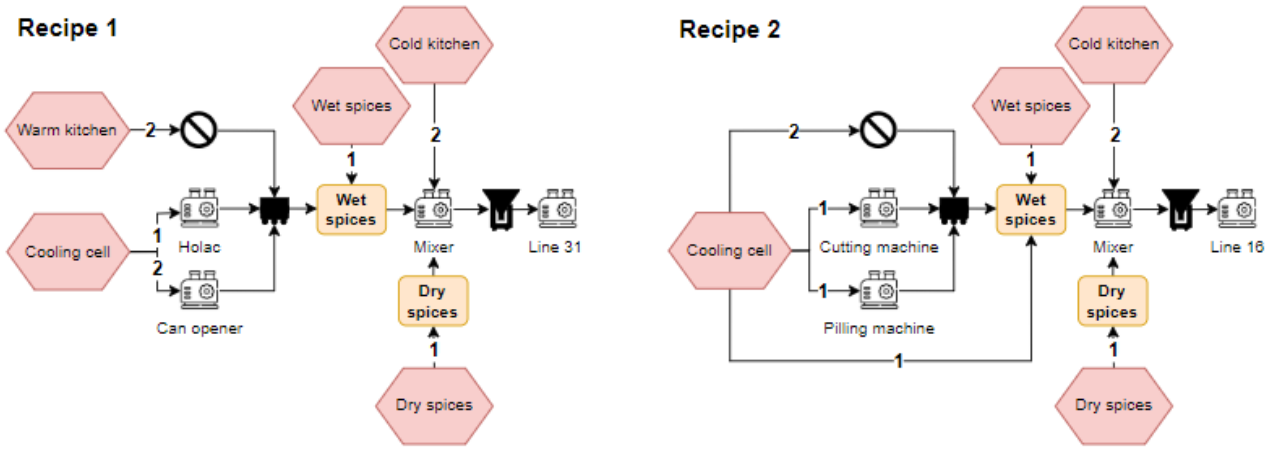


Figure 2.3: Process flow of two different recipes.

2.2.2 Preparation department

The preparation department department processes ingredients to form a sub-mixture of salad. Figure 2.4 shows the process flow in the preparation department.

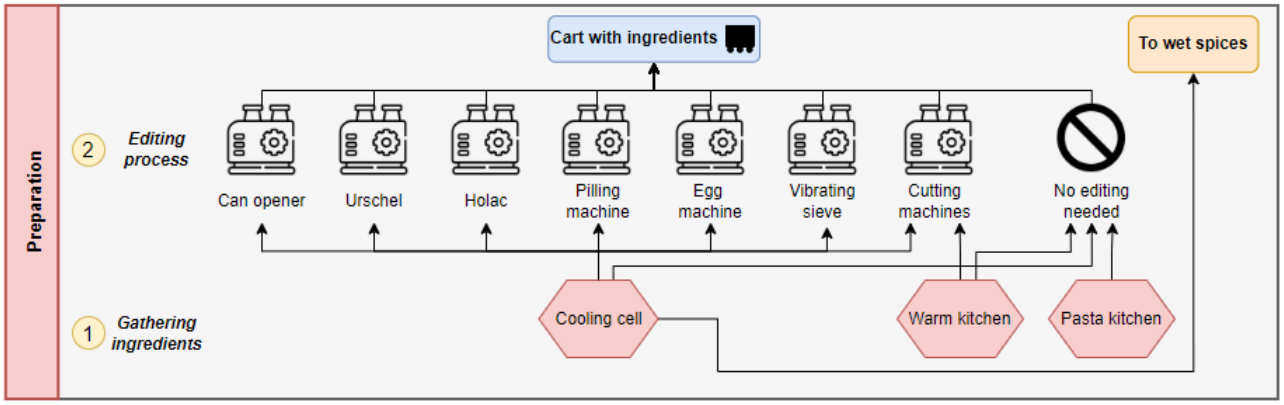


Figure 2.4: Detailed process flow of activities within the preparation department.

First, operators gather the required ingredients from several other departments. The following departments are involved:

- **Cooling cell:** The cooling cell is a large warehouse where ingredients, such as tuna, chicken, cans with pineapple or pickles, and various pre-made sauces (bought from external suppliers), are stored in a chilled or frozen condition. The ingredients within closed plastic bags are transported via a conveyor belt to be decontaminated. Other ingredients in open bags or buckets can be picked up using a pallet transporter.
- **Warm kitchen:** The warm kitchen blanches several ingredients. Blanching is cooking the ingredient for a short period and then submerging it in ice water. Ingredients that need blanching are for example cabbage, carrots, and walnuts.
- **Pasta kitchen:** The pasta kitchen cooks all kinds of pasta, which are used as an ingredient in some salad recipes.

The gathered ingredients fall into two main categories, namely processed and to-be-processed. While processed ingredients like pre-cut apples are ready to be deposited into the cart (as shown in Figure 2.5) immediately without the need for an editing procedure, to-be-processed ingredients, such as tuna, chicken, or eggs, first need one or several editing procedures, such are pilling, cutting, or pulverizing, respectively.

Figure 2.5: Example of a cart (known as "normwagens" at Johma).

There are 69 processed ingredients. An operator either deposits the ingredient immediately into the cart or first has to open the packaging or a bucket. Other processed ingredients are ingredients made by other departments. 42 to-be-processed ingredients need to be edited first. The editing procedures are executed with machines that are operated by one or two operators. The following machines perform the following editing procedures:

- **Can opener:** Nine ingredients can come from the cooling cell on a conveyor belt to the Can opener. One employee needs to operate this machine. The machine cuts open and empties the can into the cart. The machine flattens the empty cans and a conveyor belt transports the waste to a trash container.
- **Urschel:** Four ingredients need to be cut by the Urschel, operated by one employee. The operator either cuts open the packaging and deposits the ingredient into a compartment that cuts it into pieces onto the roller belt or opens the packaging and deposits the ingredient on the roller belt. The roller belt transports the ingredient into a shredder which shreds the ingredient into small pieces into a cart.
- **Holac:** Four ingredients need to be cut by the Holac. This machine is operated by one employee. The ingredients are sausages, coming from the cooling cell, that need to be removed from the packaging and placed in the machine by the operator. The machine cuts the sausages into the right size, and the cut ingredient falls into a cart. There are six knife sets to cut the sausages in the right size.
- **Pilling machine:** Ten ingredients need to be pilled by the Pilling machine. These ingredients come from the cooling cell. Two operators are working here; one is cutting the packaging open and one is emptying it on the conveyor belt. The conveyor belt goes into the machine and the ingredients are pilled. The pilled ingredient falls into a cart.
- **Egg machine:** Eggs need to be cut by the Egg machine. One employee needs to operate this machine. A large tank with eggs stands above a smaller compartment and the operator can deposit eggs into the compartment manually. The eggs are transported via a worm screw into another compartment. There are two knife sets to cut the eggs in the right size.
- **Vibrating sieve:** Pickles need to be sieved. The pickles come from the cooling cell. A barrel is transported over a conveyor belt to the machine. One operator needs to operate the machine and open the barrel. A plastic bag containing the pickles is cut open by the operator and the content is sieved by the machine. The sieved content falls into a large container. This can be scooped into a cart by the operator during the day when needed.
- **Large and small cutting machine:** Ten ingredients need to be cut by the Cutter. One employee needs to operate this machine. These ingredients come from the cooling cell and the warm kitchen. The operator opens and empties the bag into the machine. The machine cuts the ingredients into very small pieces. The operator can deposit the ingredients with the help of the machine in the cart when the ingredient is cut to the right size. The difference between the large and the small cutting machines is the amount and size of the ingredient that fits into the machines.

When the cart is filled with the sub-mixture of processed and/or to-be-processed ingredients, it is placed into a buffer zone for the mixing department.

2.2.3 Mixing department

The mixing department is involved in the processing of the sub-mixtures made by the preparation department. Figure 2.6 shows the process flow in the mixing department.

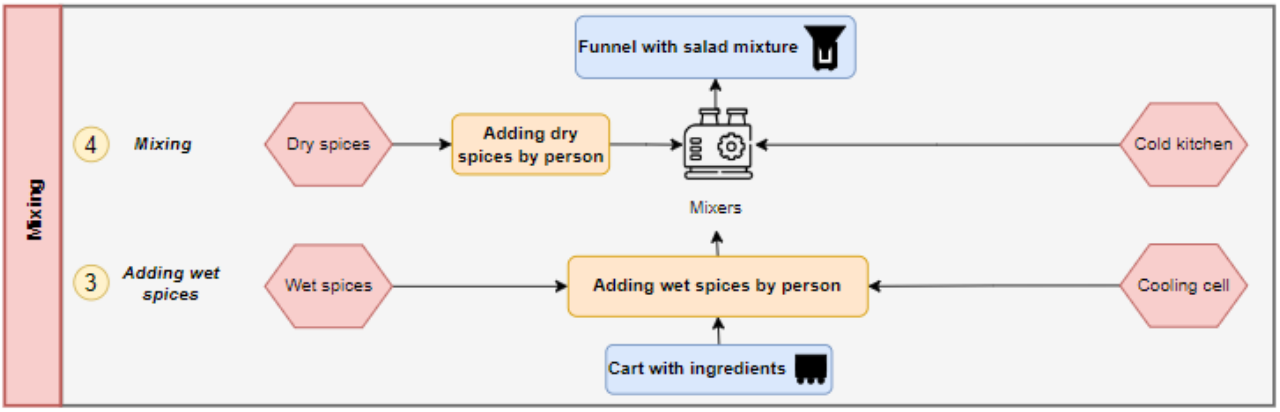


Figure 2.6: Detailed process flow of activities within the mixing department.

The following departments are involved:

- **Wet spices:** The pre-made wet spice mixtures are provided in buckets from the wet spices department.
- **Cooling cell:** Some more frequently used spices like tomato ketchup and chilli sauce are untapped from large tanks that are standing in the mixing department. They are coming from the cooling cell and have to be changed when empty.
- **Cold kitchen:** The cold kitchen makes sauces that are used in almost all recipes. These sauces are provided via pipelines to the mixers.
- **Dry spices:** The dry spices department is a warehouse where all pre-made dry spices, bought from external suppliers, are stored. These dry spices add flavour to the salad in addition to the wet spices. Furthermore, clear jell is made here, which is a binding substance.

One employee adds the wet spice mixtures into the cart and places it in the lift in front of the mixers. Two big mixers can mix two portions (the content from two carts) and one small mixer can mix one portion (the content of only one cart). When the content is in one of the mixers, the mixing procedure starts and consists of the following steps:

- **Step 1:** The operator adjusts the settings on the mixer for the right sauce, water and sugar water doses. The sauces are from the cold kitchen.
- **Step 2:** The operator transports the cart(s) with the sub-mixture(s) up with an automatic lift and empties the sub-mixture in the mixer. The operator operates the mixer to mix all the ingredients into the final salad mixture.
- **Step 3:** During the mixing, the operator adds dry spices and the clear jell (binding ingredient) manually.
- **Step 4:** The operator operates the mixer to deposit the salad mixture into a large funnel (see Figure 2.7) underneath the mixer.

Figure 2.7: Example of a large funnel (known as "trechter" at Johma).

The runner, an operator that transports goods from department to department, weighs the salad mixture and tests on quality before transporting it to the correct filling line. The operator cleans the mixer when a new recipe has to be made and otherwise continues with the next batch. The runner cleans the bottom of the mixer.

2.3 Filling lines

Several filling lines receive salad mixtures from the mixing department. Figure 2.8 shows the process flow at the filling department. In this research, we consider the filling lines that receive salad mixtures from Kitchen 1. Filling lines 6, 11, 13, 14, 16 and 31 are supplied completely by Kitchen 1, and filling lines 2, 4 and 18 are partially supplied by Kitchen 1. The runner transports the funnel with the salad mixture into a buffer zone at the filling lines. The buffer zone has space for 6 to 8 funnels per filling line. The filling lines all operate on their own with one operator and a whole setup of machines. We do not dive deeper into the specifications of this process as this is out of scope. Section 2.4 elaborates on the scheduling of jobs on the filling lines, which is important for this research.

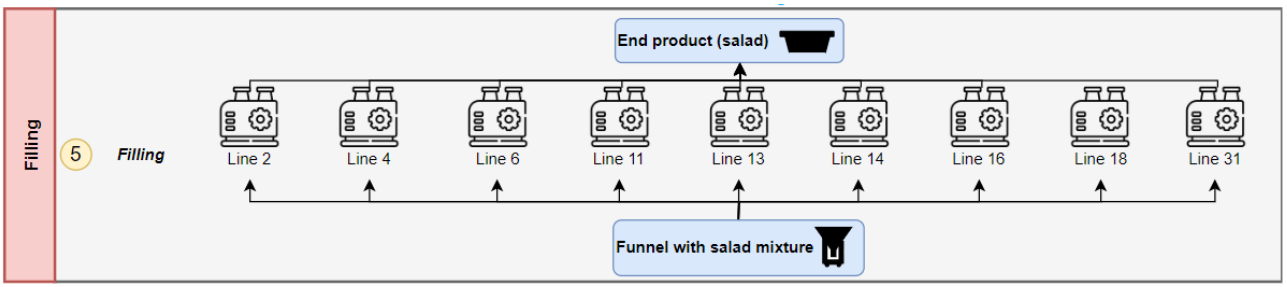


Figure 2.8: Detailed process flow filling lines.

2.4 Production planning and scheduling

This section explains the current production planning and scheduling approach. Section 2.4.1 explains the higher-level production planning in more detail and Section 2.4.2 elaborates on the lower-level scheduling of activities within Kitchen 1.

2.4.1 Higher-level production planning

Section 2.1 already gave a glimpse of what the production planning time horizon is and what is done at certain stages during this time horizon. This section explains in further detail how the production planning is made at a tactical level.

The current tactical production planning still has a short time planning horizon ranging from days to weeks. The planning department uses a software system called Johma Advanced Scheduling (JAS) to plan the production of the filling lines per two weeks, but can be changed per day. Figure 2.9 shows what one week of planning looks like in the software system.

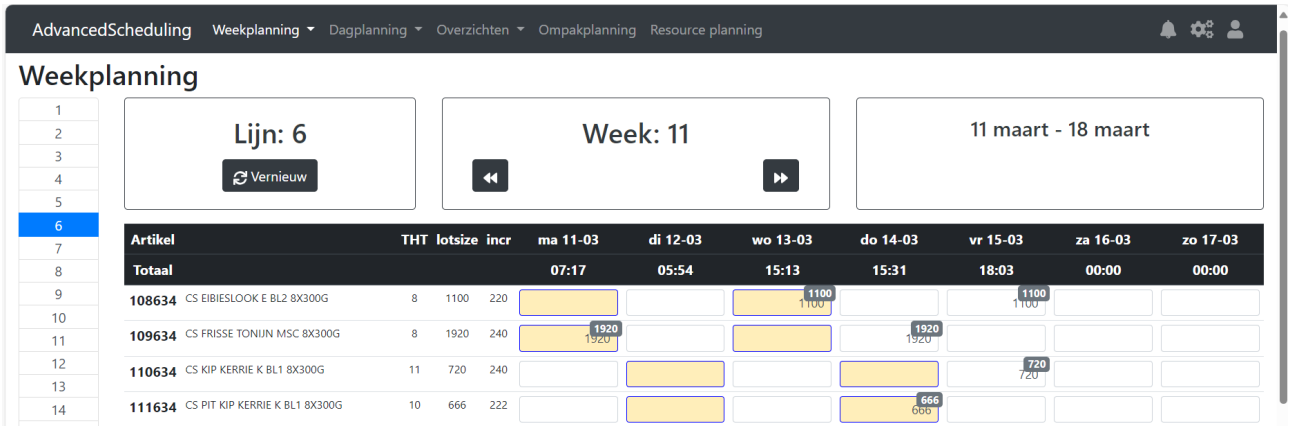


Figure 2.9: Weekly planning in JAS.

Figure 2.9 shows the planning for filling line 6 in week 11. Every recipe (article) has a dedicated filling line. The article numbers of the different recipes together with the name, internal Tenminste Houdbaar Tot (THT) (this means best before), lot size (run sizes), and increment (increment of run size) are given in the first columns and the days in the later ones. The yellow boxes show what is the best day to plan the amount of salad containers to fill on this filling line. These days have been decided to be the best days based on THT and ingredient delivery based on experience, and are put as fixed boxes by the employees. The amount in grey is based on the forecast, historical data, and information from the sales department and is called firm planned orders. Firm-planned orders can still be changed until the production planning is made final. Furthermore, the planner could deviate from the day and amount by making decisions based on his or her experience, but it does give a good guideline. Besides all this, the articles are ranked from top to bottom by priority of filling. This means that article 108634 is filled first on that day, and then 109634 and so on when this article is planned on this day.

Figure 2.10 shows what it looks like when the planning is filled in for line 6 in week 9. Here the planner made his or her own decisions for when to produce certain amounts on certain days. The planning system only indicates what has been decided by the planning department to be a good option, however, the planner can decide him- or herself if another option is better based on their experience.

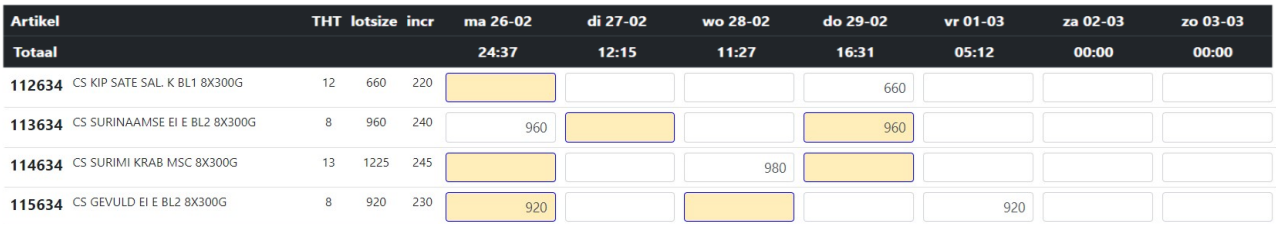


Figure 2.10: Weekly planning filled in in JAS.

Figure 2.11 shows what it looks like when the planning is made final. The amount becomes yellow and is transferred to shop orders, which means the ingredients can be ordered and the salad mixtures can be made by Kitchen 1. The filling lines fill the plastic containers indicated by the planning, and Kitchen 1 makes the salad mixtures just before they are needed by the filling lines.

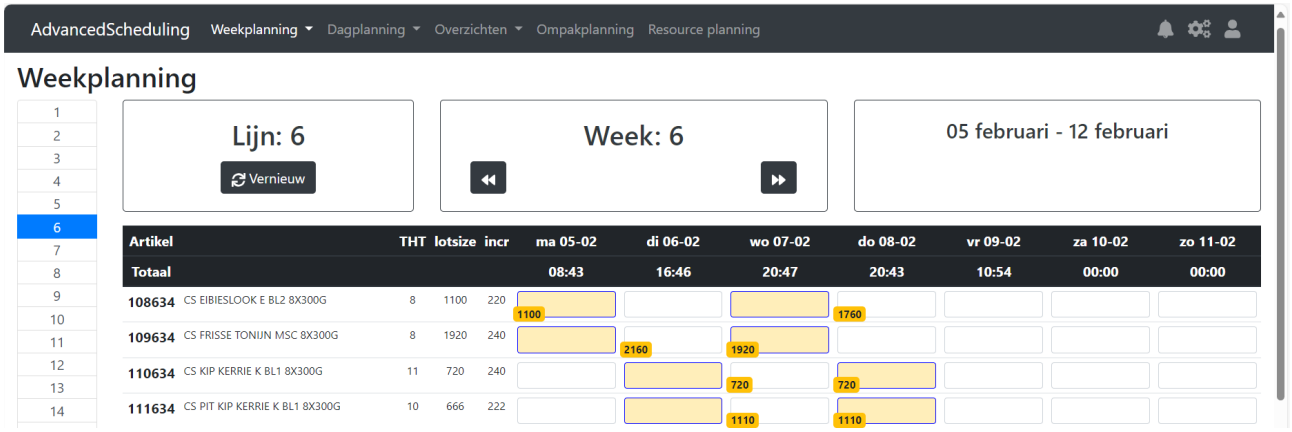


Figure 2.11: Weekly planning final planning in JAS.

Figure 2.12 shows the daily planning for the filling lines in the software system. The daily planning shows which recipes are filled in sequence on a certain day. This sequence is automatically generated from the weekly planning based on the order of the jobs from top to bottom. This first column shows the type of order (S = shop order, F = firm planned order), article number, article name with extra information, recipe number, amount per run, amount of to-be-filled plastic containers, total runs, THT (best before date), priority number, start time, total filling time, changeover time, break time, and end time.

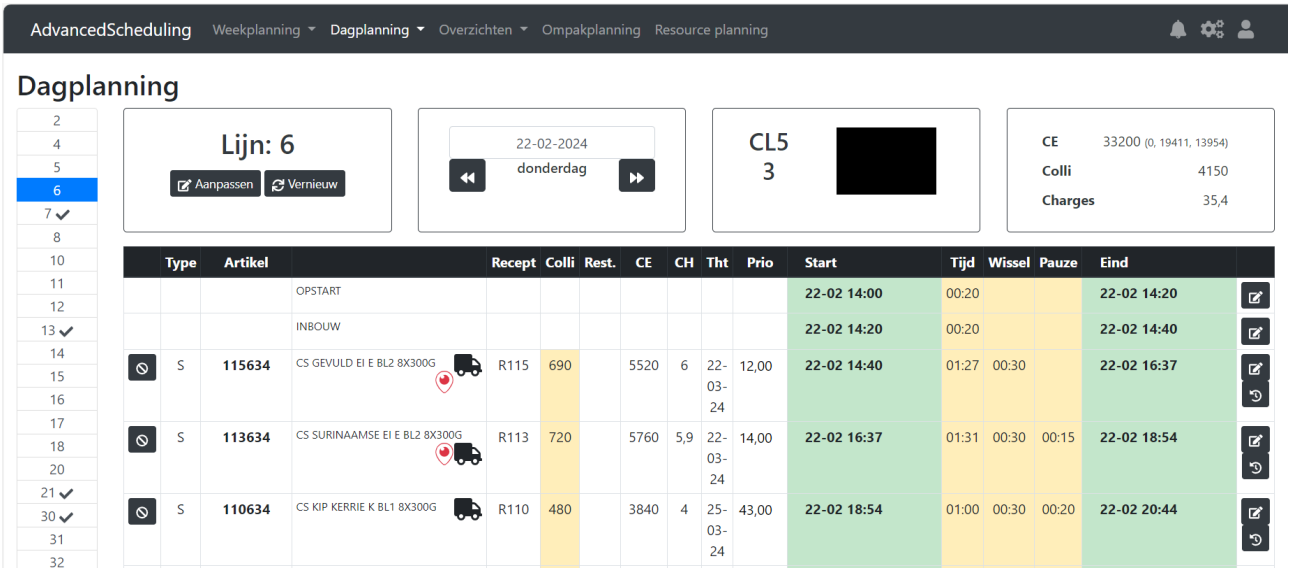


Figure 2.12: Daily planning filling lines in JAS.

Figure 2.13 shows the daily planning for Kitchen 1 in the software system. The daily planning for Kitchen 1 shows which recipes have to be finished at a certain time (prepared and mixed) in sequence. This sequence is automatically generated from the daily planning of the salads on the corresponding filling lines. The columns show the recipe number, the number of portions (one funnel consists of two portions) to produce, the corresponding filling line, the needed time (also includes setup time like cleaning), the recipe name, the number of kilograms, the number of charges, the mixing department, the type of order, and inventory.

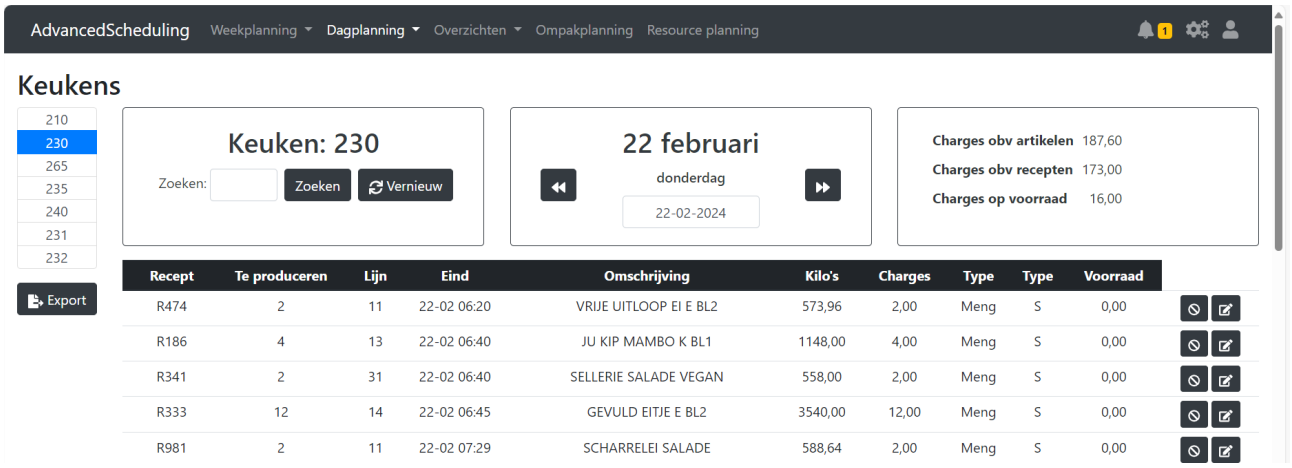
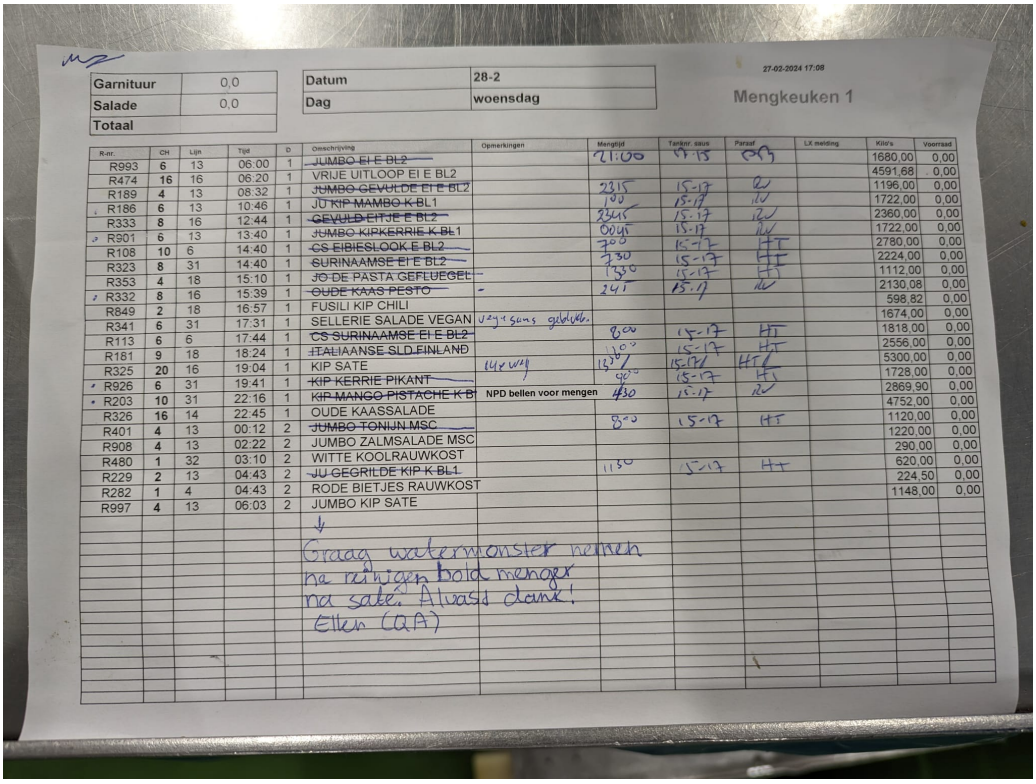


Figure 2.13: Daily planning Kitchen 1 in JAS.

Kitchen 1 receives a paper with the sequence of the day, and they can see the sequence on several computer screens in the mixing and preparation department. The operators can also see how it is going at the filling lines in real-time but do not use this that often. The list from Figure 2.14 is used mostly together with the recipe information from the company data, and the oral information from the filling lines.

2.4.2 Lower-level production scheduling

The operational production scheduling horizon is one to two days. There is not a well-thought-through approach on how to schedule the activities within Kitchen 1. Currently, the head operator (C-operator) of Kitchen 1 tells the employees what to do and when to do this. This is based on a sequence of recipes that the C-operator receives at the start of the shift (see Figure 2.14).



Garnituur	0,0
Salade	0,0
Totaal	

Datum	28-2
Dag	woensdag

27-02-2024 17:08

Mengkeuken 1

R-nr.	CH	Lijn	Tijd	D	Omschrijving	Opmerkingen	Mengtijd	Tankinr. saus	Parsel	LX melding	Kilo's	Voorraad
R993	6	13	06:00	1	~~JUMBO EI E BL2~~		21:00	15	803		1680,00	0,00
R474	16	16	06:20	1	VRIJE UITLOOP EI E BL2						4591,68	0,00
R189	4	13	08:32	1	~~JUMBO GEVULDE EI E BL2~~		2315	15-17	2		1196,00	0,00
R186	6	13	10:46	1	~~JU KIP MAMBO K BL1~~		100	15-17	1V		1722,00	0,00
R333	8	16	12:44	1	~~GEVULD EITJE E BL2~~		3545	15-17	1V		2360,00	0,00
R901	6	13	13:40	1	~~JUMBO KIPKERRIE K BL1~~		0045	15-17	1V		1722,00	0,00
R108	10	6	14:40	1	~~CS EIBIESLOOK E BL2~~		700	15-17	HT		2780,00	0,00
R323	8	31	14:40	1	~~SURINAAMSE EI E BL2~~		730	15-17	HT		2224,00	0,00
R353	4	18	15:10	1	~~JO DE PASTA GEFLUEGEL~~		1330	15-17	HT		1112,00	0,00
R332	8	16	15:39	1	~~OUDE KAAS PESTO~~	-	241	15-17	1V		2130,08	0,00
R849	2	18	16:57	1	FUSILI KIP CHILI						598,82	0,00
R341	6	31	17:31	1	SELLERIE SALADE VEGAN	vega sams gebakk.					1674,00	0,00
R113	6	6	17:44	1	~~CS SURINAAMSE EI E BL2~~		800	15-17	HT		1818,00	0,00
R181	9	18	18:24	1	~~ITALIAANSE SLD FINLAND~~		900	15-17	HT		2556,00	0,00
R325	20	16	19:04	1	KIP SATE	14x wok	1530	15-17	HT		5300,00	0,00
R926	6	31	19:41	1	~~KIP KERRIE PIKANT~~		900	15-17	HT		1728,00	0,00
R203	10	31	22:16	1	~~KIP MANGO PISTACHE K BL~~	NPD bellen voor mengen	430	15-17	2V		2869,90	0,00
R326	16	14	22:45	1	OUDE KAASSALADE						4752,00	0,00
R401	4	13	00:12	2	~~JUMBO TONIJN MSC~~		800	15-17	HT		1120,00	0,00
R908	4	13	02:22	2	JUMBO ZALMSALADE MSC						1220,00	0,00
R480	1	32	03:10	2	WITTE KOOLRAUWKOST						290,00	0,00
R229	2	13	04:43	2	~~JU GEGRILDE KIP K BL1~~		1150	15-17	HT		620,00	0,00
R282	1	4	04:43	2	RODE BIETJES RAUWKOST						224,50	0,00
R997	4	13	06:03	2	JUMBO KIP SATE						1148,00	0,00

Graag watermonster nemen na rijnigen bold menger na sate. Alvast dank! Elken (CA)

Figure 2.14: Recipe schedule for Kitchen 1 of the day.

The only indication the C-operator has is the end time (the time the salad mixtures should be available for the filling lines). With this information, it is very hard, and in most cases impossible, to know when a certain operation has to start within the preparation department, as there is no schedule for preparation and mixing operations. This leads to a sub-optimal sequence of operations being executed and time losses as the C-operator has to continuously look at the basic schedule on paper and tell everyone what to do. Besides the low availability of information on when to do what operation, the visual representations of the recipes are not intuitive. It is a basic list of all recipes with amounts that have to be made for all the corresponding filling lines in a sequence of ascending completion times (so the C-operator knows which recipe has to be finished first etc.). Every recipe takes a different processing time and thus a recipe that has to be completed earlier based on the list is sometimes not a recipe the employees at the preparation department should start with. For example, when the schedule on paper indicates that recipe 1 should be finished at 9:10 for filling line 6 and this recipe takes 10 minutes, and recipe 2 should be finished at 9:15 for filling line 11 and this recipe takes 20 minutes then you should start with recipe 2, but because there is no indication of processing times the employees start with recipe 1 as this is first on the list. All in all, the operational scheduling of jobs within Kitchen 1 does not take all necessary information into account and is based on gut feeling and experience. In this research, we look into this problem and develop a proper scheduling method to make it easier for Kitchen 1 to manufacture the salad mixtures and thus reduce the idle time at the filling lines.

2.5 Important factors

Many factors have to be taken into account during this research. The following ones are important:

- **Setups:**
  - Besides all the main production processes, cleaning is a time-consuming activity in between the processing of ingredients. The salad recipes are all different and contain different types of ingredients. These ingredients can contain allergens, which can cause harm to the end consumer. Figure 2.15 shows the possible allergens in Kitchen 1. Because of this, Johma has a high-quality standard and must prevent cross-contamination at all costs. This is why the machines including the mixers and equipment must be cleaned thoroughly when cross-contamination can occur. This happens every time an ingredient from a different recipe is processed on a machine, and when a different recipe is mixed with the mixer. Cleaning takes a lot of time but is an essential step to prevent cross-contamination and ensure quality. Furthermore, at the end of every shift, the processing of ingredients and mixing stops to clean everything for the next shift. Because of this, the start of new activities can only happen 30 minutes before the end of a shift, after this, a new activity is not initiated anymore.

Figure 2.15: Allergens in recipes at Kitchen 1 (12 of the 14).

- **Shifts:**
  - There are three shifts per day; 22:00-06:00, 6:00-14:00, and 14:00-22:00.
  - On Sunday, the production starts at 22:00 and on Friday, the production stops at 22:00. This means that there are 21 shifts in a production week.
- **Breaks and shift changes:**
  - Breaks take place at 8:30, 11:30, 16:30, 19:30, 01:00, and 04:00. Every break is 20 minutes long, however walking time should be added as Kitchen 1 is some distance away from the canteen. Because of this, 30 minutes is regarded as break time.
  - Shift changes happen in between every shift and take 10-15 minutes each when a shift starts to share information.
- **Machine and employee capacity:**
  - The number of operators is held fixed to make a feasible schedule of activities. Two operators are needed in the mixing department and four operators in the preparation department.
  - There are two large mixers and one small mixer. The small mixer is only used for tasting, so we leave it out of this research.
  - The number of carts and funnels is regarded as infinite for this research.
- **Ingredient availability:**
  - Within this research, all ingredients are immediately available to be used in Kitchen 1. Only transport time is considered. In reality, some ingredients can still be frozen, have to be waited for, or are not available at all.
- **Breakdowns:**
  - The breakdowns of all the machines in Kitchen 1 are not regarded as there is no information available, and because it does not happen that frequently compared to the other problems.

2.6 Conclusion

This chapter has outlined a comprehensive context analysis of the operational processes within Kitchen 1 and its corresponding filling lines at Johma, focusing primarily on the production of salads, referred to as bread and toast products in our analysis. Through a detailed overview of the different stages of production from ingredient preparation to the eventual filling of salads into containers, this study has highlighted several key aspects within the production chain that are critical for understanding and improving operational efficiency in Kitchen 1.

First, the complexity of the operational planning processes at Johma is highlighted by the variety of recipes and the corresponding variability in the processing requirements. With 91 distinct salad recipes and 114 unique ingredients involved, the scheduling and execution of production tasks require thorough planning and flexibility. This complexity is further increased by the strict cleaning operations required to prevent cross-contamination, particularly given the allergens involved in the recipes.

Secondly, the Pull system within the internal production flow between Kitchen 1 and the filling lines ensures that production is directly tied to demand, minimizing wastage and optimizing resource utilization. However, the current production planning and scheduling methods, particularly in Kitchen 1, reveal a need for manual coordination and experience rather than a sophisticated scheduling tool. The dependency on gut feeling and experience poses a risk to operational efficiency, especially under complex job scheduling scenarios.

As we jump into the next chapters, the insights gained from this context analysis guide the development of production scheduling practices. Emphasizing the implementation of more sophisticated scheduling methods, that do not only take into account the amount that has to be planned and some pre-determined rules based on experience, could mitigate some of the challenges identified, leading to improved operational efficiency. The following chapters explore these potential improvements in detail, aiming to propose a solution approach suitable for Johma's situation.

3 Literature review

This chapter provides a detailed literature study, which is needed to form the theoretical basis for developing a solution approach for Johma. In this chapter, we answer the following research question:

'What is currently known in the literature about production scheduling to increase production output and flow, and decrease idle times?'

Section 3.1 explains Johma's production environment and the theoretical systems that are working within it. Section 3.2 elaborates on the classification of a machine environment and the characteristics and objectives of scheduling problems. Furthermore, Section 3.3 explains more about the complexity of the scheduling problem Johma faces, and Section 3.4 describes several solution approaches to solve the classified scheduling problem. Finally, Section 3.5 provides the conclusion of this chapter.

3.1 Production environment

This section provides theoretical information about Johma's production environment to help guide the literature review and to give further context. Sections 3.1.1, 3.1.2, and 3.1.3 explain the three production system concepts, Customer Order Decoupling Point (CODP), Push vs Pull, and Planning vs Scheduling respectively.

3.1.1 Customer order decoupling points

To ensure the proposed solution fits the business objectives, the type of production system must be determined. Production systems can be classified based on the way production is realised, e.g. MTS, MTO, Assembly-To-Order (ATO) or Engineer-To-Order (ETO) (Olhager, 2010). The CODP determines which one of the above-mentioned systems is applicable. Figure 3.1 shows how to determine this with an example.

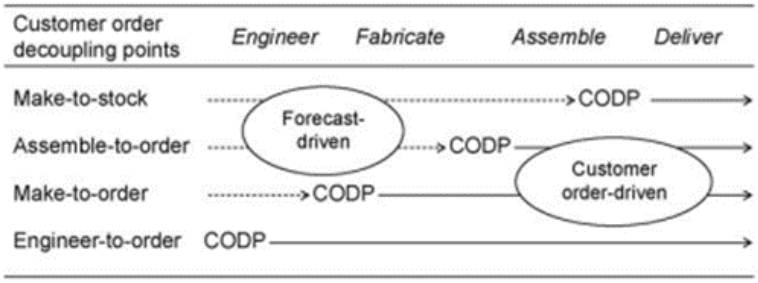


Figure 3.1: Customer order decoupling points (Olhager, 2010).

The CODP of Johma's production system lies between assemble and deliver stages and between engineer and fabricate stages. Johma's production system produces salads based on expected sales and adjusts this when an order from a client comes in. The production system of Johma's is a combination of a MTS and a MTO system as salads are made to certain inventory levels (stock), but also made to order. As explained in Section 2.1, some clients order the precise amount and some only give an indication, and thus Johma makes these products to stock.

3.1.2 Push vs pull

Besides the CODP, there is another concept that describes a production system. This concept is more relevant for internal processes within a production system. The terms Push and Pull can be used to describe these internal production processes (Benton, 2011). A Push system pushes the finished work further up the internal supply chain without knowing if the next station is ready to receive it, and within a Pull system the work is pulled by the following workstation as it is ready to receive it. Within Johma the filling lines operate as a Push system (pushes finished work into inventory), and Kitchen 1 operates as a Pull system (work is pulled by the filling lines).

3.1.3 Planning and scheduling

Section 2.4 explained that the higher-level production is planned based on the planning of salads (final products) and that the lower-level production is scheduled based on that higher-level planning. Literature shows that there is a clear distinction between planning and scheduling. Figure 3.1 shows how to distinguish between planning and scheduling based on McKay and Wiers (2003).

Table 3.1: Distinctions between planning and scheduling (based on McKay & Wiers, 2003).

	Planning	Scheduling
Horizon	Multiple, bucketed	Single, real-time
Availability of information	High	Low
Interaction with lower level	Low	High
Time pressure	Low	High
Input driven by	Expectations of the future	Reality
Output represented by	Volumes/bucket	Orders, assignments, or job/real-time
Autonomy	High	Limited
Transparency	Lower	Higher
Level of support	High	Intermediate
Information representation	Multiple buckets in a table	Gantt Chart

According to McKay and Wiers (2003), a distinction can be made between several factors. Section 2.4 explained that Kitchen 1 has a scheduling horizon of one to two days. Decisions that are made in this department are according to the types of constraints involved at any given moment during the production activities. Furthermore, limited information is available and time pressure is high. The autonomy of Kitchen 1 is also limited as the department is constantly interacting with others, and thus transparency must be high. This shows that the problem in Kitchen 1 is a typical scheduling problem, which is focused on the scheduling of operational activities. In Chapter 4 we develop a model that could present the scheduling of jobs and operations with a Gantt Chart, which is preferred.

3.2 Taxonomy

Section 3.2.1 provides the classification of our scheduling problem in Kitchen 1. Section 3.2.2 dives deeper into the characteristics and constraints of these types of scheduling problems.

3.2.1 Scheduling problem classification

There are many different types of scheduling problems known in the literature. Graham, Lawler, Lenstra, and Kan (1979), which is one of the first papers on this topic, and Allahverdi, Gupta, and Aldowaisan (1999) use the notation $\alpha/\beta/\gamma$ to classify these types of scheduling problems. α describes the machine environment, β provides the job characteristics and constraints, and γ contains the optimality criterion which is the objective of the model. The following machine environments (α) are identified:

Single machine
Single machine problems are characterized by jobs requiring a single available resource (machine), where the job is performed by the resource one at a time (Perez-Gonzalez & Framiñan, 2018). Each job has different properties, like processing time, setup time, and a due date. The job sequences can differ depending on certain criteria.

Parallel machines
In some cases there is more than one machine available for performing the same job; this is called a parallel machine problem. Arriving jobs can be processed by one or more of the available machines. Jobs require only one operation and the operations can be performed by any of the available machines. The job sequence is determined by several important criteria. There are identical, uniform, and unrelated parallel machine problems.

- *Identical:* Involves jobs where each machine has the same processing speeds and specifications (Kim, Song, & Jeong, 2020).
- *Uniform:* Each machine could have different processing speeds and specifications, however, these differences are proportional (Kim et al., 2020).
- *Unrelated:* Each job's processing time can differ from machine to machine, and jobs can be done on any of the machines (Fanjul-Peyro, Ruiz, & Perea, 2018).

Open shop
In an open shop environment, a job may be processed by the set of available machines in any sequence the job needs (Naderi & Zandieh, 2014).

Flow shop
In a flow shop with m machines, arriving jobs need to be processed on m machines at certain stages in the same technological order. Each job has o operations and operation i of each job is to be performed on machine m. The processing times for each job can differ on the different machines. There are "normal", flexible, and hybrid flow shops. The "normal" flow shop is a flow shop like described already. The flexible and hybrid flow shops add different elements to this "normal" flow shop.

- *Flexible flow shop:* This type of flow shop extends by having multiple parallel machines at at least one of the stages (Lee & Loong, 2019). This allows for alternative processing routes and greater flexibility for scheduling.
- *Hybrid flow shop:* A hybrid flow shop adds complexity and flexibility to the normal flow shop structure by incorporating multiple parallel machines at one or more stages (Lei & Guo, 2016). The difference with the flexible flow shop is that a job may skip certain stages depending on what is needed for the job, but should be processed by at least one stage.

Job shop
A job shop environment consists of different machines and an arriving job may require some or all of the machines in some specific order. A job cannot use the same machine more than once. There are "normal" and flexible job shops.

- *Flexible job shop:* This type of job shop allows operations to be processed on any machine from a given set of machines that can perform this operation (Cinar, Topcu, & Oliveira, 2015).

The scheduling problem in Kitchen 1 can be seen as a Flexible Job Shop Scheduling Problem (FJSP), as the jobs, in our case the salad recipes, require a distinct set of operations (unique sequences). This is different from the flow shop because, within a flow shop, the order of operations is fixed. Within our problem, each job has a unique set of operations which can also have a different processing sequence. The difference with the classical job shop is that the FJSP contains an additional problem, i.e., assigning operations to machines (Ziaee, 2013). The following section provides more information on the β and γ fields of the FJSP.

3.2.2 Characteristics and objectives

In recent decades, the complexity of scheduling problems, specifically the Flexible Job Shop (FJS), has increased the need for comprehensive frameworks to understand these problems better. As already mentioned in Section 3.2.1 we classified our scheduling problem as a FJS. This section dives deeper into the characteristics (β) and objectives (γ) that are attached to FJSPs.

Graham et al. (1979), are one of the first, if not the first to use the $\alpha/\beta/\gamma$ notation, laying the foundation for classifying deterministic scheduling problems. The use of this notation has been widely extended by other researchers to address the attributes of the FJS, to provide more relevant information for real-world manufacturing and production environments.

Allahverdi et al. (1999), extend the β classification by including setup times, machine eligibility, and buffer capacities. Their work also adds more γ objectives such as makespan, flowtime, and tardiness minimisation.

More recent contributions from Cinar et al. (2015) & Pinedo (2016) & Perez-Gonzalez and Framinan (2014), have continued to refine the β and γ classifications, by using attributes to more modern-day problem instances. This shows that the FJSP is continuously becoming more complex, and frameworks need to be adapted to the new information from the literature.

By combining the attributes from all the relevant papers we propose a taxonomy for the FJSP, and also provide the classification of information from the papers and of our problem. Figure 3.2 provides the taxonomy for classifying the FJSP.

Job characteristics (β)		
Time considerations	processing times	Machine/ operator dependent
		Operation dependent
		Job dependent
		Fixed
	Transport/move times	Job / operation dependent
		Fixed
		Part of processing time
		Negligible or none
	Release times/dates	Job dependent
		Machine dependent
		Operation dependent
		Frequency dependent
		Independent
		Ready at time zero
	Due dates	Job dependent
		Negligible or none
	Setups/ changeovers	Sequence dependent
		Sequence independent
		Machine dependent
		Part of processing time
		Negligible or none
	Batch	Parallel
		Serial
		None
Machine	Exclusivity	Exclusive
	Availability	Always available
		Forbidden intervals
	Maintenance	Fixed
		Variable
		No planned
	Machine breakdowns	Based on distribution
		Random
		Fixed
		None
	Operator-to-machine	Flexibility
		Fixed
		None

Other		
Other	Job precedence	
	Operation precedence	
	No preemption	
	Job families	
	Eligibility	
	Blocking	
	No-wait	
	Recirculation/ reocurring	
	Other	

Optimality criterion (γ)	
Objective	Completion time
	Cost / unit penalty
	Makespan
	Setup changeover cost
	Setup changeover time
	Flowtime
	Lateness
	Earliness
	Tardiness
	Total workload
	Critical machine workload
	Due date
	Other

Figure 3.2: Taxonomy framework for β and γ classification.

3.2.3 Classification based on taxonomy

The machine environment (α) belongs to the FJS class. This system diverges from a conventional job shop due to having more unique routing options for jobs (recipes), having unique sequences for the operations of jobs, and having a unique set of machines assigned per job. Each job, in our case a recipe, has its unique route to following through the shop, Kitchen 1. Certain operations can be processed in parallel (more non-identical machines to perform the operations that are allowed to be processed simultaneously) and some need to be processed in series (identical machine(s) to perform the operations that cannot be processed simultaneously). Chapter 4 explains this in more detail for the model design.

Regarding the job characteristics (β), processing times are machine and operation-dependent, known in advance, and fixed per type of machine in combination with a certain operation. Transportation of an ingredient from a different department towards Kitchen 1 is substantial and thus relevant to incorporate. However, internal transport in Kitchen 1 can be neglected, as these transportation times are incorporated into the processing time or are very small. Release dates are zero as all jobs can be processed from time zero, due dates are job-dependent, and setups depend on the sequence of operations and the type of machine. This is because certain operations can be executed immediately after each other without a setup and for some, a setup needs to happen before a new operation can be executed. Each machine can process only one job at a time and each operation can be processed on only one machine at a time. Furthermore, at certain time intervals (breaks and shift changes) no jobs can be scheduled, and machines are assumed to not have breakdowns or maintenance. Besides machines, operators should also be assigned to botch operations and machines. Furthermore, there is no blocking as there is enough space (buffer capacity) for finished operations to wait for the following operation. Moreover, operations from a job must be processed in a specific order which is the precedence, operations are only eligible for a certain set of machines, and certain operations of a job can be processed simultaneously on different types of machines. No preemption is allowed, which means that once an operation has started it cannot be interrupted until finished.

The optimality criterion (γ), also known as the objective function, for our problem, is to minimise the idle time at the filling lines. This can be transformed into minimising total tardiness, which is the amount of time by which a job, in our case a certain recipe, exceeds its due date (the time it is needed at the corresponding filling line). To give different importance to the different recipes based on the importance of the recipe, weights are given. The objective is thus minimising the total weighted tardiness.

Based on this classification we can conclude that our scheduling problem is a FJc$|prec, S_{i_j}, M_{i_j}, W_m, UWP_u^{m,w}|\sum w_i Tar_i$. This means that it is a flexible job shop scheduling problem with precedence, sequence-dependent setups, machine and worker eligibility, and forbidden work intervals, where the objective is the total weighted tardiness.

3.2.4 Benchmarking our problem

In this section, the FJSPs in the literature are compared to our problem to see how the different problems relate to each other. Identifying commonalities, differences and gaps between the literature and our problem is the aim of this section.

Much research has been executed to review FJS problems in the literature. Chaudhry and Khan (2013), Perez-Gonzalez and Framinan (2014), Cinar et al. (2015), and X. Li, Xie, Peng, Li, and Gao (2019) have reviewed a lot of relevant FJS literature over the past decades. We use these reviews to identify several papers that cover a similar FJSP as what we have. We continuously look into related papers to find relevant other papers to compare with our problem, and we are confident that all aspects of our problem are found in the collected papers. Figure 3.3 provides a classification table by using the taxonomy table from Figure 3.2.

			our problem	Saidi-Mehrabad & Fattahi, 2006	Ganesan & Sivakumar, 2006	Fattahi et al., 2009	Yazdani et al., 2010	Li et al, 2010	Zhang et al., 2012	Dalfard & Mohammadi, 2012	Na & Park, 2013	Mousakhani, 2013	Sobeyko & Mönch, 2016	Lei & Tan, 2016	Paksi & Ma'ruf, 2016	Gong et al., 2017	Shen et al., 2018	Gong et al., 2018	Deng et al., 2019	Kress et al., 2019
Job characteristics (β)																				
Time considerations	processing times	Machine/ operator dependent	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
		Operation dependent	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
		Job dependent																		
		Fixed																		
	Transport/move times	Job / operation dependent							x											
		Fixed																		
		Part of processing time	x											x	x					
		Negligible or none	x	x	x	x	x	x		x	x	x	x	x		x	x	x	x	x
	Release times/dates	Job dependent								x										
		Machine dependent																		
		Operation dependent																		
		Frequency dependent																		
		Independent																		
		Ready at time zero	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x
	Due dates	Job dependent	x							x	x	x	x	x	x					
		Negligible or none		x	x	x	x	x	x							x	x	x	x	x
	Setups/ changeovers	Sequence dependent	x	x													x			
		Sequence independent																		
		Machine dependent	x	x								x								
		Part of processing time			x										x					
		Negligible or none			x	x	x	x	x	x	x		x	x		x		x	x	x
	Batch	Parallel																		
		Serial																		
		None	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x	x
Machine	Exclusivity	Exclusive	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	Availability	Always available		x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x
		Forbidden intervals	x							x										
	Maintenance	Fixed								x										
		Variable																		
		No planned	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x
	Machine breakdowns	Based on distribution																		
		Random																		
		Fixed								x										
		None	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x
	Operator-to-machine	Flexibility	x											x	x	x		x		x
		Fixed																		
		None		x	x	x	x	x	x	x	x	x	x				x		x	
	Other	Job precedence																		
		Operation precedence	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x
		No preemption	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
		Job families																		
		Eligibility	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
		Blocking																		
		No-wait																	x	
		Recirculation/ reocurring																		
		Other				x														
Optimality criterion (γ)																				
Objective		Completion time			x															
		Cost / unit penalty																x		
		Makespan		x		x	x	x	x	x				x		x	x	x		x
		Setup changeover time																		
		Flowtime			x														x	
		Lateness																		
		Earliness																		
		Tardiness	x							x	x	x	x	x	x					
		Total workload														x				
		Critical machine workload														x				
		Other							x	x								x		

Figure 3.3: Comparison between our problem and problems from the literature.

Figure 3.3 shows that our problem has not been researched in one paper as a whole, but several parts of the problem are covered in a combination of several papers. Job characteristics specific to our problem like processing times, transportation times, release times, batch, exclusivity, maintenance, breakdowns, precedence, preemption, and eligibility have been thoroughly researched by almost all papers. Characteristics specific to our problem that are less researched are operator-to-machine flexibility and job-dependent due dates. However, these are still covered in several papers. One characteristic that is researched very few is the availability of working time. Only one paper by Dalfard and Mohammadi (2012) covers a similar approach by having intervals for completing maintenance. This is different from the forbidden time intervals for our problem but can be helpful as a source of inspiration.

Because of the aforementioned points, there is a gap in the literature, which is that there is not yet a problem as a whole like ours researched before. We fill this gap by using the elements from all the papers and our expertise to form a solution approach for our unique FJSP.

3.3 Additional research

Within this section some additional found literature is explained as this is critical for developing the model and choosing the right solution approach. Section 3.3.1 explains the NP-hardness of our scheduling problem, and Section 3.3.2 explains in what way the scheduling should be performed; either backward or forward.

3.3.1 Computational complexity

Saidi-Mehrabad and Fattahi (2006), state that the general FJSP is strongly **NP-hard**. The job shop problem from Ganesan and Sivakumar (2006) and many others is stated to be NP-hard. Furthermore, Figure 3.4 shows that the flexible job shop (which is part of the job shop class) with the objective function to minimize the makespan ($C_{\max}$, which is less complex than minimising weighted tardiness ($\sum w_i * Tar_i$)) is already an NP-hard problem and that instances with only 40 jobs can be solved within polynomial time (Pinedo, 2016). All elements from their problems are also in our problem with even larger problem instances of 70+ jobs, thus making it also NP-hard. NP-hard stands for non-deterministic polynomial-time hard. This means that the problem is so complex, that it cannot be solved exactly within polynomial time. Finding a good solution within a reasonable amount of time can be done with the help of heuristics. Section 3.4.3 and 3.4.4 explain this further.

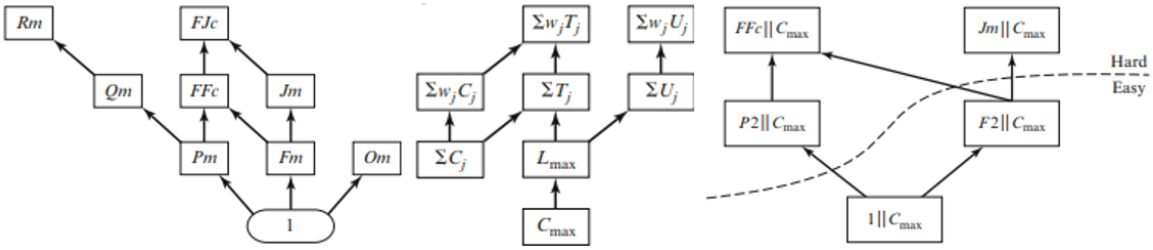


Figure 3.4: Complexity of scheduling problems (based on Pinedo, 2016).

3.3.2 Forward and backward scheduling

The way of scheduling affects the outcome of the solution drastically. Therefore, it is an important choice when deciding what solution approach to use. The two main scheduling approaches are forward and backward scheduling (Kalinowski, Grabowik, Cwikla, Paprocka, & Balon, 2018). Forward scheduling consists of inserting jobs and operations into the schedule based on the first to last operation sequence. Thus taking into account the precedence relationships. Backward scheduling involves inserting the jobs and operations in reverse order, from the last to the first operation, starting from the job's due date. The latest possible start time for a job is obtained from this. According to Kalinowski et al. (2018), **forward** scheduling is beneficial when jobs have a certain release date, thus certain jobs can only start *after* a certain time. In contrast, **backward** scheduling is better when jobs have to be finished at a certain due date or earlier, thus certain jobs have to be finished *before* at a certain time. We can conclude that backward scheduling is more beneficial for our scheduling problem as our goal is to obtain the start times of jobs and operations that must be finished before the job's due date. Although backward scheduling is the best based on the literature, we compare both methods in Chapter 5 to see what conclusions we can make.

3.3.3 Lexicographic optimisation

Our main scheduling objective is minimising the weighted tardiness, which we will also refer to in this report as the main objective. With this we want to minimise the idle time of jobs. However, there is a phenomenon called lexicographic optimisation which first optimises a first objective and after this optimises a second one (Bissoli, Zufferey, & Amaral, 2019). In our case, we first want to optimise minimal weighted tardiness and when this is 0 or we find a solution with the same tardiness, we want to look at minimising the makespan. The makespan is not the most important so this is the second objective. With this we can find even better solutions that has the same minimal tardiness, but potentially a lower makespan. In Chapter 4 we use the lexicographic optimisation approach in the exact model and in the heuristics.

3.4 Solution approaches

This section aims to identify appropriate solution approaches for our scheduling problem classified in Section 3.2. In the past decades, much research has been done on how to solve the FJSP. Almost all papers consider the same two methods; exact and heuristic. Moreover, many researchers split the heuristic method into constructive heuristics, and improvement heuristics, also known as meta-heuristics. As already explained in Section 3.3 our problem is NP-hard and thus an exact approach is not suitable as a solution approach. However, we still formulate an exact model in Chapter 4 to increase the understanding of the problem. Section 3.4.1 explains what knowledge is needed before a solution approach can be chosen. Section 3.4.2 explains the potential exact approach for smaller problem instances, Section 3.4.3 describes how to use the constructive heuristics, and Section 3.4.4 elaborates on the improvement heuristics. At last, Section 3.4.5 dives deeper into neighbourhood operators used within improvement heuristics.

3.4.1 Choosing the right solution approach

Before searching for relevant literature for potential solution approaches, certain performance indicators based on Johma's needs must be taken into account. Johma should be able to work with the developed solution approach. Furthermore, the solution approach should be easy to implement as it must be able to be integrated into the daily operations of Johma. No one at Johma has any experience with algorithms or other sophisticated scheduling optimisation methods. However, we do want to make sure the solution is implemented and used. Therefore the following performance indicators for our solution approach are constructed:

- ***Accessible:*** Easy enough for both production managers and operators to grasp.
- ***Interactive:*** Capable of enabling user engagement.
- ***Flexible:*** General enough to facilitate effortless adjustments, such as correcting wrong input settings.
- ***Forward-looking:*** Open to future enhancements and developments.
- ***Efficient:*** Capable of providing satisfactory solutions without excessive computational effort.

With these performance indicators in mind, relevant literature is found. Figure 3.5 shows the literature used for finding relevant solution approaches, specifically for the improvement heuristics.

	Our problem	Saidi-Mehrabad et al., 2006	Ganesan & Sivakumar, 2006	Fattahi et al., 2009	Yazdani et al., 2010	Li et al, 2010	Zhang et al., 2012	Dalfard & Mohammadi, 2012	Na & Park, 2013	Mousakhani, 2013	Sobeyko & Mönch, 2016	Lei & Tan, 2016	Paksi & Ma'ruf, 2016	Gong et al., 2017	Shen et al., 2018	Gong et al., 2018	Deng et al., 2019	Kress et al., 2019
Objective functions																		
Tardiness	x							x	x	x	x	x	x					
Makespan		x		x	x	x	x	x				x		x	x	x		x
Other			x					x						x		x	x	
Important constraints																		
Every operation should be scheduled on a machine	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Machine eligibility	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Operator to machine assignment	x											x	x	x		x		x
Precedence	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x
Only one operation per machine per time interval	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Setups	x	x								x					x			x
No preemption	x	x	x	x	x	x	x	x	x	x	x	x	x		x	x	x	x
Available working time	x							x										
Other							x	x			x							
Solution approaches																		
Iterative Local Search (ILS)	x									x	x	x						
Simulated Annealing (SA)	x		x	x				x			x	x						
Tabu Search (TS)	x	x		x		x	x								x			
Genetic Algorithm (GA)							x	x	x				x			x		
Other					x						x			x			x	x

Figure 3.5: Classification of improvement heuristic solution approaches in the literature.

The figure shows that the papers cover all important constraints of our FJSP, and some papers cover more in total than others. The constructive heuristics used in these papers are almost all different. Because of this, Section 3.4.3 elaborates on the proper constructive heuristics for our problem. The improvement heuristics, however, are more similar. Table 3.5 shows that Iterative Local Search (ILS), Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithm (GA) are methods often used for comparable problems as our FJSP. Mousakhani (2013), state that ILS reports high performance and that it is simple to implement. Moreover, Sobeyko and Mönch (2016), conclude that ILS is a fast approach and that it can find high-quality solutions. When looking at SA, Dalfard and Mohammadi (2012), Ganesan and Sivakumar (2006), and Fattahi, Jolai, and Arkat (2009) explain that it is a technique that has produced good results for combinatorial optimisation problems. Saidi-Mehrabad and Fattahi (2006) use TS as it is widely applied for finding optimal solutions to combinatorial optimisation problems. In addition, Fattahi et al. (2009), and J. Li, Pan, Suganthan, and Chua (2010) use TS for testing the performance of the proposed algorithm. This shows good reliable results in a relatively short time. GA is also used as a good method in several papers but is a harder method to understand. Johma needs a method that is understandable as they do not know algorithms. Therefore, due to the need for an accessible solution approach, this is left out of this research. Other approaches that have been found, but not used often, are parallel variable neighbourhood search (Yazdani, Amiri, & Zandieh, 2010), memetic algorithm (Paksi & Ma'ruf, 2016), hybrid discrete group search (Deng, Zhang, Jiang, & Zhang, 2019), and lastly MIP and CP solvers (Kress & Müller, 2019). We exclude these from our research as they are minimally used for our type of problem. Section 3.4.4 covers the ILS, SA, and TS improvement heuristics, used in this research, in more detail, but first, the exact method is explained in short, and following this the constructive heuristics are elaborated upon in Section 3.4.3.

3.4.2 Exact

Exact methodologies, like linear or dynamic programming, grounded in mathematical precision, strive for the optimal solution, adhering strictly to problem constraints. These methods are particularly effective for smaller-scale problems, where computational resources can feasibly achieve the best possible outcome. Cinar et al. (2015) and Pinedo (2016), explain that the branch and bound algorithm and several decomposition approaches can be used to guarantee an optimal solution. As already stated in Section 3.3.1, our problem cannot be solved with an exact approach. However, to get a better understanding of the problem, we develop an exact mathematical formulation for our problem in Chapter 4, and solve it with a heuristic approach.

3.4.3 Constructive heuristics

Acknowledging the computational constraints in solving larger instances, as discussed in Section 3.3.1, heuristic strategies offer a pragmatic and efficient solution pathway. This section explains the constructive heuristic strategies and Section 3.4.4 elaborates on the improvement heuristics. The papers from Section 3.4.1 used all kinds of constructive heuristics. Because of this, we cannot determine which one would be the most suitable for our problem, thus with the use of different literature the right constructive heuristics are determined for our problem.

Constructive heuristics start without an initial schedule and construct a schedule by gradually adding one job at a time (Pinedo, 2016). According to Cinar et al. (2015), good solutions can be found for large-size instances in a convenient computational time, and the first development of heuristics was based on priority dispatching rules. Furthermore, dispatching rules are easy to implement and have a low time complexity, this is why it is so frequently used (Baykasoglu & Ozbakir, 2010). Table 3.2 shows some of these dispatching rules (Pinedo, 2016) & (Baykasoglu & Ozbakir, 2010).

Table 3.2: Description of Rules

Rule name	Objective
Service In Random Order (SIRO)	Optimizing anything
Earliest Release Date first (ERD)	Minimizing variance of waiting times
Earliest Due Date first (EDD)	Minimizing lateness
Minimum Slack first (MS)	Minimizing tardiness
Shortest Processing Time first (SPT)	Minimizing total completion time
Weighted Shortest Processing Time first (WSPT)	Minimizing total weighted completion time
Longest Processing Time (LPT)	Balancing loads on parallel machines
Shortest Processing Time – Longest Processing time (SPT-LPT)	Balancing the sequence of jobs
Critical Path (CP)	Minimizing makespan
Largest Number of Successors first (LNS)	Minimizing makespan
Shortest Setup Time first (SST)	Minimizing makespan
Least Flexible Job first (LFJ)	Minimizing makespan
Longest Alternative Processing Time first (LAPT)	Balancing workloads on parallel machines
Shortest Queue first (SQ)	Minimizing waiting times at machines
Shortest Queue at the Next Operation (SQNO)	Minimizing idleness on machines
Least Work Remaining Time (LWRT)	Minimizing makespan
Most Work Remaining Time (MWRT)	Balancing machine workload
Process time/ total remaining time (PDR)	Balancing makespan and machine workload
Apparent Tardiness Costs first (ATC)	Minimizing tardiness

The dispatching rules are mostly used for determining the sequence of operations of the different jobs. However, attention must also be given to the machine assignment for these operations. A wide range of constructive heuristics are developed over time, partially based on the dispatching rules, to help with the sequence of operations and the machine assignment (K. Gao, Suganthan, Tasgetiren, Pan, & Sun, 2015). The main benefit of a constructive heuristic, just as the dispatching rules, is that it saves a lot of time by providing a relatively good solution in little time (Ziaee, 2013). Many alterations and unique versions of constructive heuristics exist in the literature. The right constructive heuristic depends a lot on the objective, and the outcome may not be the best schedule.

For our objective, total weighted tardiness, two dispatching rules from Table 3.2 can be useful to form a constructive heuristic, namely Earliest Due Date (EDD) and Minimum Slack (MS). Apparent Tardiness Costs (ATC) cannot be used as it recalculates the costs over time, which is dynamic. We are interested in a static schedule, and not a dynamic one as Kitchen 1 needs to know the schedule at the start of the day. The dynamic characteristics to make the schedule more robust lie in the fast computational strength of the heuristic used, and thus being able to re-run the model in a reasonable quick time. With this, the schedule can be re-run during the day to adjust for changes, therefore taking into account the dynamic nature of real-world manufacturing. In addition, the Random rule is used besides EDD and MS. These rules are explained as follows:

- **EDD:** With the EDD first rule, jobs are prioritised based on their due dates, with those having the earliest due dates going first. This rule is aimed at minimising tardiness.

- **MS:** With the MS first rule, slack time is calculated, which is the difference between the job's due date and its needed processing time. Jobs with the least slack time are prioritised to minimise the risk of tardiness. In our case, we can use time zero until its due date as the remaining time. This rule is a little bit different than the EDD rule as jobs that require more processing time can potentially be scheduled earlier than jobs with an earlier due date.
- **Random:** With the random rule the jobs are sorted based on a random ratio. This dispatching rule can be used to get to an initial solution which is most likely sub-optimal. However, by using the Random rule, local minima can potentially be escaped by creating a very different initial solution, which can be used within an improvement heuristic to find better neighbouring solutions. This is further explained in the next section.

These three rules are used for making a constructive heuristic to form an initial solution. To reach a more optimal solution and thus a better schedule, improvement heuristics can help by searching in the neighbourhood of the initial solution, made by a constructive heuristic. Section 3.4.4 explains which relevant improvement heuristics can be used to tackle our FJSP. Section 4.3 in the next chapter explains the logic of the constructive heuristic in more detail to form an initial solution with the dispatching rules mentioned above.

3.4.4 Improvement heuristics

Improvement heuristics are different from constructive heuristics as these algorithms start with an initial schedule, possibly generated by a constructive heuristic, and iteratively refine this schedule. One important class of improvement heuristics are local search procedures (Pinedo, 2016). The idea is to not find an optimal solution (as this is not possible due to the NP-hard nature of the problem) but to try to find a schedule better than the current one. With an improvement heuristic, neighbouring solutions are found, and each iteration is a search for potentially better neighbours than the current one. The neighbour is either accepted or not, based on the acceptance-rejection criterion. The acceptance-rejection criterion significantly sets apart local search procedures in their design and operational essence.

Because of the performance indicators from Section 3.4.1 the GA improvement method is excluded from our research as this method is not accessible for production managers and operators, and is significantly harder to develop further in the future thus not being forward-looking. Johma wants a more pragmatic approach, which can be reached with ILS, SA, and TS. In the following three parts, these improvement heuristics are discussed.

Iterative Local Search (ILS) is a simple improvement heuristic that starts with an initial solution made with a constructive heuristic and iteratively explores the neighbourhood of this solution to find a better one. This is done with the use of local search operators, which are also known as neighbourhood operators. This is discussed in more detail in Section 3.4.5, but in short, these operators alter the current sequence of jobs or operations to form a neighbour solution. If the neighbour solution is better than the best found so far, then this becomes the new best solution. This continues until a stopping criterion is met. The main downfall of this method is that the solution space stays in a local optimum and that a better solution is not found. Figure 3.6 shows this phenomenon. The starting point (black dot) is found with a constructive heuristic, and better solutions are found, thus going down the curve, until no better solution can be found (red dot). The figure shows that if no worse solution is accepted (going up the curve to the right) with a certain acceptance criterion, it cannot escape the local optimum (red dot) to get to a global optimum (green dot), which is the best solution. SA and TS are methods that can escape a local optimum, and thus potentially reach a global optimum.

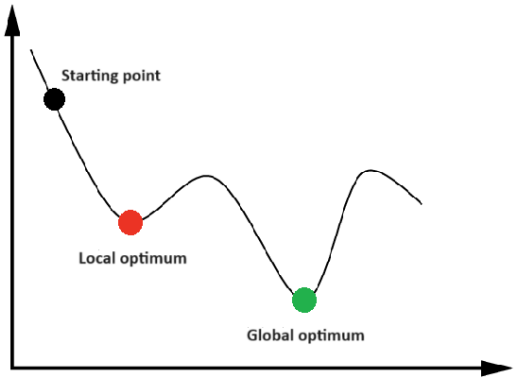


Figure 3.6: Showcasing of a local optimum.

Simulated Annealing (SA) is a procedure that goes through several iterations. Figure 3.7 shows the pseudo-code for SA. First, an initial solution is formed based on a constructive heuristic, and all parameters are initialized. At each iteration, a neighbouring solution is generated, which is done by altering the current solution slightly (for example done with operators from Section 3.4.5). For this new solution, the objective value is calculated and compared to the best solution found so far. The new solution is accepted when this solution is better than the best solution found so far. If the new solution is worse, the new solution can still be accepted with a certain probability. The probability is determined by the temperature, and this temperature is updated according to a cooling schedule. The cooling schedule determines how much the temperature decreases and thus reduces the probability of accepting a worse solution over time. With this, the solution space can be explored more extensively at higher temperatures and converges to reasonably good solutions at lower temperatures. This whole procedure is executed until the temperature is beneath a certain threshold, which is called the stopping criterion.

	Simulated annealing
1	Construct initial solution S_start
2	Initialize: S_best = S_start, temp = temp_start, temp_stop, markov_chain_length, and alpha
3	**While** temp > temp_stop **do**
4	**For** k = 1 **to** markov_chain_length **do**
5	S_new = generate_neighbour(S_start)
6	**If** F(S_new) < F(S_best) **then**
7	S_best = S_new
8	**If** F(s_new) <= F(S_start) **or** exp(- (F(S_start)- F(s_new)) / temp) < random(0,1) **then**
9	S_start = S_new
10	Temp = alpha * Temp
11	**Return** S_best

Figure 3.7: Pseudo code for the simulated annealing algorithm.

Tabu Search (TS) is very similar to SA in that it also goes from one solution to another with the next solution beginning possibly worse than the one before. Figure 3.8 shows the pseudo-code for TS. TS is different than SA in the way it accepts a neighbour solution. With TS a Tabu-list is made, which is the past X-amount of operations that are done. The Tabu-list shows the operations that are not allowed in the next iteration. One example of such an operation is the swapping of two particular jobs (see Section 3.4.5 for more examples). If an operation is done to create a neighbourhood solution then this operation is added on top of the Tabu-list, and the bottom one is deleted and can be used again during the next iteration. This procedure avoids returning to a local optimum and thus reaching a better neighbour solution space. The complete procedure is stopped when it has executed a certain amount of iterations.

Tabu search	
1	Construct initial solution S_start
2	Initialize: S_best = S_start, tabu_list = (), max_tabu_length
3	**While** stopping criteria **do**
4	S_new = generate_neighbour(S_start, tabu_list)
5	S_best = choose_best_neighbour(S_best, tabu_list)
6	S_start = S_best
7	**If** S_start < S_best **then**
8	S_best = S_start
9	**If** Length(tabu_list) >= max_tabu_length **then**
10	Remove_old(tabu_list)
11	Add_new(tabu_last)
12	**Return** S_best

Figure 3.8: Pseudo code for the tabu search algorithm.

3.4.5 Neighbourhood operators

The effectiveness of improvement heuristics is significantly influenced by the definition and utilization of neighbourhood operators, which dictate how a solution transitions to its neighbours. These operators try to move away from the existing solution space to get to a different one that might hold a better solution than the one already found.

The first successful neighbourhood structure is the **swapping** of any adjacent pair of critical operations on the same machine (J. Li et al., 2010). However, due to the large set of possibilities, other neighbourhood operators have been proposed. Moving an internal operation to the beginning or the end of the current block is one of them, and swapping the first two operations or the last two is another one (J. Li et al., 2010). However, one of the most effective neighbourhood operators is the **insertion**. This works by inserting an internal operation before the blockhead or after the block rear (J. Li et al., 2010). Thus moving an operation in between two other operations that have enough time in between them to place the selected operation. Mastrolilli and Gambardella (2000) use the **moving** operator, which moves an operation to any other position in the block. In addition, the insertion operator is used. Shahgholi, Katebi, and Daniavi (2018) also use the swap and insertion operators but also use the **reversion** operator, which takes a set of operations and reverses the sequence. Figure 3.4.5 shows how these three operators work graphically. The numbers indicate the jobs that are scheduled, and a reoccurring number indicates the operation sequence. For example the second 3 at "(a) Swap" means operation 2 of job 3 .

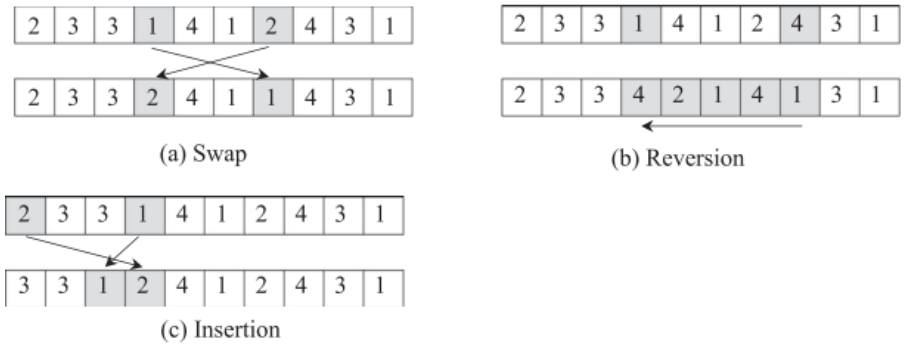


Figure 3.9: Neighbourhood operators (based on Shahgholi et al., 2018).

With the use of these neighbourhood operators, the improvement heuristics can search for new better solutions, and potentially get out of local optima.

3.5 Conclusion

This chapter provides an extensive literature review on production scheduling. Key elements of Johma's production system include the identification of its CODP being a mixed MTO and MTS system, and its internal processes are characterised by a Pull system. Furthermore, the review highlights the distinctions between planning and scheduling within such contexts, with a detailed explanation of the theoretical frameworks that guide these processes. This understanding informs the classification of Johma's scheduling problem within a taxonomy that addresses the complexities of scheduling in a manufacturing environment like Johma's.

Moreover, the scheduling problem faced by Johma is identified as a FJc$|prec,\ S_{ij}, M_{ij}, W_m, UWP_u^{m,w}|\sum w_i Tar_i, C_{max}$, revealing that Johma's problem aligns with a FJSP with characteristics like precedence, sequence-dependent setups, machine and worker eligibility, and available working time intervals. The goal hereby is to minimise the total weighted tardiness of jobs (recipes) and when this is realised try to find the lowest makespan.

Our exploration of solution approaches for the FJSP underscores the NP-hard nature of the problem, moving us away from exact methods towards heuristic solutions. In addition, we concluded that backward scheduling is the preferred method for scheduling jobs due to the nature of jobs having due dates, however, we will also examine the performance of the forward approach as our problem is unique. Based on this constructive heuristics are discussed, with dispatching rules being considered for their simplicity and effectiveness in generating initial feasible solutions. Specific attention is given to rules such as EDD, MS, and Random, which are identified as particularly relevant to Johma's operational characteristics.

Furthermore, the chapter compares various improvement heuristics and identifies Iterative Local Search (ILS), Simulated Annealing (SA), and Tabu Search (TS), as potential candidates for further exploration due to their ability to effectively explore the solution space and find near-optimal solutions within reasonable computational times. From these, SA and TS are highlighted as particularly promising for Johma's context due to their ability to escape local optima through the use of certain acceptance-rejection criteria. The potential of ILS is also examined to ensure a comprehensive approach to solving the scheduling issues. All these improvement heuristics use neighbourhood operators like move, swap, insertion or reversion to form different, and possibly better solutions.

As we progress into the model design in Chapter 4 in the next chapter, the insights gathered from this literature review guide the development of a scheduling model that is not only based on theory but also on the practical realities of Johma's production environment.

4 Model design

This chapter aims to determine an exact mathematical formulation for the problem at hand and to model several solution approaches with the use of constructive and improvement heuristics. The following research question is answered:

'How can the flexible job shop scheduling problem in Kitchen 1 be modelled?'

Section 4.1 provides the information on all the input data that is needed for our model. Section 4.2 describes the mathematical model together with the decisions, assumptions and simplifications. Moreover, this section also determines the exact mathematical formulation for our problem. Section 4.3 explains the developed constructive heuristic and Section 4.4 explains the neighbourhood strategy and the improvement heuristics in more detail. Section 4.5 provides all the possible solution options to be further experimented with in Chapter 5. Finally, Section 4.6 gives the conclusion of this chapter.

4.1 Input data

Figure 4.1 shows an example of all the input data of recipe R181. We can see per job and operation combination, the operator and machine needed together with the duration. Data on transportation, buffer, and cleaning times are also needed to make a viable schedule of the operations within Kitchen 1, and Figure 4.1 also shows this. In the lower-left corner, the dependence is shown in the figure. During observations and visits on the shop floor, this data is gathered using a timestamp app, looking into company data, and by talking to employees. This is needed as there was almost no data available. The data of operations per machine and recipe are converted into general times because not every unique combination of an operation and an ingredient can be measured. Together with the operators on the shop floor, a standardised framework is made to see which operations are very comparable. It thus can be regarded that similar operations on the same type of machine have the same processing, cleaning, and transport time. The amounts of kg per recipe are also in many situations different, therefore we convert the data to correspond to the amount in the recipe. Available work times are gathered from operators and production managers, and due dates are calculated by a tool that is specifically made for this research.

Job number	Operation number	Machine	Operator	Duration machine (min)	Duration operator (min)
14	1	n.a.	1	n.a.	1
14	1	n.a.	2	n.a.	1
14	1	n.a.	3	n.a.	1
14	2	1	1	6,083333333	6,083333333
14	2	1	2	6,083333333	6,083333333
14	2	1	3	6,083333333	6,083333333
14	3	1	1	0,933333333	0,933333333
14	3	1	2	0,933333333	0,933333333
14	3	1	3	0,933333333	0,933333333
14	4	3	1	4,95	4,95
14	4	3	2	4,95	4,95
14	4	3	3	4,95	4,95
14	5	n.a.	1	n.a.	1
14	5	n.a.	2	n.a.	1
14	5	n.a.	3	n.a.	1
14	6	n.a.	5	n.a.	2,15
14	7	n.a.	5	n.a.	1,433333333
14	8	9	6	5,133333333	2
14	8	10	6	5,133333333	2

Job, Operation	Transport times (min)	Buffer times (min)	Cleaning times (min)
14, 1	1,2	1	0
14, 2	2	1	1,05
14, 3	1,2	1	1,05
14, 4	1,4	1	12,75
14, 5	1,6	1	0
14, 6	1,6	1	0
14, 7	2	1	0
14, 8	0,8	1	4,216666667

Job number	Operation	Job, Operation	Depends on
14	1	14,1	n.a.
14	2	14,2	n.a.
14	3	14,3	n.a.
14	4	14,4	14,3
14	5	14,5	14,4
14	6	14,6	14,1;14,2;14,5
14	7	14,7	14,6
14	8	14,8	14,7

Figure 4.1: Input data of recipe R181.

Table 4.1 shows the available and unavailable working time during the days of the week. Unavailable work times are between the following intervals (time units in minutes): [0, 15), [180, 200), [360, 390), [465, 495), [620, 640), [810, 840), [945, 975), [1110, 1130), [1290, 1320), [1425, 1440).

	Night shift	Morning shift	Afternoon shift
Sunday/Monday	22:00–06:00	06:00-14:00	14:00-22:00
Breaks	01:00-01:20, 04:00-04:30	8:30-8:50, 11:30-12:00	16:30-16:50, 19:30-20:00
Shift change	22:00-22:15	06:00-06:15	14:00-14:15
Cleaning	05:45-06:00	13:45-14:00	21:45-22:00
Monday/Tuesday	22:00–06:00	06:00-14:00	14:00-22:00
Breaks	01:00-01:20, 04:00-04:30	8:30-8:50, 11:30-12:00	16:30-16:50, 19:30-20:00
Shift change	22:00-22:15	06:00-06:15	14:00-14:15
Cleaning	05:45-06:00	13:45-14:00	21:45-22:00
Tuesday/Wednesday	22:00–06:00	06:00-14:00	14:00-22:00
Breaks	01:00-01:20, 04:00-04:30	8:30-8:50, 11:30-12:00	16:30-16:50, 19:30-20:00
Shift change	22:00-22:15	06:00-06:15	14:00-14:15
Cleaning	05:45-06:00	13:45-14:00	21:45-22:00
Wednesday/Thursday	22:00–06:00	06:00-14:00	14:00-22:00
Breaks	01:00-01:20, 04:00-04:30	8:30-8:50, 11:30-12:00	16:30-16:50, 19:30-20:00
Shift change	22:00-22:15	06:00-06:15	14:00-14:15
Cleaning	05:45-06:00	13:45-14:00	21:45-22:00
Thursday/Friday	22:00–06:00	06:00-14:00	14:00-22:00
Breaks	01:00-01:20, 04:00-04:30	8:30-8:50, 11:30-12:00	16:30-16:50, 19:30-20:00
Shift change	22:00-22:15	06:00-06:15	14:00-14:15
Cleaning	05:45-06:00	13:45-14:00	21:45-22:00

Table 4.1: Shift schedule and corresponding unavailable working time intervals.

The inputs for the model are therefore, the processing times, the due dates of the recipes for the filling lines, transportation times, setup times (cleaning), buffer times in between operations, and the intervals where no work is allowed (breaks and shift changeovers). Besides this, other important inputs are the sequence of operations within a job and the due dates. The due dates are calculated with an Excel tool that is specifically made for this research. This tool can calculate different due dates based on the schedule from Figure 2.14 and on the dispatching rules used in the constructive heuristic from Section 3.4.3, to create a job list as input for the model. Section 4.3.2 explains this with an example.

4.2 Problem statement and formulation

This section determines a problem description and an exact mathematical model formulation using the taxonomy from Chapter 3. Section 4.2.1 describes the problem. Section 4.2.2 provides the decisions that must be incorporated into the model, and Section 4.2.3 the model assumptions and simplifications. Moreover, Section 4.2.4 provides the sets, parameters and variables used in the exact model formulation and Section 4.2.5 explains the mathematical formulation of the scheduling model.

4.2.1 Description

The scheduling problem at hand revolves around scheduling the operations within Kitchen 1, reflecting a FJS scenario, where distinct recipes (jobs) have a specific sequence of operations, each tailored to their unique characteristics and constraints. The goal is to optimise the scheduling process by minimising the average tardiness of recipes (jobs). Tardiness occurs when a job is completed after its due date d_i. Kitchen 1, has a specific set of recipes (job orders) to be scheduled per shift and consequently per day from the set of all available jobs I. All the machines are from the set M and all the operators are from the set W. These resources are used to process the operations O_i, where each job consists of q operations that have to be processed in a certain sequence by one of the eligible machines from $M_{i,j}$ and eligible workers from W_m. The operations have specific processing times $p_{i,j}$, and setup times $set^m_{i,j,g,p}$, which are predetermined and unique to a certain operation on a certain machine in case of the setup time. The number of operations varies depending on the specific ingredients of the corresponding job. Furthermore, the scheduling consists of transportation logistics, as ingredients must be timely transported to the department for processing. This transport time $t_{i,j}$ is included per ingredient that needs to be transported from another department to Kitchen 1. Internal transport times are taken into account in the processing time. The decision variables come into play in determining the optimal starting times for each operation denoted as $ST_{i,j}$. Additionally, binary variables such as $Y^m_{i,j;k,l}$, $X^{m,w}_{i,j}$, and $X^w_{i,j;k,t}$ dictate the assignment of operations to machines and operators, operators to machines, ensuring that these operations can be executed concurrently while also maintaining the prescribed sequence constraints. $Y^m_{i,j;k,l}$ is used if an operation k_t is processed on machine m directly after i, j, $X^{m,w}_{i,j}$ if operation i, j is processed by worker w on

machine m, and $X^{w}_{i,j;k,t}$ if operation k,t is processed by worker w directly after i,j.

This scheduling puzzle shows the complication of flexible job shop scheduling, wherein the complexity is highlighted by the relationships between recipe characteristics (β) and optimisation objectives (γ). Notably, the objective function aims to minimize idle time at the filling lines, which leads to minimising the tardiness of recipes (jobs) at the filling lines.

4.2.2 Modelling decisions

The model should decide the following:

- The assignment of operations to the available eligible machines and operators (Constraints (5), (6), (7), (11), and (12) in Section 4.2.5).
- The sequence and routing of operations of the recipes to be scheduled (Constraints (7), (8), and (9) in Section 4.2.5).
- Starting and ending times of the operations of the recipes on machines (Constraints (8), (9), (13), and (14) in Section 4.2.5).
- If setups need to happen in between operations of different recipes (Constraints (8), (9), (13), and (14) in Section 4.2.5).

4.2.3 Model assumptions and simplifications

The following assumptions are made to simplify the modelling:

- All data for input is deterministic and thus known in advance. This is done because we have time limitations, it uses less computational time which is required at Johma, and it is more accurate with the measured input data as we do not introduce any bias or errors from data Johma currently has. However, in the real world, some data is stochastic as we are dealing with many variables that can change in real-time.
- Internal transportation time is negligible or regarded within processing time ($p^{m,w}_{i,j}$). Only transportation time ($t_{i,j}$) from ingredients towards the department is taken into account as this is significant.
- No machine breakdowns or maintenance. In the real world, machines do break down, and maintenance is needed. However, no data on this is available so we do not incorporate this in the model.
- We only use the two large mixers (machine $m = 9, 10$) in our model. In the real world, a small mixer is also available, however, this mixer is only used for testing and tasting.
- All operators (W) are always available during working hours (forbidden intervals between *uwtstart* and *uwtend*). In the real world, operators sometimes have to go to the toilet or have meetings for example.
- The number of carts and funnels is regarded as infinite. In the real world, there is a fixed amount of these present in the production facility for all departments. Johma states that the amount of carts and funnels is enough.
- All ingredients are available to be used from time zero. In the real world, ingredients might be not available at all or available later during the day due to the dependency on other departments.
- Miscellaneous activities are added to the operations as buffer time ($b_{i,j}$):
  - Disposal of waste (plastic packaging and buckets) is included in buffer time.
  - Communication between all the operators within the department is included as buffer time.
  - Label writing and other tasks, like changing gloves and apron, are either neglected or allocated to buffer time.
- Operations cannot be interrupted once started. In the real world this is possible, but not recommended due to the need for remembering ingredient amounts. Quality issues occur when operations are allowed to be interrupted.
- We assume infinite buffer space in between and after operations. In the real world, the buffer space is finite. However, this space is rarely full.

- An operator can work immediately on the next operation when he or she is finished with the previous one. In the real world, some time is needed in between to get from one operation to another. This is included in the buffer time.
- All operators are regarded to work at the same pace. In the real world, some full-time operators have a lot of knowledge and experience and can operate without the need for guidance. However, flex operators have less knowledge or sometimes even none. They need more guidance.

4.2.4 Sets, parameters, and variables

Table 4.2 shows the defined sets, parameters, and variables.

Table 4.2: Description of Notation, Sets, Parameters, and variables.

Notation	Description
i	Indice of jobs (recipe)
q	Last operation of job
j	Indice of operations within job
m	Index of machines
w	Index of workers
u	Index of unavailable working periods
Sets	
$I = \{1, \ldots, n\}$	Set of all jobs
M	Set of all machines
W	Set of all workers
$O_i = \{i, 1; \ldots; i, q\}$	Set of j operations for job i where q is the last operation
$M_{i,j} \subseteq M$	Set of eligible machines for operation j of job i
$W_m \subseteq M$	Set of eligible workers for machine m
$V := \bigcup_{i \in I} O_i \cup \{0_1\}$	Set of all operations including dummy operation 0_1
$\bar{V} := V \setminus \{0_1\}$	Set of all operations excluding dummy operation
$V_{i,j} := V \setminus \{i, k \mid k \leq j\} \quad \forall i, j \in V$	Set including all operations from V except for the operations i, k where k (the index of the operation within job i) is less than or equal to j. It excludes all operations of job i up to and including the j^{th} operation
$\tilde{V}_{i,j} := V \setminus \{i, k \mid k \geq j\} \quad \forall i, j \in V$	Set including all operations from V except for the operations i, k where k (the index of the operation within job i) is greater than or equal to j. It excludes all operations of job i starting from the j^{th} operation onwards
$\bar{V}_{i,j} := V_{i,j} \setminus \{0_1\} \quad \forall i, j \in V$	Set including all operations from $V_{i,j}$ except for the dummy operation. Thus it excludes the operations of job i up to and including the j^{th} operation and also excludes the dummy operation.
Parameters	
$p_{i,j}^{m,w}$	Processing time of i, j with m and w
$b_{i,j}$	Buffer time of i, j
$t_{i,j}$	Transport time of i, j
$set_{i,j;g,p}^{m}$	Setup time when processing operation i, j after operation g, p on machine m
d_i	Due date of recipe i that the recipe should be finished
w_i	Importance weight based on due date d_i of job i
$uwtstart_u^{m,w}$	Starting times of unavailable work period u for m and w
$uwtend_u^{m,w}$	Ending times of unavailable work period u for m and w
$UWP_u^{m,w}$	The u^{th} no available working time period of m and w
Variables	
Tar_i	Tardiness of recipe i
C_i	Completion time of recipe i
$ST_{i,j}$	Starting time of i, j
$S_{i,j;g,p}^{m,w}$	Binary decision variable if setup time is between i, j and g, p by worker w on machine m
$Y_{i,j;k,l}^{m}$	Binary decision variable if operation k, l is processed on machine m directly after i, j
$X_{i,j}^{m,w}$	Binary decision variable if operation i, j is processed by worker w on machine m
$X_{i,j;k,l}^{w}$	Binary decision variable if operation k, l is processed by worker w directly after i, j

4.2.5 Mathematical formulation

This section constructs and explains the mathematical formulation of the model. The different elements for this so-called Mixed Integer Linear Program (MILP) are based on the literature from Bissoli et al. (2019), Kress and Müller (2019), Sobeyko and Mönch (2016), Pinedo (2016), and J. Gao, Gen, and Sun (2006) extended with elements specific to our problem.

The main objective is to minimise the total weighted tardiness of recipes, and when this is minimised we also want to minimise the makespan. The objective functions below show this. This objective is derived from the goal of minimising the idle times at the filling lines. When minimising the total weighted tardiness the idle times at the filling lines are also minimised. As stated in Section 1.2.1 idle time in our case means the starvation of the filling lines due to not having salad mixture from Kitchen 1.

$$minimise \quad f_1 = \sum_{i \in I} w_i * Tar_i, \quad \forall i \in I \tag{1}$$

$$minimise \quad f_2 = C_{max} \tag{2}$$

Constraint 1 shows the first objective. We sum over the tardiness of all jobs and make sure they are corrected with the corresponding weight of that job. This sum is minimised to make sure we reach our objective. Constraint 2 shows the second objective. Here we minimise the makespan. The idea is to first minimise the weighted tardiness and when we have the same tardiness, but a lower makespan, we will pick that solution. However, when we have a higher tardiness and a lower makespan this solution is not picked.

$$Tar_i \geq C_i - d_i, \quad \forall i \in I \tag{3}$$

Constraint 3 shows that tardiness is calculated by getting the difference between the completion time of a job and the due date of a job. To make sure it does not become negative a greater than or equal to sign is added as tardiness can only be zero or positive. A job can only be too late, but not early as a filling line only needs a recipe to just be there when needed.

$$\begin{aligned} C_i \geq ST_{i,q} + \sum_{m \in M_{i,q}} \sum_{w \in W_m} X^{m,w}_{i,q} * ((p^{m,w}_{i,q} + b_{i,q} + t_{i,q}) + (S^{m,w}_{i,q;g,p} * set^m_{i,q;g,p})), \\ \forall i \in I, \forall i,q \in \bar{V}, \forall g,p \in V_{i,j}, i,q \neq g,p \end{aligned} \tag{4}$$

Constraint 4 shows the calculation of the completion time of a job. A job has several operations and the completion time of a job corresponds to the finishing time of the last operation (q) of that job. The finishing time of an operation is the start time plus the processing, buffer, transport, and setup time (when needed).

$$\sum_{i,j \in \tilde{V}_{k,l}} \sum_{m \in M_{i,j;k,l}} Y^m_{i,j;k,l} = 1, \quad \forall k,l \in \bar{V} \tag{5}$$

Constraint 5 shows that each operation is assigned to exactly one of its eligible machines. This is done by summing over all possible assignments of that operation and making this equal to one.

$$\sum_{i,j \in \bar{V}} Y^m_{0_1;i,j} \leq 1, \quad \forall m \in M \tag{6}$$

Constraint 6 shows that a dummy operation is processed first on each machine. This is done to initiate the model as the first job on a machine also needs a preceding operation to be able to be scheduled on a machine.

$$\sum_{k,l \in \tilde{V}_{i,j} \text{ with } m \in M_{k,l}} Y^m_{k,l;i,j} - \sum_{k,l \in V_{i,j} \text{ with } m \in M_{k,l}} Y^m_{i,j;k,l} = 0, \quad \forall i,j \in V, m \in M_{i,j} \tag{7}$$

Constraint 7 ensures that binary variables are set to their correct values for the specific machine that processes an operation, as well as its preceding and succeeding operations (including the operation of the dummy job) on this very machine. $Y^m_{k,l;i,j}$ is equal to 1 when operation k,l is immediately followed by operation i,j on machine m, and $Y^m_{i,j;k,l}$ is equal to 1 when operation i,j is immediately followed by operation k,l on machine m. Thus

for every operation i, j on machine m, the number of times i, j is preceded by another operation is equal to the number of times operation i, j is followed by another operation on machine m. This leads to ensuring that each operation has exactly one predecessor and one successor thus ensuring a logical sequence of operations on machines.

The following example explains this in more detail. Machine m can process all operations i_1, f_1, k_1, $V = i_1; f_1; k_1; 0_1$, and $\bar{V}_{i_1} = f_1; k_1; 0_1$. The sequence on this machine is 0_1, i_1, f_1, k_1. The binary decision variables become; $Y^m_{0_1;i_1} = 1$, $Y^m_{i,1;f,1} = 1$, and $Y^m_{f,1;k,1} = 1$. Thus the sum of predecessors of $i, 1$ is $Y^m_{0_1;i,1}$ and the sum of successors of $i, 1$ is $Y^m_{f,1;k,1}$. Filling in the constraints results in 1 - 1 = 0, thus ensuring that there is a logical sequence of operations.

$$\begin{aligned} ST_{i,j} + \sum_{w \in W_m} X^{m,w}_{i,j} * ((p^{m,w}_{i,j} + b_{i,j} + t_{i,j}) + (S^{m,w}_{i,j;g,p} * set^m_{i,j;g,p})) - ST_{k,l} \\ \leq (1 - Y^m_{i,j;k,l})M, \quad \forall i, j \in \bar{V}, \forall k, l \in \bar{V}_{i,j} \forall g, p \in V_{i,j}, m \in M_{i,j;k,l}, i, q \neq k, l, i, q \neq g, p \end{aligned} \tag{8}$$

Constraint 8 prevents overlapping of two consecutive operations (g, p and i, j) on the same machine. Also, other precedence relations between consecutive operations are taken into account. This constraint makes sure that the finish time of the current operation is before the starting time of the next operation. The big M part makes sure that the equation is not violated. It must be large enough to ensure the model behaves correctly, but not excessively large to avoid numerical issues or inefficiencies. The precise determination of this value is not in the scope of this research. Furthermore, this constraint restricts that operation g, p precedes operation i, j, and not the other way around.

$$\begin{aligned} ST_{i,j} + \sum_{m \in M_{i,j}} \sum_{w \in W_m} X^{m,w}_{i,j} * ((p^{m,w}_{i,j} + b_{i,j} + t_{i,j}) + (S^{m,w}_{i,j;g,p} * set^m_{i,j;g,p})) \leq ST_{i,j+1}, \\ \forall i, j \in \bar{V} \text{ with } j \leq i, q-1, \forall g, p \in V_{i,j}, i, q \neq g, p \end{aligned} \tag{9}$$

Constraint 9 is needed for the previous constraint to improve the computational performance, and hierarchical scheduling for the sequence of operations in our tests. It specifically focuses on the finish time of the current operation before the starting time of the successor operation.

An example can be given for operation i, j. The starting time of this operation is $ST_{i,j}$. Furthermore, the processing time is $p^{m,w}_{i,j}$, the buffer time is $b_{i,j}$, the transport time is $t_{i,j}$, and the setup time is $set^m_{i,j;g,p}$ when a setup is needed. These parameters are taken into account when an operator and or machine is selected for operation i, j. So in short, the constraint ensures that the next operation $i, j+1$ can only start when operation i, j is started at time $ST_{i,j}$ and all needed production steps are over.

$$ST_{i,j} + p^{\min}_{i,j} + b_{i,j} + t_{i,j} + (S^{\min}_{i,j;g,p} * set^{\min}_{i,j;g,p}) \leq ST_{i,j+1}, \quad \forall i, j \in \bar{V} \text{ with } j \leq i, q-1, \forall g, p \in V_{i,j}, i, q \neq g, p \tag{10}$$

Constraint 10 is needed for the previous constraint to improve the computational performance in our tests. $p^{\min}_{i,j}$ reflects the minimum processing time of operation j of job i, considering all eligible machines and the fastest worker for each operation. Furthermore, the other times also show the quickest possibilities. These values reflect the shortest time in which an operation can be completed, irrespective of the specific machine or worker assignments. Therefore, this constraint decreases the number of possible options, which leads to a smaller solution space, which in terms decreases computation time.

$$\sum_{m \in M_{i,j}} X^{m,w}_{i,j} + \sum_{m \in M_{k,l}} X^{m,w}_{k,l} - 1 \leq X^w_{i,j;k,l}, \quad \forall i, j; k, l \in \bar{V}, i, j \neq k, l; w \in W_m \tag{11}$$

Constraint 11 is needed to ensure consistency between the assignment of workers and the binary variables that indicate such assignments. It effectively determines the situation where a worker is marked as assigned to two preceding operations. It enforces that a worker is then also assigned the other binary variable that shows this precedence (represented by $X^w_{i,j;k,l}$).

An example can be given for operation i, j. $X^{m,w}_{i,j}$ is equal to 1 when operation i, j is done by worker w on machine m, $X^{m,w}_{k,l}$ is equal to 1 when operation k, l is done by worker w on machine m, $X^w_{i,j,kl}$ is equal to 1 when worker w processes operation i, j and k, l consecutively. We have machine m_1 for operation i, j and machine

m_2 for k,l. Operation k,l is done after i,j by worker w. Thus part one (sum over $X_{i,j}^{m,w}$) becomes 1, part two (sum over $X_{i,j;k,l}^{w}$) becomes 1, and part 3 ($X_{i,j;k,l}^{w}$) becomes 1. Putting this into the constraint results in 1 + 1 - 1 $\leq$ 1 which is correct.

$$\sum_{k,l \in \bar{V}_{i,j} \text{ with } m \in M_{i,j}} Y_{i,j;k,l}^{m} = \sum_{w \in W_m} X_{k,l}^{m,w}, \quad \forall k,l \in \bar{V}, m \in M_{k,l} \tag{12}$$

Constraint 12 connects the sequencing variables to the worker assignment variables. By setting these two sums equal, the constraint ensures that the precedence relationships are properly reflected in the assignment of workers to operations. It ties the order in which operations are scheduled to the actual assignment of workers to those operations. This ensures that workers are assigned to operations in the correct sequence and that the sequencing of operations is consistent with the worker's assignments.

An example can be given for operation i,j. $Y_{i,j,k,l}^{m}$ is equal to 1 when operation i,j is immediately followed by operation k,l on machine m and $X_{k,l}^{m,w}$ is equal to 1 when operation k,l is done by worker w on machine m. We have machine m_1 for operation i,j and k,l, and machine m_1 can be operated by worker w_1 or w_2. We assume that these operations are done with worker w_1. Furthermore, $Y_{i,1;k,1}^{m_1} = 1$ means that operation $i,1$ is followed by $k,1$ on machine m_1, and that $X_{k,1}^{m_1,w_1} = 1$ means that worker w_1 is assigned to operation $k,1$ on machine m_1. Following this we can rewrite the constraint as $Y_{i,1;k,1}^{m_1} = X_{k,1}^{m_1,w_1} + X_{k,1}^{m_1,w_2}$ results in $1 = 1 + 0$, which satisfies.

$$uwtend_{u}^{m,w} \leq ST_{i,j} \leq uwtstart_{u+1}^{m,w}, \quad \forall i,j \in \bar{V}, \forall u \in UWP, \forall m \in M_{i,j}, \forall w \in W_m \tag{13}$$

$$\begin{aligned} uwtend_{u}^{m,w} \leq ST_{i,j} + \sum_{m \in M_{i,j}} \sum_{w \in W_m} X_{i,j}^{m,w} * ((p_{i,j}^{m,w} + b_{i,j} + t_{i,j}) + (S_{i,j;g,p}^{m,w} * set_{i,j;g,p}^{m})) \\ \leq uwtstart_{u+1}^{m,w}, \quad \forall i,j \in \bar{V}, \forall g,p \in V_{i,j}, \forall u \in UWP, \forall m \in M_{i,j}, \forall w \in W_m, i,q \neq g,p \end{aligned} \tag{14}$$

Constraints (13) and (14) show that operations can only be scheduled in between certain time intervals where working time is available. Constraint (13) shows that the starting time of the operation can only be between the end time of the previous forbidden interval and before the start time of the next forbidden interval. However, to make sure that the start and finish time of an operation do not span over a full forbidden interval Constraint (14) must be added. This constraint makes sure that the finish time must also be in the same available working time interval as the start time of that operation.

$$Y_{i,j;k,l}^{m} \in \{0,1\} \quad \forall i,j \in V, k,l \in V_{i,j}, m \in M_{i,j;k,l} \tag{15}$$

$$X_{i,j}^{m,w} \in \{0,1\} \quad \forall i,j \in \bar{V}, m \in M_{i,j}, w \in W \tag{16}$$

$$X_{i,j;k,l}^{w} \in \{0,1\} \quad \forall i,j,k,l \in \bar{V}, i,j \neq k,l, w \in W \tag{17}$$

$$S_{i,j;g,p}^{m,w} \in \{0,1\} \quad \forall i,j \in V, g,p \in V_{i,j}, i,j \neq g,p; m \in M_{i,j}, w \in W \tag{18}$$

$$V_{i,j;k,l} \in \{0,1\} \quad \forall i,j;k,l \in \bar{V}, i,j \neq k,l \tag{19}$$

$$ST_{i,j} \in N_0^+ \quad \forall j \in \bar{V} \tag{20}$$

$$C_i \in N_0^+ \quad \forall i \in I \tag{21}$$

$$Tar_i \geq 0, \quad \forall i \in I \tag{22}$$

Constraints 15-22 define the variables.

4.3 Constructive heuristic

Heuristics can be used to find a near-optimal solution for our problem as the exact approach is not suitable for larger problem instances. This section explains how the constructive heuristic is made to form an initial feasible solution. Section 4.4 elaborates on the improvement heuristics to improve the initial solution made with the constructive heuristic.

The exact model (MILP) from Section 4.2.5 helps to understand the problem in more detail. With this knowledge, a constructive heuristic is made with elements from the exact model. The constructive heuristic for our problem consists of the following parts that logically make a feasible and realistic schedule:

- *Dispatching rules*
- *Initial job list tool*
- *Sequencing of operations*
- *Operator and machine assignment*
- *Setup assignment*
- *Other logic*

Sections 4.3.1 explains the use of the dispatching rules to create the order of jobs as input for the model. Section 4.3.2 explains the initial job list tool that is created to calculate the individual due dates of funnels as currently only the due date of the first funnel of a whole batch of the same salad is present in the schedule at Johma. Section 4.3.3 explains how the operations are sequenced and thus what relationships between the jobs and operations are taken into account in the model. Section 4.3.4 explains how operators and machines are assigned to the operations. Section 4.3.5 explains how the setups are scheduled when this is needed. Lastly, Section 4.3.6 explains about the forbidden intervals that have to be taken into account.

4.3.1 Dispatching rules

In Section 3.4.3 we introduced some dispatching rules that can be used to form an initial job list. This job list is the main input for our scheduling model. For our objective, minimising the weighted tardiness of jobs, the EDD and MS dispatching rules from Section 3.4.3 are used. These dispatching rules create two job lists; in the EDD job list the jobs are ordered based on the earliest due date, and in the MS job list the jobs are ordered based on the earliest due date and taking into account the processing time of that job. This means that jobs that have a later due date but have a high processing time are put earlier in the job list. The job list therefore determines the order in which the jobs need to be scheduled. It is the most important input for the scheduling model as it greatly determines how good the solution becomes. In addition, to potentially create very different initial solutions, the random rule is used. Together with the improvement heuristics different solution spaces can be explored, potentially escaping local optima as explained in Section 3.4.4.

4.3.2 Initial job list tool

With the dispatching rules from the previous section three types of job lists can be created with the initial job list tool. Either the EDD, MS or random job list. These are created with the use of an Excel tool that is specifically made for this research. The tool splits the initial recipe planning into jobs for Kitchen 1. Table 4.3 shows a part of an initial job list extracted from company data. We can see the jobs (R-numbers) with the number of runs, the corresponding filling line, the due date and other information. For example, for recipe R115 8 runs are scheduled for time 00:40. This means that the first funnel (2 runs are 1 funnel of salad mixture) should be available at filling line 6 at 00:40. However, this planning does not indicate the time the second funnel should be available, therefore we need to calculate this to know when the work should be finished in Kitchen 1.

Table 4.3: Part of job list from company data.

R-number	Runs	Filling line	Due Date	Day	Name	Kilos
R115	8	6	00:40	1	GEVULD EITJE	2304.00
R993	6	13	00:40	1	EI	1680.00
R474	12	14	01:45	1	VRIJE UITLOOP EI	3443.76

The job list from Table 4.3 is converted to sub-jobs to create a full job list of all funnels for Kitchen 1. This is done by converting the runs to funnels (two runs add up to one funnel) and thus creating sub-jobs for Kitchen

1 with due dates per funnel. We need this because, without the due dates for individual funnels, we cannot schedule the jobs and operations in Kitchen 1 and calculate if the work is on time at the filling lines or if there is tardiness. The due dates are calculated by converting the kilos per funnel to minutes per funnel with the speed of the filling lines. Table 4.4 shows the converted job list, which takes into account the speed of the filling lines and calculates backwards what the finish time should be for each job (funnel of salad mixture ready for the filling line). The table shows that eight runs for R115 are converted to four different sub-jobs (job 9) for Kitchen 1, all with their own finish time (due date) to be available for the filling line. With this, a schedule can be made in Kitchen 1 for the individual jobs and operations.

Table 4.4: Converted job list information.

R-number	Job	Filling line	Due Date in minutes	Name
R115	9	6	40	GEVULD EITJE
R993	89	13	40	EI
R115	9	6	56.94	GEVULD EITJE
R993	89	13	62.4	EI
R115	9	6	73.88	GEVULD EITJE
R993	89	13	84.8	EI
R115	9	6	90.82	GEVULD EITJE
R474	60	14	105	VRIJE UITLOOP EI
R474	60	14	129.95	VRIJE UITLOOP EI
R474	60	14	154.91	VRIJE UITLOOP EI
R474	60	14	179.86	VRIJE UITLOOP EI
R474	60	14	204.82	VRIJE UITLOOP EI
R474	60	14	229.77	VRIJE UITLOOP EI

With EDD the job list is created in chronological order from the earliest due date to the latest due date. With MS the jobs are also ordered from the earliest to the latest due date, but the due dates are calculated by also including the processing time. Thus jobs with more processing time could potentially be scheduled earlier than jobs with an earlier initial due date. At last, with the random rule jobs are ordered randomly to form a job list. This tool therefore already creates the job order to be scheduled in Kitchen 1, and the model in Python uses these job lists as input.

4.3.3 Sequencing of operations

Constraints (6), (7), (8) and (9) from the exact model show that preceding and succeeding operations cannot overlap and that processing, transport, buffer, and setup times (see Section 4.3.5) need to be taken into account. This knowledge forms the basis for the sequencing logic. The job lists from the previous section are used as input for the scheduling model. All jobs consist of several operations that have a pre-defined sequence, which corresponds to certain precedence relationships between operations. Figure 4.1 showed the jobs that are preceding in the "Depends on" column. With this information, the scheduling model can only schedule an operation if all preceding operations of an operation within the job are scheduled. When this is the case the model calculates the earliest possible starting time and finish time of the operation by looking at all eligible possibilities (see Section 4.3.4). The difference between backward and forward scheduling is that with forward the operations are scheduled from the first to the last, and with backward the operations are scheduled from the last to the first. Moreover, backward scheduling looks at the due date as a release date for the operations of a job, and for forward scheduling, this is time zero. Furthermore, for both backward and forward scheduling, an operation is selected from the input job list and scheduled at the earliest time possible. The sequencing logic looks at the eligible combinations and schedules the operation on the first combination possible. The earliest start time depends on when the operator and machine combination is available (see Section 4.3.4), if a setup is needed (see Section 4.3.5), and if the first available start time found is within a forbidden interval (see Section 4.3.6). All jobs with corresponding operations are scheduled with this logic until there is no one left in the job list.

4.3.4 Operator and machine assignment

All operations have a set of eligible operators and, if needed, a set of eligible machines. Figure 4.1 shows this in the "Machine" and "Operator" columns. Constraints (5), (11), and (12) from the exact model show that one operator and if needed one machine from the eligible set of operators and machines should be chosen for every operation. This is done by looking at the next available time of the eligible operators and machines and selecting the one(s) that is/are available the earliest. One special rule is that for an operation that requires machine 4, two eligible operators must be chosen, and for an operation that requires operator 6, machine 9 or 10 must be chosen. Machine 4 corresponds to the pilling machine, which must be operated by two employees because of the heavy labour, and machines 9 and 10 correspond to the mixers, which are operated by operator 6. Both require some unique scheduling logic to see which worker and when what machine is available for processing.

4.3.5 Setups

Setups, in our case cleaning a machine, happen when the previous operation on the same machine is from a different job (recipe). Our scheduling model looks at the previous job and if it is different, a setup is first scheduled and after this the operation itself. Constraints (4), (8), (9), (10), and (14) shows the setup integrated into the exact formulation.

4.3.6 Other logic

Besides all the parts mentioned above, one other thing has to be taken into account in the scheduling logic. Operations cannot be scheduled in the forbidden working time intervals from Table 4.1. Constraints (13), and (14) from the exact model show this. The start and finish time of an operation must be after the end time of an interval and before the start time of the next interval. When this is not the case the start time of the operation should be moved after the next forbidden interval.

The job list, which is used as input made with the dispatching rules, the sequencing logic, the operator and machine assignment, the setups, and other logic from the sections mentioned above are combined into a constructive heuristic. Figure 4.2 shows the pseudo code for this constructive heuristic. The difference between backward and forward scheduling is in the release date (due date versus time zero) and how start and finish times are calculated. With backward we switch the whole schedule around so we start with the last job to be scheduled and start with the last operation in that job to be scheduled and in the end turn the whole schedule around by stating that the start time is the finish time and the finish time is the start time. All logic and the pseudo-code are the same.

	Constructive heuristic
1	**Initialise**: Job list with corresponding information per job and operation with dispatching rule Predecessor list per job Other tables and information needed
2	**While** jobs to be scheduled is not empty **do**
3	**For** each job in jobs to be scheduled **do**
4	Select first job and first operation with corresponding information
5	Get predecessor information + eligible operator and/or machine
6	**If** predecessors are completed **then**
7	Select machine + calculate time information
8	**If** machine = 4 **then**
9	**For** each eligible combination of operator and/or machine **do**
10	Select machine, operator, and processing time
11	Calculate start time and next available time for operator and/or machine
12	Select best two eligible operators
13	Calculate best two starting times and set overall start and cleaning start time
14	**If** last job on machine is different **then**
15	Schedule cleaning operation and update start time
16	**For** each interval in forbidden intervals **do**
17	Check if start and end times are not in forbidden intervals otherwise adjust
18	**Update** starting times, next available time for operators and/or machines
19	**Append** operations (normal and cleaning) to schedule
20	**If** machine != 4 **then**
21	**For** each eligible combination of operator and/or machine **do**
22	Select machine, operator, and processing time
23	Calculate start time and finish time
24	**If** start time is better than best found start time **then**
25	Update start time + eligible combination
26	**Select** start time from best found start time
27	**If** last job on machine is different **then**
28	Schedule cleaning operation and update start time
29	**For** each interval in forbidden intervals **do**
30	Check if start and end times are not in forbidden intervals otherwise adjust
31	**Update** starting times, next available time for operators and/or machines
32	**Append** operations (normal and cleaning) to schedule
33	**Remove** operation from jobs operations to be scheduled
34	Create table of operations and cleaning operations
35	**Result** feasible schedule of all jobs and operations

Figure 4.2: Pseudo code for the constructive heuristic algorithm.

In Appendix A a flow chart of the constructive heuristic is located that combines the elements described in this section with the pseudo code from Figure 4.2. With the constructive heuristic, an initial solution is made to be further improved upon by the improvement heuristics in the next section.

4.4 Improvement heuristics

The constructive heuristic from Figure 4.2 can be used to form an initial solution. With the help of an improvement heuristic, this initial solution can be made better by iteratively refining it. As already discussed in Section 3.4.4, ILS, SA, and TS are suitable methods to solve our FJSP. Section 4.4.1 explains what operators can be used within the improvement heuristics to find better neighbour solutions. Furthermore, Sections 4.4.2, 4.4.3, and 4.4.4 elaborate on the different improvement heuristics respectively.

4.4.1 Neighbourhood solution generation

Neighbourhood solutions of the initial solution made with the constructive heuristic from Section 4.3 can be found using the improvement heuristics. This is done in the first place by using neighbourhood operators. The operators refine the order of jobs to find neighbouring solutions. These so-called neighbours can potentially be better than the initial solution found with the constructive heuristic. By iteratively using the right neighbouring operators together with the right improvement heuristic a near-optimal solution can be found. Based on Section 3.4.5 the following neighbourhood operators are used:

- **NO1:** *Swap* two adjacent jobs in the sequence
- **NO2:** *Swap* two jobs in the sequence
- **NO3:** *Insert* a single job between two other jobs in the sequence
- **NO4:** *Insert* a single job between two other jobs in the sequence multiple times

Many papers discuss the concept of swapping or moving operations to any other position in the sequence ((Mastrolilli & Gambardella, 2000), (Saidi-Mehrabad & Fattahi, 2006), (Fattahi et al., 2009), (J. Li et al., 2010), (Zhang, Manier, & Manier, 2012), (Sobeyko & Mönch, 2016), (Lei & Tan, 2016), (Shahgholi et al.,

2018)). However, due to the strict precedence relations of our operations within jobs, this is very illogical to do and leads many times to in-feasibility. Moreover, almost all operations can only be processed by one machine and thus changing the machine for an operation is not doable. Because of these two reasons, the neighbourhood operators mentioned above are more focused on changing the sequence of jobs to create neighbouring solutions. The first one, NO1, selects a job randomly and swaps this job with an adjacent job. This creates a close-related neighbour solution. NO2 also uses the swap operator, however, picks two jobs to swap with each other. This creates neighbour solutions that are a bit more random. The same holds for NO3, where a single job is placed somewhere random between two other jobs. NO4 uses the same operator as NO3, however, does this two times times before assessing the solution outcome. This strategy creates very different neighbour solutions.

Besides the individual neighbourhood operators, by combining the neighbourhood operators into a search strategy, diversification and intensification can be used even better to achieve more optimal solutions. Shen, Dauzere-Peres, and Neufeld (2018) for example, uses the number of iterations as a tool to diversify with a "high-level" operator before going to a "low-level" operator that is better at intensification of the solution. In our case, NO1 can make small changes to the sequence of jobs without changing the schedule too drastically. This is the intensification. NO2 and NO3 can make more random changes which leads to a bit more diversification. In the end, NO4 can make more drastic changes that completely change the sequence of jobs, which leads to even more diversification, and thus potentially escaping a local optimum.

The neighbourhood strategy that we use is to start with more diversification at the start of the improvement heuristic and gradually proceed to more intensification. In our case, we start with NO1, then progress to NO2 and N03, and end with NO4. Lei and Guo (2011), call this variable neighbourhood search. Figure 4.3 shows the pseudo code for our strategy, based on Lei and Guo (2011) and Shen et al. (Shen et al., 2018). The idea is that the neighbourhood operators are selected in a strategic order to intensify and diversify.

Neighbourhood strategy	
1	Initialise with a set of NOs (k=1,2,3,4) and stopping condition
2	Create an initial solution with constructive heuristic
3	Repeat the following until the stopping condition is met
5	k = 1
6	Repeat the following until k is not k_max
7	Generate solution with k
8	Apply improvement heuristic to the solution
9	If a better solution is found then continue with k = 1 otherwise k = k + 1

Figure 4.3: Neighbourhood strategy pseudo code.

To assess the performance of this strategy we also use a second strategy which chooses one of these four operators with equal probability at random in each stage of the improvement heuristic.

4.4.2 Iterative local search

Iterative Local Search (ILS) is the first improvement heuristic also explained in Section 3.4.4. It starts with the initial solution made by the constructive heuristic from Figure 4.2 and iteratively finds a new neighbouring solution with the operators from Section 4.4.1. When a neighbouring solution is better than the current one, the current one is updated, and when the neighbouring solution is also better than the best one found, the best found is also updated. This is done until a certain amount of iterations are completed. Appendix B.1 shows the parameter tuning process. Figure 4.4 shows the code for the ILS improvement heuristic. Line (1) shows the input needed for ILS. Line (2) uses the constructive heuristic from 4.2 to make an initial solution. Lines (3-5) initiate other inputs to be used in the heuristic. Lines (7-10) select the right neighbourhood strategy from Section 4.4.1. Line (11) finds a neighbour solution with the selected strategy. Lines (12-13) calculate the neighbour objective and the current objective. Lines (16-20) update the current solution and objective when the neighbour's objective is better than the current one, and update the best objective and solution when the neighbour is also better than the best found so far. Lines (15, 22-23) update the search strategy accordingly to go back to diversification or step into intensification.

ILS improvement heuristic	
1	**Data:** job_list, Iterations, NO_strategy
2	Solution ← Constructive_Heuristic(job_list)
3	Best_Solution ← Solution
4	Best_Objective ← Calculate_Cost(Best_Solution)
5	NO_count ← 1
6	**For** it in iterations **do**
7	**If** NO_strategy **then**
8	k ← NO_count
9	**Else**
10	k ← np.random.randint(1,5)
11	Neighbour_Solution ← Local_Search(Solution, NO_k)
12	Neighbour_Objective ← Calculate_Cost(Neighbour_Solution)
13	Objective ← Calculate_Cost(Solution)
14	**If** Neighbour_Objective < Objective **then**
15	NO_count ← 1
16	Solution = Neighbour_Solution
17	Objective = Neighbour_Objective
18	**If** Objective < Best_Objective **then**
19	Best_Solution = Solution
20	Best_Objective = Objective
21	**Else**
22	**If** NO_count <= 3 **do**
23	NO_count =+ 1
24	**Result** ← Best_Solution, Best_Objective

Figure 4.4: Code for the ILS improvement heuristic algorithm.

4.4.3 Simulated annealing

Simulated Annealing (SA) is the second improvement heuristic, which is also explained in Section 3.4.4. It starts with the initial solution made by the constructive heuristic from Figure 4.2. Different from ILS, SA can accept worse neighbour solutions with a certain chance. This change is based on the difference between the objective value of the neighbour and the current solution, but also on the stage of the heuristic, which is based on the temperature. SA always accepts a better neighbour solution, and the algorithm stops when a certain stopping temperature is reached. In the end, the best-found solution is returned. Appendix B.2 shows the parameter tuning process. Figure 4.5 shows the code for the SA improvement heuristic. Line (1) shows the input for the heuristic. Line (2) uses the constructive heuristic from 4.2 to make an initial solution. Lines (3-5) initiate other inputs to be used in the heuristic. Lines (8-11) select the neighbourhood strategy. Lines (11-12) create a neighbour solution and calculate the objective of this neighbour. Lines (14-19) assign either an acceptance probability of 1 if the neighbour is better (always accept a better neighbour) or calculate the acceptance probability based on the difference between the neighbour's objective and the current one, and the temperature (progression of the model). Lines (22-24) look if a random number is lower than the acceptance probability found. It shows that worse neighbour solutions can be accepted more often in earlier iterations and later on do not accept worse neighbours as the temperature decreases. Lines (25-27) update the best solution and objective if the neighbour would be better than the best found so far. Lines (15, 20-21) update the search strategy accordingly to go back to diversification or step into intensification. Line (28) updates the temperature.

	SA improvement heuristic
1	**Data**: job_list, markov_chain_length, temperature, cooling_rate
2	Solution ← Constructive_Heuristic(job_list)
3	Best_Solution ← Solution
4	Best_Objective ← Calculate_Cost(Best_Solution)
5	NO_count ← 1
6	**While** temperature > stopping_temperature **do**
7	**For** it to markov_chain_length **do**
8	**If** NO_strategy **then**
9	k ← NO_count
10	**Else**
11	k ← np.random.randint(1,5)
12	Neighbour_Solution ← Local_Search(Solution, NO_k)
13	Neighbour_Objective ← Calculate_Cost(Neighbour_Solution)
14	**If** Neighbour_Objective < Objective **then**
15	NO_count ← 1
16	acceptance_prob = 1.0
17	**Else**
18	Delta = Neighbour_Objective - Objective
19	acceptance_prob = exp(-delta / temperature)
20	**If** NO_count <= 3 **do**
21	NO_count =+ 1
22	**If** np.random.rand() < acceptance_prob **then**
23	Solution = Neighbour_Solution
24	Objective = Neighbour_Objective
25	**If** Objective < Best_Objective **then**
26	Best_Solution = Solution
27	Best_Objective = Objective
28	temperature = temperature * cooling_rate
29	**Result** ← Best_Solution, Best_Objective

Figure 4.5: Code for the SA improvement heuristic algorithm.

4.4.4 Tabu search

Tabu Search (TS) is the third improvement heuristic, which is also explained in Section 3.4.4. Similar to SA, TS can escape local optima. This is done by using a tabu list, which stores a certain amount of already found neighbour solutions. These solutions in the tabu list are not used until a certain criterion is met. This helps avoid going back to a similar solution space. Appendix B.3 shows the parameter tuning process. Figure 4.6 shows the code for the TS improvement heuristic. Lines (1-2) show the input for the heuristic. Line (3) uses the constructive heuristic from 4.2 to make an initial solution. Lines (4-6) initiate other inputs to be used in the heuristic. Lines (8-11) select the neighbourhood strategy. Line (12) finds a neighbour solution with the selected strategy. One difference is that this strategy generates a certain amount of neighbours first and selects the best one beforehand to get to more promising neighbours faster. Line (13) calculates the objective of this neighbour. Lines (16-17) update the current solution and objective if this neighbour is not in the tabu list and if it is better than the current solution. Lines (19-23) update the best solution and objective if the neighbour is also the best found so far, and update the tabu list accordingly by adding the neighbour and if needed deleting the first one on the list to be used again. Lines (15, 25-26) update the search strategy accordingly to go back to diversification or step into intensification. Line (27) updates the iteration count.

```
TS improvement heuristic
1   Initialise: tabu_list, iterations
2   Data: job_list, tabu_list_length, max_iterations
3   Solution ← Constructive_Heuristic(job_list)
4   Best_Solution ← Solution
5   Best_Objective ← Calculate_Cost(Best_Solution)
6   NO_count ← 1
7   While iterations < max_iterations do
8       If NO_strategy then
9           k ← NO_count
10      Else
11          k ← np.random.randint(1,5)
12      Neighbour_Solution ← Local_Search(Solution, NO_k)
13      Neighbour_Objective ← Calculate_Cost(Neighbour_Solution)
14      If Neighbour_Objective < Objective then
15          NO_count ← 1
16          Solution = Neighbour_Solution
17          Objective = Neighbour_Objective
18          If Objective < Best_Objective then
19              Best_Solution = Solution
20              Best_Objective = Objective
21              tabu_list =+ Neighbour_Solution
22              If len(tabu_list) > tabu_list_length then
23                  tabu_list.pop(0)
24      Else
25          If NO_count <= 3 do
26              NO_count =+ 1
27      iterations =+ 1
28  Result ← Best_Solution, Best_Objective
```

Figure 4.6: Code for the TS improvement heuristic algorithm.

4.5 Solution alternatives

With the improvement heuristics from Section 4.4 the initial solution can be made by using backward or forward scheduling together with a dispatching rule. The three dispatching rules are EDD, MS, and Random. Besides this, two strategies have been discussed in Section 4.4.1 to make neighbourhood solutions with the four provided neighbourhood operators. With all these options we can make several different solution approaches. Table 4.5 shows the 18 solution approach alternatives. Chapter 5 shows what solution approach is the best for our FJSP by doing experiments.

Table 4.5: Solution alternatives.

Solution approach	Dispatching rule	Improvement heuristic	Neighbourhood strategy
1	EDD	ILS	Normal
2	EDD	ILS	Random
3	EDD	SA	Normal
4	EDD	SA	Random
5	EDD	TS	Normal
6	EDD	TS	Random
7	MS	ILS	Normal
8	MS	ILS	Random
9	MS	SA	Normal
10	MS	SA	Random
11	MS	TS	Normal
12	MS	TS	Random
13	Random	ILS	Normal
14	Random	ILS	Random
15	Random	SA	Normal
16	Random	SA	Random
17	Random	TS	Normal
18	Random	TS	Random

4.6 Conclusion

This chapter has provided a thorough formulation of the FJSP specific to Kitchen 1. Initially, we identified the necessary input data required for our model, including processing times, due dates, transportation times, setup times, buffer times, and periods of unavailability. This input data is explained, setting the foundation for the problem formulation.

Furthermore, we presented a comprehensive problem description and a detailed mathematical formulation of the exact MILP. This included the decisions, assumptions, and simplifications inherent to the model, alongside the specific sets, parameters, and variables utilized. The constraints necessary to accurately represent the scheduling problem were also elaborated upon.

To address the inherent complexity of the scheduling problem, we developed a constructive heuristic. This heuristic was designed to create an initial feasible solution by applying dispatching rules to set the order of scheduling jobs, sequencing logic to give a certain precedence relation between operations within jobs, and assigning operators and machines to the different operations. Additionally, it considered setup times (cleaning) between different operations on the same machines and backward and forward scheduling. The jobs and operations are scheduled from the due date to the start time or from the start time to the end time of the schedule respectively.

Moreover, we expanded on this initial solution by introducing three improvement heuristics: ILS, SA, and TS. These heuristics aim to refine the initial solution through iterative processes, and using neighbourhood operators to explore potential improvements. The strategies for generating neighbourhood solutions were discussed, emphasising a balance between diversification and intensification to escape local optima for SA and TS, and approach near-optimal solutions.

Finally, we presented various solution alternatives by combining different dispatching rules, improvement heuristics, and neighbourhood strategies. These alternatives will be tested in Chapter 5 with experiments to determine

the most effective approach for optimising the scheduling process in Kitchen 1.

In summary, this chapter has laid a solid foundation for understanding and addressing FJSP. By combining precise mathematical modelling with strategic heuristic approaches, we are well-prepared to explore and evaluate various solution methods in the next chapter. This detailed approach aims to improve the efficiency and effectiveness of scheduling within Kitchen 1, ultimately leading to improved operational performance.

5 Experiments and results

This chapter aims to find the best solution approach for the FJSP in Kitchen 1 by experimenting with the different solution alternatives from Section 4.5. The following research question is answered:

'What solution approach performs best, and which experiments can be conducted using the model to explore this performance further?'

This chapter aims to find the best solution approach from Table 4.5. Section 5.1 determines the different problem instances which reflect the possible job lists in Kitchen 1. Section 5.2 gives an overview of the experiments to test the model performance. Section 5.3 gives insight into the results of the first three experiments. Section 5.4 provides a sensitivity analysis with the best solution alternative to test different real-world options. Section 5.5 compares the theoretical performance of the model versus the real-world performance to see how well the developed model performs compared to the current situation. Finally, Section 5.6 provides the conclusion of this chapter.

5.1 Problem instances

This section gives an overview of the problem instances used for the experiments. The instances consist of a job list, one of the model's main inputs. These job lists are extracted from company data of the past two production years and pre-optimised with one of the dispatching rules by using the Excel tool from Section 4.3.2. This tool determines the order of the jobs in the job list as input for the model. Table 5.1 shows the seven problem instances to experiment with. Every day and week is different in terms of variety and job quantity. By looking into the data, we can take the variety and quantity into account (the low job amount high variety and high job amount low variety instances do not exist). Variety means that there are more different jobs. For example, at one day there could be 100 jobs (recipes), but only 20 different jobs (so 10 jobs of one recipe for example), and at another day 100 jobs with 40 different jobs. These seven instances cover the job amount and job variety options occurring at Johma throughout the year.

The processing time is in minutes per job. Between brackets, the standard deviation is stated. The instances are based on job list data from the past production year and give a total representation of what scenarios can occur. Besides these problem instances, the available working time per day, and the forbidden time intervals are also taken into account per problem instance and are shown in Table 4.1 from Chapter 4. Furthermore, all jobs have due dates and three operators in the preparation department are used to compare the performance with the experiments. Three operators for the preparation department is standard for Kitchen 1 to do the work per shift.

Table 5.1: Problem instance-specific information.

Instance	Job amount / variety	Total jobs	Unique jobs	Total operations	Operations per job	Operator processing time	Machine processing time
1	High/ high	147	40	1100	7.48 (2.64)	34.96 (22.14)	23.41 (20.61)
2	High/ normal	148	27	1164	7.86 (2.97)	36.15 (21.61)	23.35 (20.41)
3	Normal/ high	94	32	755	8.03 (3.26)	38.73 (24.17)	26.55 (21.03)
4	Normal/ normal	96	26	669	6.97 (2.58)	30.09 (14.19)	16.99 (13.21)
5	Normal/ low	107	19	792	7.40 (2.74)	34.11 (19.55)	20.86 (18.99)
6	Low/ normal	77	24	593	7.70 (2.85)	37.25 (23.48)	24.57 (21.67)
7	Low/ low	78	17	635	8.14 (2.66)	37.93 (22.25)	24.12 (22.28)

5.2 Experimental design

This section determines the experimental design to test the different solution alternatives to evaluate the performance of the heuristics and the effects of parameter changes in real-world situations. The experiments determine which constructive heuristic solution approach performs the best, the parameter tuning is done to see which settings are best for the improvement heuristics, the second set of experiments analyse the performance of the improvement heuristic solution alternatives, and the sensitivity analysis provides insights in the effect of parameter changes in different real-world situations.

- **Experiments: Evaluating solution alternatives - Constructive heuristics**

  - This experiment aims to see which constructive heuristic, in particular, which scheduling approach (backward or forward) together with which dispatching rule (EDD, MS, or Random) performs best with the problem instances from Section 5.1.

- **Parameter tuning - Improvement heuristics**
  - This experiment aims to find the right parameter settings for the ILS, SA, and TS improvement heuristics.

- **Experiments: Evaluating solution alternatives - Improvement heuristics**
  - This experiment aims to determine what improvement heuristic performs best in combination with the constructive heuristics. All solution alternatives from Table 4.5 are compared to each other.

- **Experiment: Sensitivity analysis - Operator capacity**
  - This experiment aims to evaluate the effect of operator capacity on the total performance of the schedule. This experiment provides insight into how many operators are needed.

- **Experiment: Sensitivity analysis - Machine capacity**
  - This experiment aims to evaluate the effect of machine capacity on the total performance of the schedule. This experiment provides insights into whether Johma could use one mixing machine or needs to use two.

- **Experiments: Sensitivity analysis - Effect of buffer time on schedule robustness**
  - This experiment aims to evaluate the effect of more buffer time to make the schedule more robust on the total performance of the schedule. This experiment provides insights into whether more buffer time is needed due to new operators and if less buffer time is required due to operator training. Furthermore, higher buffer times create a more robust schedule which is more resistant to the dynamic environment.

5.3 Results of experiments

This section provides the results of the first three experiments. Section 5.3.1 provides the results of the constructive heuristics and determines which one performs the best. Section 5.3.2 explains how the parameters for the improvement heuristics are tuned. To conclude, Section 5.3.3 provides an overview of all the solution alternatives and explains which one is the best to use for the sensitivity analysis in Section 5.4.

5.3.1 Constructive heuristic results

Figure 5.1 shows the results of running all the constructive heuristic options with the different instances from Table 5.1. The run time is in seconds, the tardiness is in minutes, and the tardiness at the random dispatching rule is the average of five runs to handle the random outcomes.

Forward scheduling is better with either the EDD or MS dispatching rule for all instances (see green boxes in Figure 5.1). Sometimes the EDD rule is better and sometimes the MS rule is better. With instances 4, 5, and 6 the optimal solution is found with the backward and forward model (see green boxes in Figure 5.1). The backward model under-performs or performs equally compared to the forward model using all instances. This is because with the larger instances (1, 2, and 3) the capacity of three preparation operators is insufficient to handle all the jobs and operations. With backward scheduling, the operations are scheduled from the due date backwards. If there is insufficient capacity (operations are scheduled into negative time), all operations are shifted forward to start at time zero leading to tardiness for most of the scheduled operations. The forward model schedules operations from time zero into the future. This leads to a more optimal schedule than backward scheduling, as the schedule is more compressed. Instance 7 is also small, however, the difference with the other smaller instances is that the due dates for the jobs in this instance are very spread out. This leads to a stretched schedule with the backward model, and a more compressed and more optimal schedule with the forward model.

The forward model is the best compared to the backward model as a constructive heuristic. Moreover, in most cases, the EDD dispatching rule is the best or comparable with MS. The random rule is terrible when capacity becomes more sufficient and thus unusable for solving our problem. In Section 5.3.2 both the backward and forward models with the EDD rule are used to tune the parameters for the improvement heuristics.

Instance	Scheduling approach	Dispatching rule	Tardiness	Run time
1	Backward	EDD	67,202.77	4.41
		MS	63,153.61	4.46
		Random	56,501.9	4.66
	Forward	EDD	44,751.26	4.12
		MS	41,756.7	4.05
		Random	67,726.5	4.34
2	Backward	EDD	81,320.77	4.71
		MS	83,164.51	4.71
		Random	49,882.62	4.97
	Forward	EDD	45,740.84	4.39
		MS	48,168.15	4.37
		Random	58,804.49	4.63
3	Backward	EDD	5,200.63	2.97
		MS	12,494.97	2.9
		Random	26,212.63	3.17
	Forward	EDD	497.4	2.78
		MS	1,651.31	2.74
		Random	26,206.25	3.02
4	Backward	EDD	0.0	2.72
		MS	1,124.09	2.64
		Random	13,311.87	2.91
	Forward	EDD	0.0	2.49
		MS	0.0	2.5
		Random	17,750.13	2.77
5	Backward	EDD	0.0	3.06
		MS	0.0	2.95
		Random	13,688.62	3.13
	Forward	EDD	0.0	2.82
		MS	0.0	2.81
		Random	14,622.15	3.06
6	Backward	EDD	0.0	2.31
		MS	0.0	2.33
		Random	10,134.49	2.5
	Forward	EDD	0.0	2.44
		MS	0.0	2.3
		Random	10,329.02	2.89
7	Backward	EDD	1,771.32	2.15
		MS	2,923.01	2.28
		Random	5,643.63	2.45
	Forward	EDD	260.06	2.77
		MS	127.34	2.64
		Random	6,318.26	2.79

Figure 5.1: Results of the constructive heuristic algorithms (tardiness in minutes and run time in seconds).

5.3.2 Parameter tuning

The improvement heuristics from Section 4.4 perform better when the parameters used are determined correctly. For ILS the number of iterations is determined. We experiment with instance 1 because this is the largest and most variable job list that the model should be able to handle. Additionally, the random neighbourhood operator strategy is used with the number of replications set to 5. Furthermore, we experiment with the number of iterations equal to 10, 100, 200, 500 and 1,000 in the ILS improvement heuristic. Appendix B.1 explains the parameter-tuning process in more detail. Figures 5.2 and 5.3 shows the results of the parameter tuning process. For both backward and forward 500 iterations are chosen as the outcome is the best or comparable with more iterations, while maintaining a reasonable run time.

Instance	ILS Iterations	Dispatching rule	Tardiness	Run time
1	10	EDD	62,372.13	46.58
	100	EDD	43,983.57	389.82
	200	EDD	40,508.71	750.78
	500	EDD	35,546.62	1,966.72
	1000	EDD	32,041.21	3,731.94

Figure 5.2: Parameter tuning results backward model ILS.

Instance	ILS Iterations	Dispatching rule	Tardiness	Run time
1	10	EDD	41,692.24	43.26
	100	EDD	22,877.32	381.41
	200	EDD	21,892.64	758.11
	500	EDD	14,835.48	1,886.66
	1000	EDD	18,117.78	3,768.11

Figure 5.3: Parameter tuning results forward model ILS.

For SA the starting temperature, the stopping temperature, the Markov chain length, and the temperature decrease factor alpha are determined. We start with determining the starting temperature. This is done with the backward and forward scheduling models with instance 1 and the EDD dispatching rule. The stopping temperature is set to 1.0, the cooling rate to 0.85 and the Markov chain length to 100. Besides this, the initial starting temperature is set to 100. By looking at the acceptance ratio, which is the ratio of accepted neighbour solutions that are worse than the current solution, we can see what temperature would be a good start temperature to in the beginning have more diversification (higher ratio) and to further on in the heuristic have more intensification (lower ratio) (Pandey & Gajendran, 2016) (Battistutta, Schaerf, & Urli, 2015). For this experiment, we use makespan as the objective, because tardiness is always zero and thus no other worse neighbours are found. Appendix B.2 explains the starting temperature parameter-tuning process in more detail. Figures 5.4 and 5.5 show the graphs of the acceptance ratio plotted against the temperature. We can see that at a temperature of 52 for the backward model and 61 for the forward model the plotted ratio goes down fast, which means there is less diversification after this temperature. Because of this, we choose 52 and 61 as the starting temperatures for the backward and forward models respectively, to still have enough diversification at the start but not too much running time.

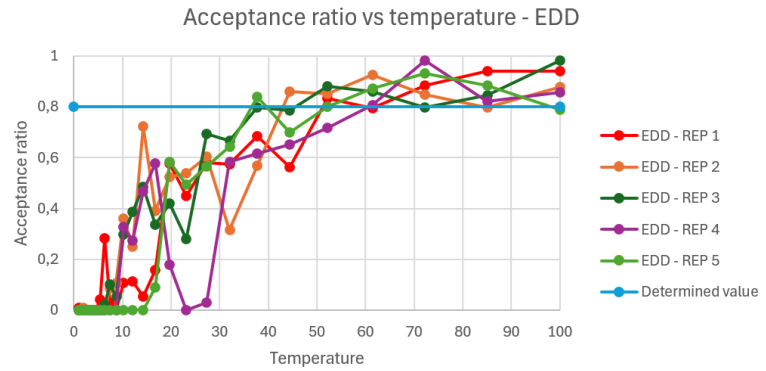


Figure 5.4: Starting temperature tuning backward SA with EDD.

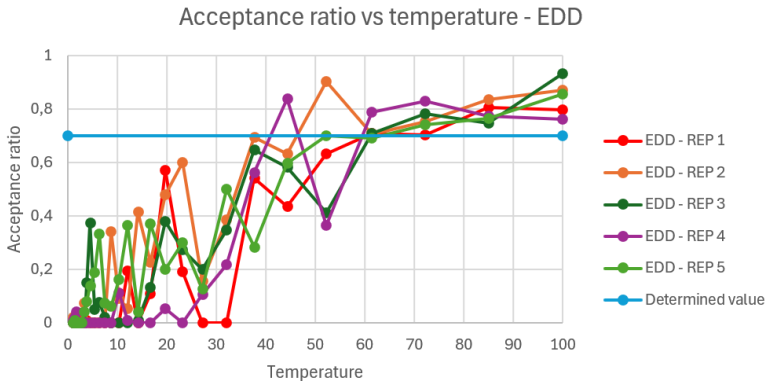


Figure 5.5: Starting temperature tuning forward SA with MS.

With the start temperature determined, we experiment with the stopping temperature, the cooling rate, and the Markov chain length. We again use the backward and forward scheduling models with the EDD dispatching rule and instance 1. Moreover, we experiment with three different values per parameter. Appendix B.2 explains the parameter-tuning process in more detail. From these experiments, we can conclude that for the backward model a stopping temperature of 5, a cooling rate of 0.95, and a Markov chain length of 20 will result in the lowest tardiness. For the forward model this is with a stopping temperature of 5, a cooling rate of 0.95, and a Markov chain length of 10.

For TS the maximum number of iterations, the tabu list length, and the number of initial neighbour evaluations need to be determined. Again the same modelling approach is used as at ILS and SA, and three options are determined per parameter to experiment with. Appendix B.3 explains the parameter-tuning process in more detail. From these experiments, we can conclude that for the backward model with the number of maximum iterations at 75, a Tabu list length of 5, and the number of evaluated neighbours at 15 will result in the lowest tardiness. For the forward model, this is with the number of maximum iterations at 75, a Tabu list length of 15, and the number of evaluated neighbours at 15.

The best parameter settings for the improvement heuristics are determined. With this, all the solutions alternatives from section 4.5 are analysed in the following section to see which solution approach performs best.

5.3.3 Improvement heuristic results

This section provides an overview of the performance of the solution approaches from Section 4.5. Therefore we have 18 solution approaches in total. We use the parameter settings that resulted from the previous section. For ILS we use 500 iterations for both the backward and forward scheduling model. For SA we use a starting temperature of 52 and 61, a cooling rate of 0.95 and 0.95, a Markov chain length of 20 and 10, and a stopping temperature of 5 and 4 respectively. Moreover, for TS we set the maximum number of iterations to 75 and 75, the Tabu list length to 5 and 15, and neighbours to evaluate to 15 and 15 for the backward and forward model respectively. With these settings, we run all solution approaches for all seven instances and per instance, we run three replications to handle randomness.

Figure 5.6 shows the tardiness results in minutes of all solution alternatives with the improvement heuristics. We can see from bars 13-18 and 31-36 that the Random dispatching rule results in worse solutions than the EDD and MS dispatching rules furthermore, when we compare the backward and forward scheduling approach, the forward approach, bars 19-30, results in better tardiness in general than with the backward approach, bars 1-12. Moreover, the forward model with the EDD dispatching rule, bars 19-24, is very comparable with the results of the forward model with the MS dispatching rule, bars 25-30. Only the TS improvement heuristic with the EDD dispatching rule seems to be slightly better. When we compare the neighbourhood strategies of the forward model with the EDD dispatching rule and TS, bars 23-24, we can see that with the Normal rule, we have one outlier, but in general, both do not differ substantially. Figure 5.7 shows the makespan results in minutes for all solution alternatives. The figure clearly shows that the forward model, bars 19-36, results in lower makespan in general. For all solution alternatives, this is very comparable. Figure 5.8 shows the run time results in seconds. It shows that TS, bars 5, 6, 11, 12, 17, 18, 23, 24, 29, 30, 35, and 36 have a higher run time in general compared to ILS and SA. However, also discussed in Section 5.3.2 this run time is still usable.

We can conclude that the best solution approach is the forward scheduling approach with the TS improvement heuristic, the EDD dispatching rule, and either the Normal or Random neighbourhood structure. We achieve reasonable running times, a comparable low makespan, and the lowest tardiness. There are no substantial differences between the Normal or Random neighbourhood strategy.

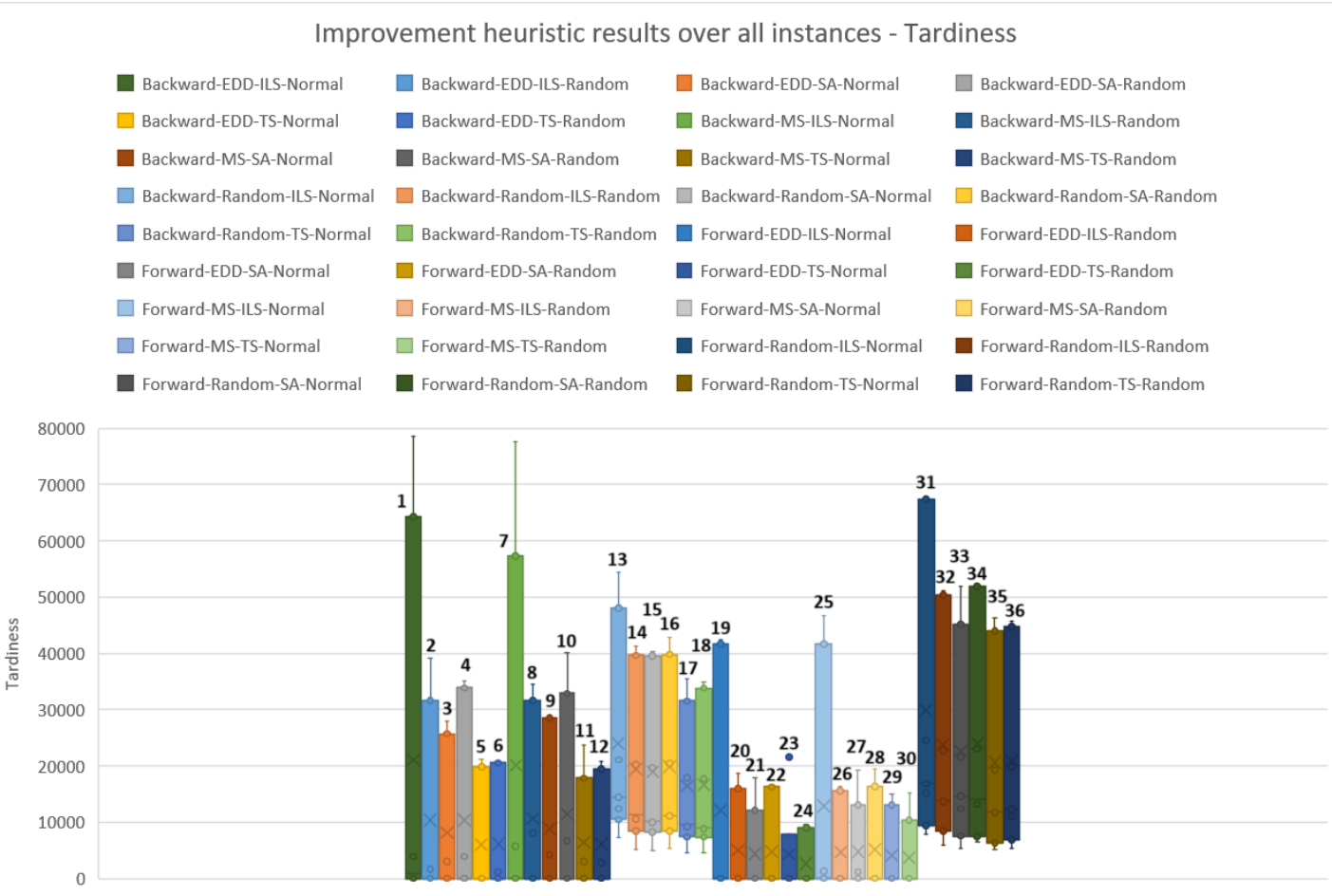


Figure 5.6: Improvement heuristics tardiness results.

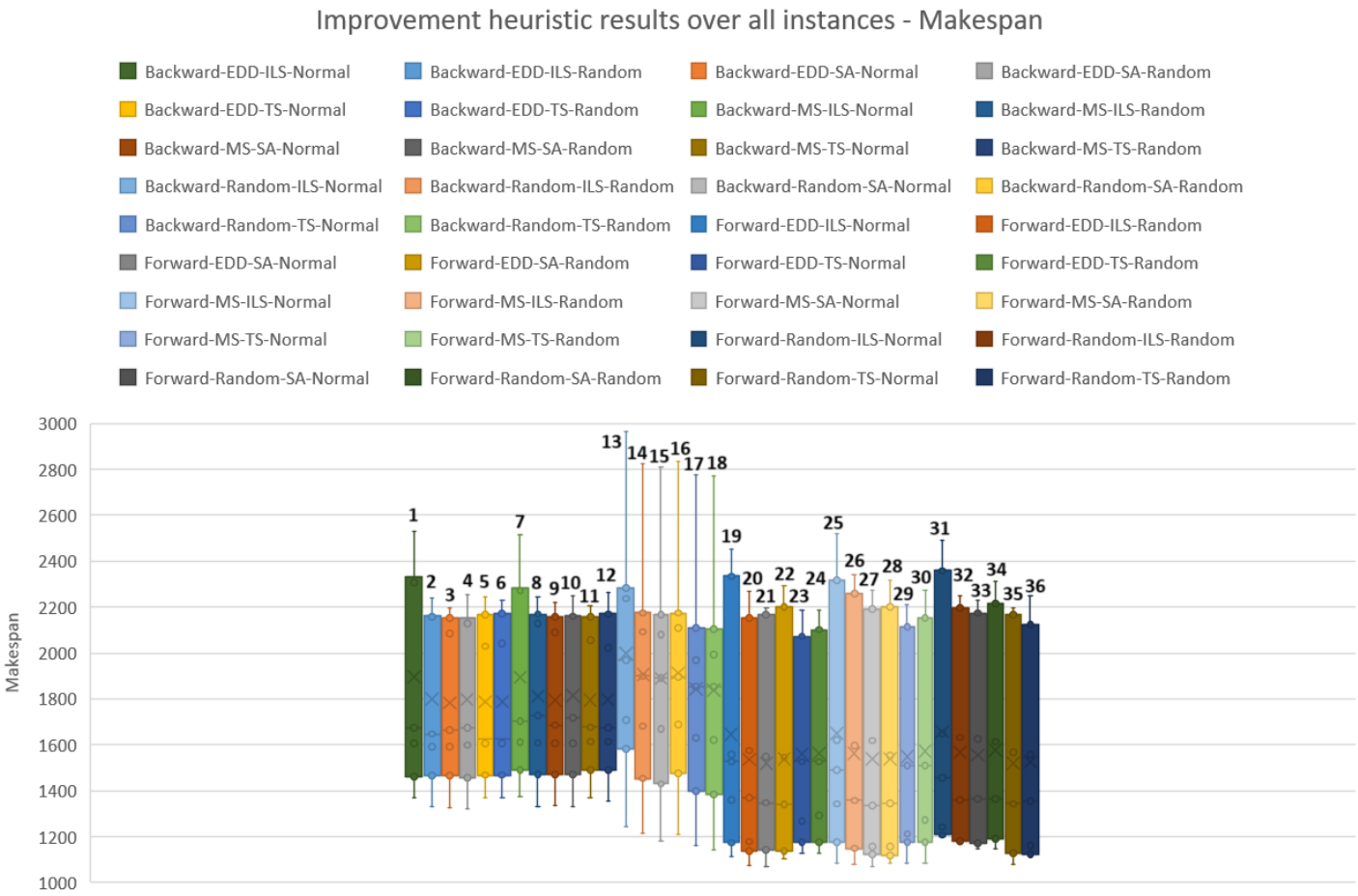


Figure 5.7: Improvement heuristics makespan results.

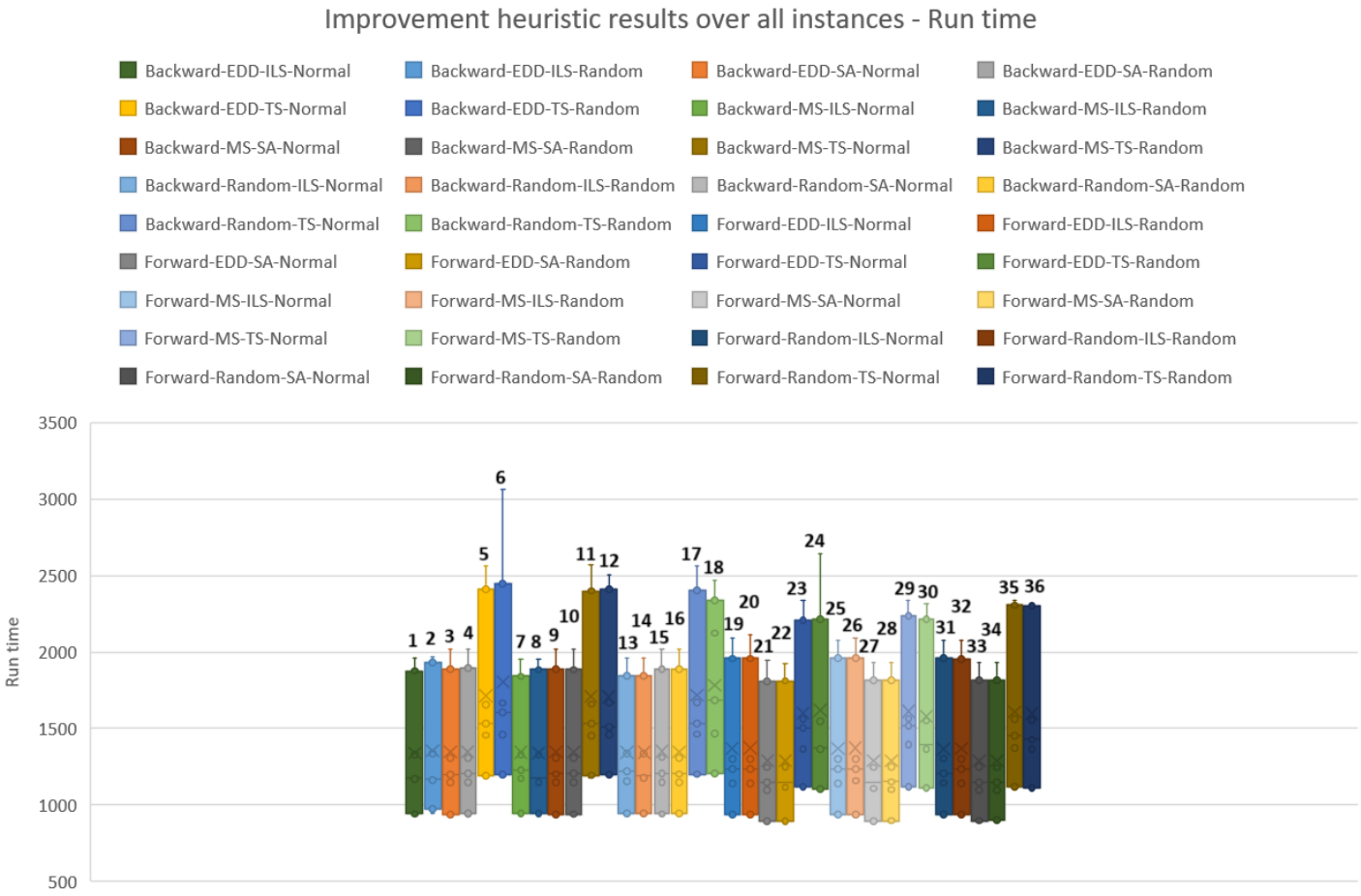


Figure 5.8: Improvement heuristics run time results.

5.4 Sensitivity analysis

This section provides the sensitivity analysis. The following sub-sections test several scenarios compared to the basic real-world scenario, which has three preparation operators, two mixers, and the buffer time measured on the shop floor. In addition, we use the best-performing solution approach from Section 5.3.3, which is the forward scheduling approach with TS and EDD. Because there are no clear differences between the neighbourhood strategies, we chose the Random neighbourhood strategy to not have a bias. In Section 5.4.1 we test the scenarios with extra operators, in Section 5.4.2 we test the scenario with only one mixer, and in Section 5.4.3 we test with less and with more buffer time in between operations.

5.4.1 Operator capacity

Chapter 2 explained that Kitchen 1 consists of a mixing and preparation department that works together to manufacture the salad mixtures. The number of operators in the mixing department is fixed. Only one operator is required to operate the mixers and one is needed to add the wet spices. However, the need for more or fewer operators in the preparation department greatly depends on the amount of work. Because of this, we experiment with four, and five operators in the preparation department for the different instances to see the effect on the performance.

Table 5.2 shows the results. By adding one extra operator, we can see that the tardiness becomes zero for all instances. Furthermore, the makespan decreases between 4% and 25%. The improvement heuristic that we use already creates optimal solutions for tardiness for instances 4, 5, 6, and 7. This means that for these instances three operators for preparation would be enough. For instances 1, 2, and 3, which are bigger instances, the need for an extra preparation operator is necessary to reach optimal tardiness.

Table 5.2: Effect of an extra preparation operator.

Instance	3 (Basic scenario)		4			
	Tardiness	Makespan	Tardiness	Makespan	Gap T	Gap M
1	9828.04	2069.63	0.00	1739.90	-100.00%	-15.93%
2	11947.50	2287.93	0.00	1728.95	-100.00%	-24.43%
3	48.85	1512.57	0.00	1326.60	-100.00%	-12.29%
4	0.00	1129.63	0.00	918.87	-0.00%	-18.66%
5	0.00	1526.68	0.00	1255.48	-0.00%	-17.76%
6	0.00	1176.32	0.00	996.00	-0.00%	-15.33%
7	0.00	1252.23	0.00	1199.92	-0.00%	-4.18%

5.4.2 Machine capacity

The mixing department in Kitchen 2 consists of three mixers; two big ones and one small one. The small mixer is not used for daily operations and is thus left out of the scope of this research. The two big mixers can be used both to mix the provided salad sub-mixtures from the preparation department. Currently, the mixing operator can use both mixers, however this is not always done. Because of this, we want to see what effect using only one mixer will have on the performance.

Table 5.3 shows the results. We can see that by only using one mixer the tardiness and makespan do not increase that much in absolute value, it even decreases in some cases. According to the current planning of recipes of the filling lines, we cannot say if using only 1 mixer makes the performance worse by using the schedule produced by the improvement heuristic.

Table 5.3: Effect of using only one mixer.

Instance	2 (Basic scenario)		1			
	Tardiness	Makespan	Tardiness	Makespan	Gap T	Gap M
1	9828.04	2069.63	8904.07	2149.62	-9.40%	3.86%
2	11947.50	2287.93	11935.00	2228.27	-0.10%	-2.61%
3	48.85	1512.57	142.37	1553.97	191.43%	2.74%
4	0.00	1129.63	0.00	1138.17	-0.00%	0.76%
5	0.00	1526.68	0.00	1526.68	-0.00%	-0.00%
6	0.00	1176.32	0.00	1177.00	-0.00%	0.06%
7	0.00	1252.23	0.00	1321.53	-0.00%	5.53%

5.4.3 Schedule robustness

The production environment at Johma is very dynamic and many things can happen that are not accounted for beforehand. Because of this, we added extra buffer time for operations to be able to handle unexpected events. In this experiment, we want to see what will happen if we have more or less buffer time. In other words, we want to see what happens if fewer of these unexpected events happen due to mitigation in the future, and what happens if the flex workers are trained and thus can operate faster without the need for assistance. The other way around is also interesting to experiment with. We want to see what happens if no good operators can be obtained and continuous assistance is needed.

Figure 5.9 shows the 3 scenarios, namely basic as measured on the shop floor, lower, and higher buffer time. All results are in minutes. We see in Table 5.4 that with less buffer time the tardiness and makespan increases for instance 1, which we would not assume beforehand because we have less time that needs to be scheduled. However, we do see a decrease in tardiness and makespan for instance 2, and a decrease in tardiness for instance 3. We cannot for certain say if less buffer time creates more or less tardiness and makespan. We see in Table 5.5 that for instance 2 the tardiness and makespan decreases, which we would not assume because time is added. In general, we cannot say if adding or decreasing buffer time affects the schedule performance substantially in a logical way. Because of this, more buffer time would not be a problem for the current situation at Kitchen 1.

Buffer time code	Task(s) included	Basic	Lower	Higher
1	cleaning knife, throw away garbage, new task	2.00	1.67	2.33
2	new task	0.50	0.33	0.67

Figure 5.9: Buffer time scenarios.

Table 5.4: Effect of having less buffer time.

Instance	Basic		Lower			
	Tardiness	Makespan	Tardiness	Makespan	Gap T	Gap M
1	9828.04	2069.63	11573.59	2131.38	17.76%	2.98%
2	11947.50	2287.93	9597.76	2154.55	-19.97%	-5.83%
3	48.85	1512.57	11.98	1526.62	-75.47%	0.93%
4	0.00	1129.63	0.00	1129.63	0.00%	0.00%
5	0.00	1526.68	0.00	1526.68	0.00%	0.00%
6	0.00	1176.32	0.00	1176.32	0.00%	0.00%
7	0.00	1252.23	0.00	1257.32	0.00%	0.41%

Table 5.5: Effect of having more buffer time.

Instance	Basic Tardiness	Makespan	Higher Tardiness	Makespan	Gap T	Gap M
1	9828.04	2069.63	11956.60	2106.45	21.66%	1.78%
2	11947.50	2287.93	11040.30	2233.17	-7.59%	-2.39%
3	48.85	1512.57	53.05	1557.03	8.60%	2.94%
4	0.00	1129.63	0.00	1129.63	0.00%	0.00%
5	0.00	1526.68	0.00	1526.68	0.00%	0.00%
6	0.00	1176.32	0.00	1203.92	0.00%	2.35%
7	0.00	1252.23	0.00	1255.67	0.00%	0.27%

5.5 Model performance

The previous sections showed that the model can achieve zero tardiness with a feasible makespan to execute all operations for all scenarios with certain operator settings. Figure 5.10 shows that our constructive heuristic already performs very well compared to the best improvement heuristic for instances 4, 5 and 6 as the improvement heuristic can only improve a small percentage of the makespan. For instances 1, 2, 3 and 7 the improvement heuristic performs up to 100% percent better, and massively better in absolute values for instances 1 and 2. Thus some alterations of the job list order result in a better schedule, however, because of the due date structure and the well-performing constructive heuristic the improvement heuristic performs only a lot better with larger instances (1 and 2) or instances where the due dates are far apart from each other (7).

	Basic heuristic Forward - EDD			*Improvement heuristic TS - Forward - EDD - Random*				
Instance	**Tardiness (min)**	**Makespan (min)**	**Run Time (sec)**	**Tardiness (min)**	**Makespan (min)**	**Run Time (sec)**	**Gap T**	**Gap M**
1	44751.26	2360.53	4.45	9828.04	2069.63	2406.93	-78.04%	-12.32%
2	45740.84	2500.87	4.75	11947.50	2287.93	2830.26	-73.88%	-8.51%
3	497.40	1561.33	2.97	48.85	1512.57	1648.15	-90.18%	-3.12%
4	0.00	1132.13	2.74	0.00	1129.63	1540.83	-0.00%	-0.22%
5	0.00	1529.18	2.92	0.00	1526.68	1661.95	-0.00%	-0.16%
6	0.00	1178.82	2.42	0.00	1176.32	1223.25	-0.00%	-0.21%
7	260.06	1382.38	2.44	0.00	1252.23	1211.64	-100.00%	-9.41%

Figure 5.10: Model performance.

Based on these findings, the question remains how does the scheduling model perform based on the current schedule in Kitchen 1? As explained in Section 2.4.2, Johma does not have a scheduling method in Kitchen 1. The planning of the filling lines only indicates when the full batch of salad mixtures has to be ready for a certain filling line, and no data is saved on the operations in Kitchen 1. Furthermore, the planning for the filling lines changes several times during the day, which shows that testing pre-made planning with the realised planning is not a valid experiment. Moreover, our model uses the due dates as input and uses this as a reference point to schedule from. When we give the realised due dates of the filling as input, we do not get any relevant insight into the performance of Kitchen 1 because much can change at the filling lines.

The focus of this research is on Kitchen 1, and performance is currently only visible at the filling lines. Therefore, it is not possible to measure the performance directly with real-world data. The newly developed model purely shows that it would be possible to decrease tardiness drastically with a feasible makespan with certain settings of operators, mixers, and buffer time, and gives guidance on when to start with what recipe on the shop floor.

5.6 Conclusion

This chapter has presented an analysis of the experiments conducted to evaluate the performance of different solution approaches for the FJSP specific to Kitchen 1. Initially, we defined several problem instances that accurately reflect the real-world scenarios encountered in Kitchen 1. By looking at job variety and quantity, these instances provided the basis for testing the effectiveness of our scheduling methods.

Furthermore, we executed experiments using constructive heuristics to establish baseline solutions for each problem instance. The forward scheduling method together with the EDD dispatching rule resulted in the best.

To enhance the performance of these initial solutions, parameter tuning is done for the ILS, SA, and TS improvement heuristics. This involved extensive testing to determine optimal parameter settings, ensuring that each heuristic was fine-tuned for maximum effectiveness.

With the parameters determined the improvement heuristic performances are tested. The forward scheduling approach with the TS improvement heuristic with the EDD dispatching rule performs the best out of all solution alternatives. No major differences are concluded from the use of either the Normal or Random neighbourhood strategy.

In addition, a sensitivity analysis is conducted to explore the impact of various parameters on scheduling performance. This analysis included factors such as operator capacity, machine capacity, and schedule robustness, providing valuable insights into their effects on the overall efficiency of the schedule. By adding an operator the tardiness is optimal and the makespan is decreased greatly. Moreover, using only one mixer does not affect the outcome substantially for all instances, and higher buffer times can be used while having no substantial effect on the scheduling performance.

Finally, we compared the performance of our model in Kitchen 1 to highlight performance increases and potential areas for improvement. Although direct comparison with real-world performance in Kitchen 1 is not possible, the model demonstrated its potential to drastically reduce tardiness and provided a structured approach to scheduling operations. This underscores the importance of implementing a more sophisticated scheduling method in Kitchen 1.

In summary, this chapter has established a solid foundation for understanding and addressing the scheduling challenges in Kitchen 1. By experimenting with the heuristic approaches, we have identified effective solutions that can significantly enhance operational performance. The insights gained from these experiments will be important in transitioning to a more efficient scheduling approach, as discussed in the subsequent chapter.

6 Usage and measures plan

This chapter describes how Johma can transition to the operational usage of the model. The following research question is answered:

'What measures can Johma adopt to ease the transition to the proposed scheduling approach?'

Section 6.1 describes how to set up the input data to use the planning model in Python. Section 6.2 outlines the steps to use the planning model. Furthermore, Section 6.3 explains how the output can be interpreted and used in practice. At last, Section 6.4 is dedicated to how to deal with an ever-changing organisation and how this impacts the implementation and usability of the model.

6.1 Model input

Section 4.1 described what input is needed to run the model. Most data is measured on the shop floor, like operation process times for operators and machines, setup times for cleaning, transport times of ingredients, and buffer times between operations. Precedence relations of operations within jobs, machine-worker (operator) eligibility, forbidden work intervals, and most importantly the job list planning for Kitchen 1 for the filling lines are gathered from Johma's database or employees. This input data is all combined into one Excel file and forms the main input for the Python model. The only input that has to be manually changed is the input job list. Appendix C.1 explains how to do this per type of job list.

The measured input data is subject to change as new machines can be bought, operators can work more efficiently in the future, new recipes are added, and ingredients change. To make sure the input data is relevant, it should be updated accordingly. Appendix C.2 explains how to do this.

This method is still extensive and requires many manual steps in Excel, which can be time-consuming and errors can easily be made. However, Johma is currently developing an ERP system which could use elements of this planning model or any knowledge gathered from this. The front-end and back-end designs are still unknown, and because of this, we do not further develop the automation of input. We solely focus on the development of the model to test the possibilities.

6.2 Model usage

The main model is made in Python as this is a well-known programming language used a lot worldwide. However, the employees at Johma are not used to programming in Python at all. For accessibility and usability, a manual is made for Johma that explains how to use Python for our model.

With the input documents, the installation of Python, and the explanation of how to use this, the results are stored in an Excel file that can be used as an operational schedule for the operations in Kitchen 1. Section 6.3 explains how to interpret these results.

6.3 Output and interpretation

The model in Python generates two types of output; the list of scheduled operations and the Gantt Chart, which is the graphical representation of the scheduled operations. Figure 6.1 shows the output, which is the operational planning of a certain day. The first column represents the recipe number, the second column the ingredient number, the third column the machine, the fifth column the operator, the sixth column the start time of the operation, the seventh column the finish time of the operation and the eighth column the due date of the operation. Table 6.1 shows the interpretation of the machine and operator numbers from Figure 6.1.

Recipe Name	Ingredient Name	Job Number	Operation Number	Machine	Machine duration	Operator	Operator duratio	Finish Time	Start Time Machine	Start Time Operator
R115	VU EIEREN E BL2 C/P TANK	9	20	nan	nan	4	2.5	2353.7	nan	2351.2
R993	VU EIEREN E BL2 C/P TANK	89	20	nan	nan	4	2.5	2358.43	nan	2355.93

Figure 6.1: Scheduled operations list.

Table 6.1: Information on operator and machine number.

Number	Operator	Machine
1	Preparation 1	Can opener
2	Preparation 2	Urschel
3	Preparation 3	Holac
4	Runner	Pilling machine
5	Wet spices	Egg machine
6	Mixing	Vibrating sieve
7	n.a.	Large cutter
8	n.a.	Small cutter
9	n.a.	Mixer 1
10	n.a.	Mixer 2

Figure 6.2 shows the corresponding Gantt Chart. Because of the large number of operations, it does not give a clear representation, however, it can give some insights into which machines, and operators are busier during the day. The grey rectangles are the forbidden time intervals where breaks or changes take place, the coloured rectangles per operator and machine are the individual operations, and the rectangles per operator and machine with an 'S' means there is a cleaning operation. One thing to note is that the c-operator, who decides what happens in Kitchen 1, is left out of the schedule. This is because, he or she should manage the tasks and operators, and this is also what Johma wants.

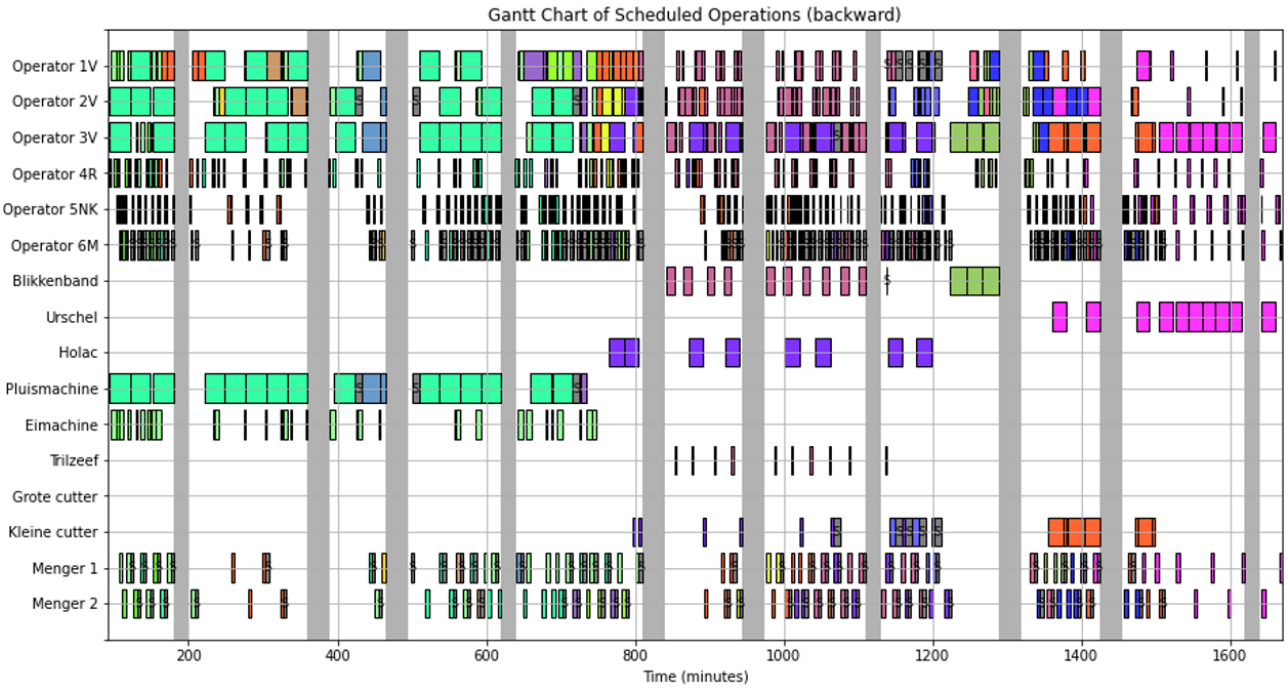


Figure 6.2: Gantt Chart of scheduled operations.

6.4 Dealing with change

This section outlines the challenges when implementing something into an organisation. Before being able to implement something new or even considering doing this, certain things have to be taken into account. Most humans do not like change and there are several reasons why. Some of these reasons are fear of the unknown, loss of control, attachment to the past, perceived risk, cognitive biases, disruption of routines, and getting out of the comfort zone.

To mitigate the above-mentioned points, change management is needed. Change is inevitable, and businesses are required to adapt to the continuously changing environment to maintain competitiveness (Appelbaum, Habashy, Malo, & Shafiq, 2012). However, the implementation of new projects can have a failure rate of up to 80% when not done correctly. One of the most well-known and widely used models for dealing with change

is John Kotter's Eight-Step Model. Kotter's model organizes these steps into three distinct phases (Campbell, 2008):

- **Creating a Climate for Change (Steps 1-3)**
- **Engaging and Enabling the Whole Organization (Steps 4-6)**
- **Implementing and Sustaining the Change (Steps 7-8)**

These steps can be explained the following:

- ***Step 1: Establishing a Sense of Urgency:*** The first step in Kotter's model is to create a sense of urgency about the need for change. Employees must understand why change is necessary; otherwise, efforts to initiate change will lack the required momentum and credibility. This urgency can be supported by clearly presenting the current challenges, potential opportunities, and the risks of not changing (Pollack & Pollack, 2015). Utilizing external consultants can also help confront the current norms and reinforce the message (Appelbaum et al., 2012).
- ***Step 2: Creating a Guiding Coalition:*** As a sense of urgency grows among employees, managers must turn their attention to developing a guiding team with a well-defined skill set (Campbell, 2008). No one person can lead and manage a change process alone. Forming a guiding coalition of influential and motivated individuals is crucial. This coalition should possess:
  - *Position Power:* Enough key players to prevent obstruction.
  - *Expertise:* Diverse and relevant viewpoints for informed decision-making.
  - *Credibility:* Respect and trust within the organization.
  - *Leadership:* Ability to drive and sustain the change effort.
- ***Step 3: Developing a Vision and Strategy:*** A clear and convincing vision guides the transformation effort and aligns all stakeholders. This vision should express the desired future state, explain why the change is necessary, and outline the strategy for achieving it. A well-defined vision prevents confusion and ensures that all efforts are directed towards a common goal (Pollack & Pollack, 2015).
- ***Step 4: Communicating the Change Vision:*** Effective communication is essential to reduce uncertainty and gain "buy-in" from employees. The vision for change must be communicated consistently and frequently across all channels. Satisfied employees, who understand the vision and its benefits, are more likely to support the change. Regular team meetings and open discussions can further enhance understanding and reduce resistance (Pollack & Pollack, 2015) (Appelbaum et al., 2012).
- ***Step 5: Empowering Broad-Based Action:*** Empowering employees involves removing barriers that hinder change, such as outdated structures, skills gaps, and unsupportive systems. Providing training, coaching, and encouraging innovation helps employees feel responsible and in control of the change process (Appelbaum et al., 2012). Empowered employees are more likely to contribute positively to the change effort.
- ***Step 6: Generating Short-Term Wins:*** Achieving and celebrating short-term wins is vital for building momentum and demonstrating the viability of the change. These early successes boost confidence and show that the efforts are yielding results. Recognizing and rewarding contributions also reassures employees that the change is on the right track (Pollack & Pollack, 2015) (Appelbaum et al., 2012).
- ***Step 7: Consolidating Gains and Producing More Change:*** After initial successes, it is important to build on the momentum and continue driving further changes. This involves using the credibility gained from early wins to tackle additional challenges and reinforce the new approaches. Sustaining change requires ongoing effort and alertness to prevent falling back into old habits (Pollack & Pollack, 2015) (Appelbaum et al., 2012).
- ***Step 8: Anchoring New Approaches in the Corporate Culture:*** For long-term success, new behaviours and practices must become part of the organisational culture. This can be achieved by continuously demonstrating how the new approaches have improved performance and by ensuring that future leadership represents the new ways of thinking and acting. Standardising change prevents the organisation from slipping back into old routines (Appelbaum et al., 2012).

These steps form the basis for change management. The following practical steps can be taken by Johma to ease the transition to the developed scheduling method:

- **Step 1:** To establish a sense of urgency, the problem at hand 'not having a scheduling method for Kitchen 1' should be understood by the employees. The continuous improvement team should address the current challenge that operators do not know with what recipe (job) they should start to decrease idle times at the filling lines. The action problem 'having 18% idle time' can be taken as one of the risks of not changing. Furthermore, potentially other things like increased worker efficiency and waste reduction can also be achieved with a schedule of operations.
- **Step 2:** A team of influential and motivated employees should be assembled to lead the change process. There should be someone from the continuous improvement team who leads the team as he or she has the expertise and potential leadership skills to drive change. There should be a person with power to prevent obstruction. This can be a manager or director; the head of continuous improvement, the operations manager or even the director. Furthermore, maybe the most important members are the employees with credibility. These are respected people who can be trusted. Employees who have been around for a long time are good options as they have a lot of expertise and are respected by other employees. All these members must have the same motivation to let the project succeed.
- **Step 3:** The next step is to develop a vision and strategy. The team should determine a future state, which could be the reduction of idle time to become 15% as is in this research. The vision can be made with members from the previous step and should be well-defined.
- **Step 4:** It is necessary to communicate the change project to all involved employees. This can be done with team meetings where the project leader from step 2 explains the project's benefits. Per shift team, this can be explained together with a person who has credibility. Do this every week in short to keep all stakeholders involved.
- **Step 5:** Training and coaching is essential. The operators in Kitchen 1 should be trained to work with the new scheduling method. Give these operators ownership over this process by teaching other operators. This makes them feel responsible and in control of the process.
- **Step 6:** The project takes time, and there should be moments in between to celebrate. This can be done with short-term wins. When certain parts of the project are done correctly, for example, operators know how to interpret the schedule of operations. This can be celebrated to boost morale and tackle new challenges to come. An option is to let operators leave a bit earlier on Friday for their shift because they did a great job.
- **Step 7:** It is important that the project team from step 2 sustain the required change initiatives. They should be alert that the way of working is not falling back into old habits.
- **Step 8:** The change mindset should be anchored in company culture. This will increase the likelihood of adoption by employees and the success rate of future projects.

All in all, change management is a complex but essential process for organisational success. By following Kotter's Eight-Step Model, businesses can effectively navigate the challenges of change, engage their employees, and implement new practices into their company culture. This structured approach not only increases the likelihood of successful change but also helps organisations remain competitive and adaptive in a constantly evolving business environment.

7 Conclusions and recommendations

This chapter answers the following research question:

'What conclusions and recommendations can be drawn for Johma based on the findings from this research?'

Section 7.1 provides the main findings of this research and answers the main research question. Section 7.2 gives the recommendations. Section 7.3 discusses the limitations of this research and opportunities for future research. Lastly, Section 7.4 describes the academic and practical contributions.

7.1 Conclusions

This research explored the solution approaches for scheduling operations in Kitchen 1 to decrease idle times at the corresponding filling lines. The scheduling method used in Kitchen 1 is currently very basic and only involves the schedule of the filling lines, and the experience of the operators. By analysing the current situation, conducting a literature review, formulating a mathematical model, and developing this model in Python the following main research question is answered:

'How can Johma near-optimally schedule the required jobs and operations to prepare and mix the salad mixtures in Kitchen 1 to meet the demand of the filling lines, decrease the idle time, and increase flow in the production facility?'

The scheduling problem at hand is the FJc$|prec, S_{ij}, M_{ij}, W_m, UWP_u^{m,w}| \sum w_i Tar_i, C_{max}$. This problem consists of characteristics like precedence, sequence-dependent setups, machine and worker eligibility, and available working time intervals. The goal hereby was to minimise the total weighted tardiness of jobs (recipes) and when this is realised try to find the lowest makespan. A mathematical model was formulated based on the literature review, and accordingly, a Python model was built. For this unique scheduling problem at Johma, several solution alternatives were proposed that include the following elements;

- Constructive heuristic: This consisted of the EDD, MS, or Random dispatching rule to create a job list. The jobs and operations are scheduled according to the problem characteristics above-mentioned with either the forward or backward method.
- Improvement heuristic: By using the ILS, SA, and TS improvement heuristics the initial schedule is optimised by using the Normal and Random neighbourhood strategy.

The performance of the solution alternatives has been evaluated and the EDD dispatching rule performed the best with the forward scheduling method to create an initial solution. The parameters for the improvement heuristics have been tuned to ensure good solution outcomes with a reasonable run time. The TS improvement heuristic performs the best out of the improvement heuristics with the forward scheduling approach and the EDD dispatching rule. Furthermore, no major differences have been seen with using the Normal and Random neighbourhood strategy.

Using the best solution approach, which is TS combined with EDD and Random neighbourhood strategy several scenarios have been analysed. Zero tardiness can be achieved for instances 4, 5, 6, and 7 in the basic scenario with three preparation operators. Instances 1, 2, and 3 also achieve zero tardiness when a fourth preparation operator is added. In all cases, the makespan is substantially decreased by adding an operator, which makes sure the operational activities can be done in the available working time. Working with only one mixer impacts the overall tardiness and makespan with only a few percent, thus not being a problem with the current planning of the filling lines. Sometimes the solution would even become better showing that working with one mixer should not be a problem. Moreover, buffer time has been analysed. Having higher or lower buffer times increases or decreases the buffer time for both cases, which indicates that having it does not have a logical impact on the scheduling performance. A higher buffer time should therefore not be a problem. Model performance could not be compared directly with the real-world performance of Kitchen 1, due to not having performance data available for this department. What can be concluded is that the model can reach zero tardiness in a low run time. Furthermore, the constructive heuristic in the model performs already well compared to the improvement heuristic for some instances, and only for large instances or instances where the due dates are spread out a lot it can reduce the tardiness and makespan substantially.

It can be concluded that the constructive heuristic already provides a good solution in the first place for instances 4, 5, and 6 and that the improvement heuristics improve the solution a lot for instances 1, 2, 3, and 7.

At last, an overview is created on how to set up the model input for the model. An Excel tool is made to generate the job list for a given dispatching rule, and an Excel file with all needed input data is constructed to have all information for the model in one place. In addition, a manual has been made to show how Python can be used to run the model, and an explanation of how to interpret the output is given. For dealing with change a framework is established to ease the transition to the newly developed scheduling approach.

7.2 Recommendations

This section provides the following recommendations to Johma:

- Make the scheduling model operational with the provided settings, transition plan and manuals to decrease idle times at the filling lines.
- Use the model to check if the planning of the filling lines is feasible for Kitchen 1 and to see how many operators are needed. In general, four preparation operators are necessary, but for a smaller job list, three is sufficient. This insight can be used to plan the needed operators for shifts.
- Have one person (C-operator most likely) in Kitchen 1 who completely focuses on giving tasks. This greatly benefits the flow of going from one operation to the other and decreases the downtime of operators.
- With the schedule of operations it can be seen when ingredients are needed. Let the runner communicate beforehand to the employees in the cooling cell when certain ingredients are needed. This decreases downtime in Kitchen 1.
- Further investigate when the salad mixture should be finished in Kitchen 1. The Excel tool currently takes the due date and the speed of the filling lines and uses this to calculate the finish times at Kitchen 1. Probably some extra buffer time in between is beneficial.
- Update recipes with the correct ingredient numbers, actions, amounts etc. Currently, it is not normalised and every recipe overview is different.
- Measure more processing times to make the model even more accurate. Currently, all operations have been measured, but more measurements make the data more reliable and up-to-date.
- Store the actual finish times of the salad mixtures in Kitchen 1. This would fill the data gap and improve the performance of the model. Additionally, it would allow us to better determine if the kitchen meets the demand of the filling lines. With this data, more data-driven decisions for the improvement of Kitchen 1 can be made.

7.3 Limitations and future research

This section describes the limitations of this research and provides opportunities for future research. Only Kitchen 1 is considered and not the surrounding departments that influence Kitchen 1. Kitchen 1 greatly depends on the ingredients provided by the cooling cell, dry spices, cold kitchen, warm kitchen, and pasta kitchen departments. We assumed that all these ingredients are available, but in reality, this is not always the case. Future research could further broaden the scheduling of operations at those departments to align with the schedule in Kitchen 1.

The data in our model is all deterministic. In the real world, data can change, especially the planning at the filling lines. The model can be re-run with the short run times, however, it would be better if there is an automatic rescheduling approach. A possible future research topic could be dynamic rescheduling instead of having to re-run the model when changes occur.

The model does not implement machine breakdowns as there was no data available for this. Research can be done on the impact of machine failures on the production efficiency in Kitchen 1. Although breakdowns do not happen that often, it is interesting to measure the performance and see how this impacts the overall performance in Kitchen 1.

The number of carts and funnels is regarded as infinite in our model. An interesting topic for future research would be to model the total amount of carts and funnels needed in the factory and per department. Many employees on the shop floor argued that there were not enough carts and funnels. It was also observed that sometimes the work is shut down because there is no funnel available at the mixers.

The current model cannot directly be used as an operational tool. Future research could be done to see how this scheduling model approach can be fully implemented in daily operational activities. To extend this research, the planning of the filling lines can be researched as we are unsure if this is an optimal planning. In this research all is done to adhere to the planning of the filling lines, however, if this planning is not efficient then it is not good to schedule based on that planning.

7.4 Academic and practical contributions

This section describes the academic and practical contribution of this research. The FJSP at Johma is a unique problem that has not been researched before as a whole. Some elements are researched in literature, but overall not in a real-world practical setting. Section 3.2.4 stated that most elements combined have been researched before, however, the forbidden interval part has not been researched before with the characteristics from our problem. In addition, we developed a full mathematical model that represents our problem and modelled this in Python with three dispatching rules, two scheduling methods, and three improvement heuristics. Therefore, we were able to provide a broad theoretical analysis of the performances of the solution alternatives. Furthermore, the literature states that backward scheduling is the most beneficial with problems that consists of due dates. However, in our research, we have shown that forward scheduling is the better-performing approach for our unique problem. This shows that in practice things can be very different.

The practical contribution lies in the extensive insights gained for Kitchen 1. In the first place, Johma did not know much about the activities in Kitchen 1. A complete context analysis together with a data measurement study made a basis for the operations happening in Kitchen 1. Furthermore, with the developed scheduling model Johma can work towards an operational scheduling approach for Kitchen 1, which Johma did not have. Already, insights into operator capacity, schedule feasibility, and separate finish times for recipes can be obtained. All in all, Johma can make better data-driven decisions with the developed scheduling model.

References

Allahverdi, A., Gupta, J. N., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, *27(2)*, 219-239. doi: https://doi.org/10.1016/S0305-0483(98)00042-5

Appelbaum, S. H., Habashy, S., Malo, J., & Shafiq, H. (2012). Back to the future: revisiting kotter's 1996 change model. *Journal of Management Development*, *31*, 764-782. doi: 10.1108/02621711211253231

Battistutta, M., Schaerf, A., & Urli, T. (2015). Feature-based tuning of single-stage simulated annealing for examination timetabling. *Annels of Operations Research*, *252(2)*, 239-254. doi: https://doi.org/10.1007/s10479-015-2061-8

Baykasoglu, A., & Ozbakir, L. (2010). Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system. *Journal of Production Economics*, *124(2)*, 369-381. doi: https://doi.org/10.1016/j.ijpe.2009.11.032

Benton, W. (2011). Push and pull production systems. *Wiley encyclopedia of operations research and management science*. doi: https://doi.org/10.1002/9780470400531.eorms0690

Bissoli, D., Zufferey, N., & Amaral, A. (2019). Lexicographic optimization-based clustering search metaheuristic for the multiobjective flexible job shop scheduling problem. *International Transactions in Operational Research*. doi: 10.1111/itor.12745

Campbell, R. (2008). Change management in health care. *The Health Care Manager*, *27*, 23-39. doi: 10.1097/01.HCM.0000285028.79762.a1

Chaudhry, I., & Khan, A. (2013). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, *23(3)*, 551-591. doi: https://doi.org/10.1111/itor.12199

Cinar, D., Topcu, Y., & Oliveira, J. (2015). Taxonomy for the flexible job shop scheduling problem. springer proceedings. *Journal of Mathematics Statistics*, 17-37. doi: https://doi.org/10.1007/978-3-319-18567-5_2

Dalfard, V., & Mohammadi, G. (2012). Two meta-heuristic algorithms for solving multi-objective flexible job-shop scheduling with parallel machine and maintenance constraints. *Computers and Mathematics with Applications*, *64(6)*, 2111-2117. doi: https://doi.org/10.1016/j.camwa.2012.04.007

Deng, G., Zhang, Z., Jiang, T., & Zhang, S. (2019). Total flow time minimization in no-wait job shop using a hybrid discrete group search optimizer. *Applied Soft Computing Journal*, *81*, 105480. doi: https://doi.org/10.1016/j.asoc.2019.05.007

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2018). Reformulations and an exact algorithm for unrelated parallel machine scheduling problem with setup times. *Computers Operations Research*. doi: https://doi.org/10.1016/j.cor.2018.07.007

Fattahi, P., Jolai, F., & Arkat, J. (2009). Flexible job shop scheduling with overlapping in operations. *Applied Mathematical Modelling*, *33(7)*, 3076-3087. doi: https://doi.org/10.1016/j.apm.2008.10.029

Ganesan, V., & Sivakumar, A. (2006). Scheduling in static jobshops for minimizing mean flowtime subject to minimum total deviation of job completion times. *International Journal of Production Economics*, *103(2)*, 633-647. doi: https://doi.org/10.1016/j.ijpe.2005.12.004

Gao, J., Gen, M., & Sun, L. (2006). Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing*, *17(4)*, 493-507. doi: https://doi.org/10.1007/s10845-005-0021-x

Gao, K., Suganthan, P., Tasgetiren, M., Pan, Q., & Sun, Q. (2015). Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion. *Journal of Computers Industrial Engineering*, *90*, 107-117. doi: https://doi.org/10.1016/j.cie.2015.09.005

Graham, R., Lawler, E., Lenstra, J., & Kan, A. R. (1979). Optimization and approximation in deter-

ministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, *5*, 287-326. doi: https://doi.org/10.1016/S0167-5060(08)70356-X

Heerkens, H., & van Winden, A. (2017). *Solving managerial problems systematically* (1st ed.). Noordhoff Uitgevers.

Johma Salades. (2023). *Over ons.* Retrieved 11/12/2023, from `https://www.johma.nl/over-ons`

Kalinowski, K., Grabowik, C., Cwikla, G., Paprocka, I., & Balon, B. (2018). Production orders planning using additional backward pass scheduling approach. *IOP Conference Series: Materials Science and Engineering*, *400*. doi: https://doi.org/10.1088/1757-899X/400/6/062015

Kim, J.-G., Song, S., & Jeong, B. (2020). Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times. *International Journal of Production Research*, *58*, 1628 - 1643. doi: 10.1080/00207543.2019.1672900

Kress, D., & Müller, D. (2019). Mathematical models for a flexible job shop scheduling problem with machine operator constraints. *IFAC-PapersOnLine*, *52(13)*, 94-99. doi: https://doi.org/10.1016/j.ifacol.2019.11.144

Lee, T., & Loong, Y. (2019). A review of scheduling problem and resolution methods in flexible flow shop. *International Journal of Industrial Engineering Computations*. doi: 10.5267/J.IJIEC.2018.4.001

Lei, D., & Guo, X. (2011). Variable neighbourhood search for minimising tardiness objective on flow shop with batch processing machines. *Journal of Production Research*, *49(2)*, 513-529. doi: https://doi.org/10.1080/00207540903536130

Lei, D., & Guo, X. (2016). Hybrid flow shop scheduling with not-all-machines options via local search with controlled deterioration. *Comput. Oper. Res.*, *65*, 76-82. doi: 10.1016/j.cor.2015.05.010

Lei, D., & Tan, X. (2016). Local search with controlled deterioration for multi-objective scheduling in dual-resource constrained flexible job shop. *2016 Chinese Control and Decision*. doi: https://doi.org/10.1109/CCDC.2016.7531874

Li, J., Pan, Q., Suganthan, P., & Chua, T. (2010). A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing*, *52(5-8)*, 683-697. doi: https://doi.org/10.1007/s00170-010-2743-y

Li, X., Xie, J., Peng, K., Li, H., & Gao, L. (2019). Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing*. doi: https://doi.org/10.1049/IET-CIM.2018.0009

Mastrolilli, M., & Gambardella, L. (2000). Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, *3(1)*, 3-20. doi: https://doi.org/10.1002/(SICI)1099-1425(200001/02)3:1¡3::AID-JOS32¿3.0.CO;2-Y

McKay, K. N., & Wiers, V. C. (2003). Planning, scheduling and dispatching tasks in production control. *Cognition, Technology Work*, *5*, 82-93. doi: https://doi.org/10.1007/s10111-002-0117-4

Mousakhani, M. (2013). Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. *International Journal of Production Research*, *51(12)*, 3476-3487. doi: https://doi.org/10.1080/00207543.2012.746480

Naderi, B., & Zandieh, M. (2014). Modeling and scheduling no-wait open shop problems. *International Journal of Production Economics*, *158*, 256-266. doi: 10.1016/J.IJPE.2014.06.011

Olhager, J. (2010). The role of the customer order decoupling point in production and supply chain management. *Computers in Industry*, *61*, 863-868. doi: https://doi.org/10.1016/j.compind.2010.07.011

Paksi, A. B. N., & Ma'ruf, A. (2016). Flexible job-shop scheduling with dual-resource constraints to minimize tardiness using genetic algorithm. *IOP Conference Series: Materials Science and Engineering*, *114*, 012060. doi: https://doi.org/10.1088/1757-899X/114/1/012060

Pandey, H. M., & Gajendran, A. (2016). Function optimization using robust simulated annealing. *Information Systems Desing and Intelligent Applications*, 347-355. doi: https://doi.org/10.1007/978-81-322-2757-1$_3$5

Perez-Gonzalez, P., & Framinan, J. (2014). A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research*, *235(1)*, 1-16. doi: https://doi.org/10.1016/j.ejor.2013.09.017

Perez-Gonzalez, P., & Framiñan, J. (2018). Single machine scheduling with periodic machine availability. *Comput. Ind. Eng.*, *123*, 180-188. doi: 10.1016/j.cie.2018.06.025

Pinedo, M. L. (2016). Scheduling: Theory, algorithms, and systems. *Textbook*. doi: https://link.springer.com/book/10.1007/978-3-319-26580-3

Pollack, J., & Pollack, R. (2015). Using kotter's eight stage process to manage an organisational change program: Presentation and practice. *Systemic Practice and Action Research*, *28*, 51-66. doi: 10.1007/S11213-014-9317-0

Saidi-Mehrabad, M., & Fattahi, P. (2006). Flexible job shop scheduling with tabu search algorithms. *Journal of Advanced Manufacturing Technology*, *32(5-6)*, 563-570. doi: https://doi.org/10.1007/s00170-005-0375-4

Shahgholi, M. Z., Katebi, Y., & Daniavi, A. (2018). A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times. *International Journal of Production Research*, 1-16. doi: https://doi.org/10.1080/00207543.2018.1524165

Shen, L., Dauzere-Peres, S., & Neufeld, J. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, *265(2)*, 503-516. doi: https://doi.org/10.1016/j.ejor.2017.08.021

Sobeyko, O., & Mönch, L. (2016). Heuristic approaches for scheduling jobs in large-scale flexible job shops. *Computers Operations Research*, *68*, 97-109. doi: https://doi.org/10.1016/j.cor.2015.11.004

Yazdani, M., Amiri, M., & Zandieh, M. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, *37(1)*, 678-687. doi: https://doi.org/10.1016/j.eswa.2009.06.007

Zhang, Q., Manier, H., & Manier, M. A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers Operations Research*, *39(7)*, 1713-1723. doi: https://doi.org/10.1016/j.cor.2011.10.007

Ziaee, M. (2013). A heuristic algorithm for solving flexible job shop scheduling problem. *Journal of Advanced Manufacturing Technology*, *71(1-4)*, 519-528. doi: https://doi.org/10.1007/s00170-013-5510-z

A Constructive heuristic flow chart

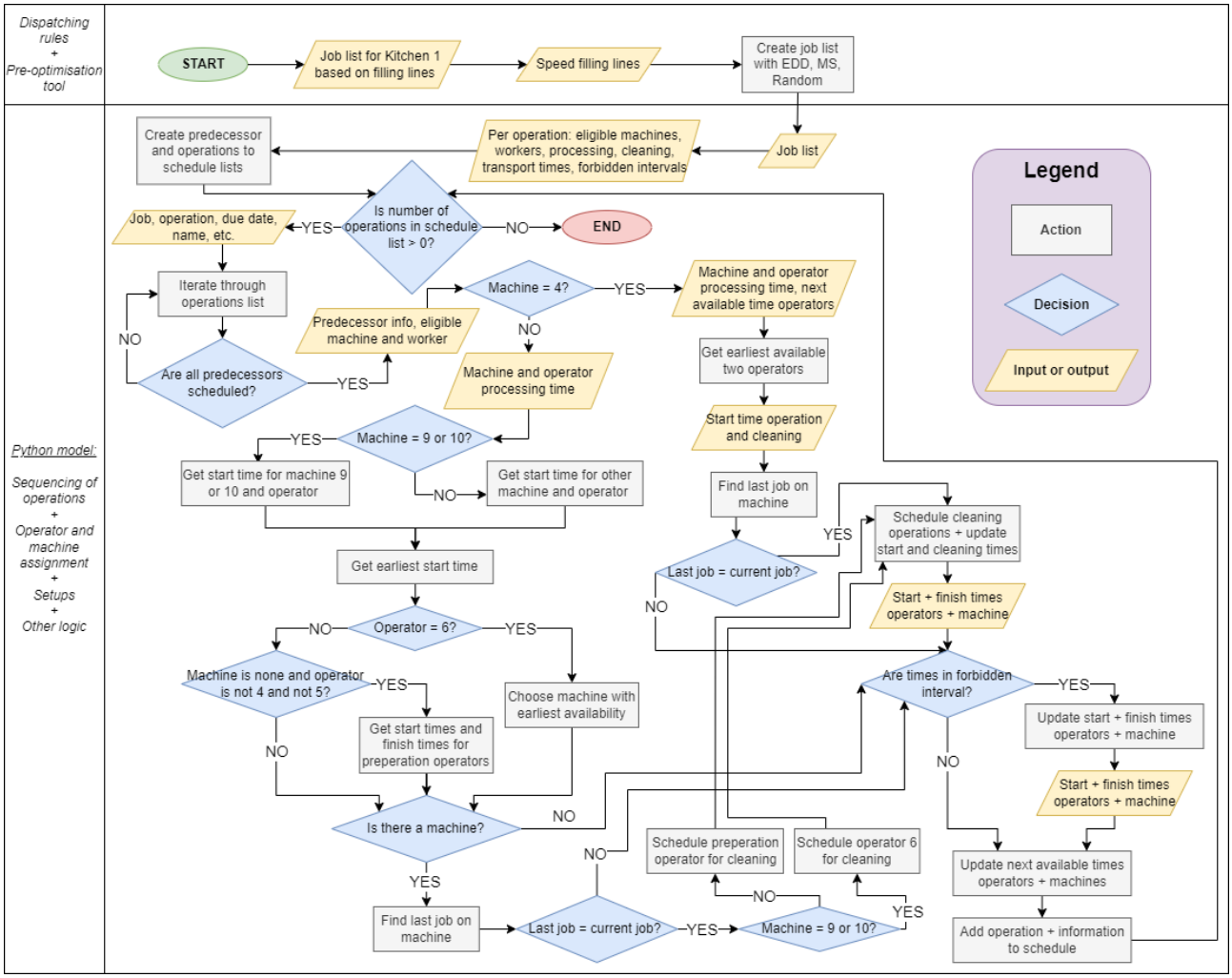


Figure A.1: Constructive heuristic flow chart.

B Parameter tuning

To have good performing improvement heuristics, the parameters should be determined correctly. For static parameters we tune them by experimenting with three different values and then looking at which performs the best together with the largest problem instance (instance 1), the best performing dispatching rule (EDD), and the backward and forward models. Section B.1 explains this for the ILS parameter, Section B.2 for the SA parameters, and Section B.3 for the TS parameters.

B.1 Iterative Local Search parameter tuning

For ILS we only have to tune the iteration parameter, which is static. We define 10, 100, 200, 500, and 1000 iterations to experiment with and choose to have 5 replications per experiment because of randomness in the improvement heuristic. Figures B.1 and B.2 show the average results over 5 runs per iteration size. The objective to optimise is tardiness, and we can see that with more iterations, the objective improves in general. However, this comes at a cost of higher run time, which is not ideal. Johma wants to have a feasible good outcome in a reasonable running time. Because of this, we choose to have 500 ILS iterations for the backward model, which leads to a comparable outcome as with 1000 iterations but requires almost half the run time. The forward model gives better outcomes with 500 iterations compared to 1000 iterations. Therefore because the running time is also reasonable we also choose 500 iterations for the forward model.

Instance	ILS Iterations	Dispatching rule	Tardiness	Run time
1	10	EDD	62,372.13	46.58
	100	EDD	43,983.57	389.82
	200	EDD	40,508.71	750.78
	500	EDD	35,546.62	1,966.72
	1000	EDD	32,041.21	3,731.94

Figure B.1: Parameter tuning results backward model with ILS.

Instance	ILS Iterations	Dispatching rule	Tardiness	Run time
1	10	EDD	41,692.24	43.26
	100	EDD	22,877.32	381.41
	200	EDD	21,892.64	758.11
	500	EDD	14,835.48	1,886.66
	1000	EDD	18,117.78	3,768.11

Figure B.2: Parameter tuning results forward model with ILS.

B.2 Simulated Annealing parameter tuning

For SA, we have to tune four parameters, namely starting temperature, stopping temperature, cooling rate, and Markov chain length. According to Pandey and Gajendran (2016) and Battistutta et al. (2015), the starting temperature can be tuned based on the acceptance ratio. The acceptance ratio determines how many worse neighbours are accepted at a certain stage (in this case temperature level) of the heuristic. When the ratio is high, many neighbourhoods are explored, increasing the chances of getting out of local optima. At the start stages of the heuristic, we want to explore a broader solution space and get more into intensification. The settings for running the starting temp experiments are a starting temperature of 100, a cooling rate of 0.85, and a Markov chain length of 100. We save the acceptance ratio at each temperature stage to create the graphs in Figures B.3 and B.4. We determine the value of the temperature where the heuristic quickly drops in acceptance ratio as this is the point where a lot less worse neighbours are accepted. This is at a temperature of 52, with an acceptance ratio of 0.8 for the backward model. Thus for tuning the rest of the parameters we choose a starting temperature of 52. For the forward model, this is at a temperature of 61.

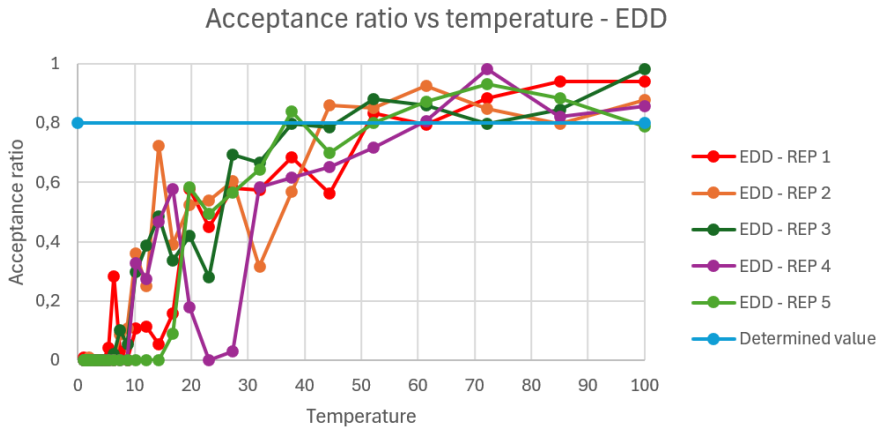


Figure B.3: Starting temperature tuning results backward model with SA and EDD.

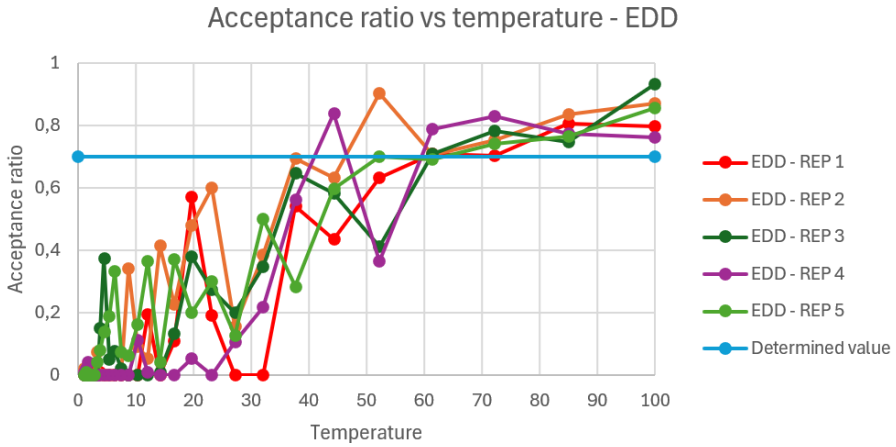


Figure B.4: Starting temperature tuning results forward model with SA and EDD.

With the starting temperatures of 52 and 61 we will experiment with the static parameters, which are the stopping temperature, cooling rate, and the Markov chain length. The three values per parameter are 2.5, 5, 7.5 for stopping temperature, 0.85, 0.90, 0.95 for cooling rate, and 10, 15, 20 for Markov chain length. Figures B.5 and B.6 show the average results of the experiments with 5 replications per experiment to handle randomness.

Stopping temp	Markov chain length	Cooling rate alpha	Tardiness	Run time
2.5	10	0.85	40,698.21	402.25
5	10	0.85	40,325.96	287.06
7.5	10	0.85	44,099.79	240.84
2.5	15	0.85	34,943.26	553.33
5	15	0.85	39,528.23	429.04
7.5	15	0.85	48,286.12	344.67
2.5	20	0.85	46,598.29	740.22
5	20	0.85	37,351.66	571.41
7.5	20	0.85	34,160.99	457.82
2.5	10	0.9	45,571.13	571.27
5	10	0.9	38,274.76	438.06
7.5	10	0.9	37,654.06	363.32
2.5	15	0.9	35,604.49	848.66
5	15	0.9	38,024.00	655.28
7.5	15	0.9	39,125.00	542.55
2.5	20	0.9	36,245.09	1,133.37
5	20	0.9	41,386.52	871.91
7.5	20	0.9	43,255.50	721.36
2.5	10	0.95	41,813.60	1,197.75
5	10	0.95	37,400.33	871.80
7.5	10	0.95	38,129.70	719.47
2.5	15	0.95	33,447.84	1,789.04
5	15	0.95	35,165.58	1,307.23
7.5	15	0.95	37,563.86	1,079.77
2.5	20	0.95	35,077.17	2,299.90
5	20	0.95	29,672.02	1,752.88
7.5	20	0.95	37,824.35	1,439.26

Figure B.5: Parameter tuning results backward model with SA and EDD.

Stopping temp	Markov chain length	Cooling rate alpha	Tardiness	Run time
2.5	10	0.85	25,230.56	384.19
5	10	0.85	25,824.72	296.43
7.5	10	0.85	26,088.2	242.22
2.5	15	0.85	26,096.69	551.88
5	15	0.85	22,497.42	442.15
7.5	15	0.85	21,237.96	360.86
2.5	20	0.85	19,879.76	732.39
5	20	0.85	22,948.01	587.68
7.5	20	0.85	21,445.63	478.65
2.5	10	0.9	19,445.96	570.26
5	10	0.9	22,347.06	442.57
7.5	10	0.9	22,303.01	373.09
2.5	15	0.9	17,493.64	851.36
5	15	0.9	17,741.52	660.41
7.5	15	0.9	18,562.14	552.05
2.5	20	0.9	18,501.75	1,137.81
5	20	0.9	20,587.5	878.55
7.5	20	0.9	21,113.66	734.62
2.5	10	0.95	18,113.24	1,152.46
5	10	0.95	16,875.12	896.85
7.5	10	0.95	17,203.07	752.38
2.5	15	0.95	19,943.85	1726.7
5	15	0.95	18,224.65	1,346.83
7.5	15	0.95	18,815.87	1,149.29
2.5	20	0.95	16,980.0	2,298.61
5	20	0.95	18,347.69	1,790.99
7.5	20	0.95	20,727.51	1,502.11

Figure B.6: Parameter tuning results forward model with SA and EDD.

From these experiments, we can see that a higher cooling rate of 0.95 for the backward model results in lower tardiness in general. In addition, a Markov chain length of 20 results in lower tardiness and with a stopping temperature of 5 the lowest tardiness is achieved. To conclude we choose 0.95 for the cooling rate, 20 for the Markov chain length, and 5 for the stopping temperature for the backward model. For the forward model also a cooling rate of 0.95 results in lower tardiness in general. A Markov chain length of 10 is in this case already in general better, and with a stopping temperature of 5 the lowest tardiness is achieved. Therefore, we choose a cooling rate of 0.95, a Markov chain length of 10, and a stopping temperature of 5 for the forward model.

B.3 Tabu Search parameter tuning

For TS, we have to tune three parameters, namely maximum iterations, tabu list length, and neighbours evaluated. The three values per parameter are 25, 50, 75 for maximum iterations, 5, 10, 15, for tabu list length, and 5, 10, 15 for neighbours evaluated. Figures B.7 and B.8 show the average results of the experiments with 5 replications per experiment to handle randomness.

Maximum iterations	Tabu list length	Neighbours evaluated	Tardiness	Run time
25	5	5	38,540.29	291.2
25	5	10	36,668.98	534.17
25	5	15	30,379.6	755.92
25	10	5	35,187.09	288.25
25	10	10	30,387.07	542.26
25	10	15	36,293.9	755.87
25	15	5	32,954.38	290.03
25	15	10	28,722.67	534.8
25	15	15	26,386.96	784.53
50	5	5	34,997.77	580.83
50	5	10	27,910.82	1,037.95
50	5	15	30,940.91	1,530.28
50	10	5	33,124.28	580.1
50	10	10	27,340.9	1,032.94
50	10	15	25,655.87	1,489.37
50	15	5	32,011.53	558.45
50	15	10	28,371.68	1,022.92
50	15	15	21,578.45	1,475.49
75	5	5	25,952.51	850.24
75	5	10	25,791.0	1,545.28
75	5	15	20,408.38	2,250.34
75	10	5	27,136.68	827.44
75	10	10	24,130.84	1,510.59
75	10	15	23,789.06	2,199.36
75	15	5	29,776.76	827.56
75	15	10	22,452.3	1,513.28
75	15	15	22,193.95	2,201.35

Figure B.7: Parameter tuning results backward model with TS and EDD.

Maximum iterations	Tabu list length	Neighbours evaluated	Tardiness	Run time
25	5	5	22,267.76	289.84
25	5	10	17,921.36	499.6
25	5	15	19,472.07	808.68
25	10	5	20,361.6	281.01
25	10	10	15,562.53	513.82
25	10	15	12,990.58	723.07
25	15	5	25,253.86	275.6
25	15	10	21,449.83	498.99
25	15	15	17,945.85	740.2
50	5	5	17,097.22	557.66
50	5	10	10,138.33	1,018.55
50	5	15	14,714.12	1,475.62
50	10	5	16,700.55	558.23
50	10	10	16,934.97	1,020.75
50	10	15	11,893.31	1,477.87
50	15	5	20,461.76	558.47
50	15	10	14,135.11	1,057.47
50	15	15	12,830.19	1,474.16
75	5	5	15,452.1	832.43
75	5	10	10,013.49	1,520.96
75	5	15	9,780.11	2,206.86
75	10	5	18,753.08	830.83
75	10	10	12,936.87	1,519.55
75	10	15	12,397.72	2,207.09
75	15	5	13,942.33	849.34
75	15	10	11,928.28	1,527.43
75	15	15	9,539.51	2,290.11

Figure B.8: Parameter tuning results forward model with TS and EDD.

From these experiments, we can see that a higher number of evaluated neighbours and with the number of maximum iterations at 75 for the backward model the tardiness is in general lower than the other settings. In addition, with a Tabu list length of 5 the lowest tardiness is achieved. To conclude we choose 15 for the evaluated neighbours, 75 for the maximum number of iterations, and 5 for the Tabu list length for the backward model. For the forward model also 15 evaluated neighbours results in lower tardiness in general. A Tabu list length of 15 is in this case in general better, and with the maximum number of iterations at 75 the lowest tardiness is achieved. Therefore, we choose 15 for the evaluated neighbours, 75 for the maximum number of iterations, and 15 for the Tabu list length for the forward model.

C Usage of model

C.1 Creating job lists

For generating the right job list as input for the model we use the Excel file "Job list generator for dispatching rules". In sheet "230" the planning of Kitchen 1 from JAS is inserted. In sheet "Kg per min vullijnen" the information for the speed per filling line is located. In the sheet "Due Date Calculation" the information from sheet "230" is copied to columns G:L. In the sheet "Due Date Tool" we can click on the first button "Genereer vullijnnummers + receptnamen" to first create the recipe names and job numbers in columns G and H. Then we can click on the button "Bereken Due Date trechter" to calculate the due dates for all individual jobs. After this we click the third button "EDD joblist" to create the EDD job list in columns C:E. When we have done this we can create three different job lists depending on the dispatching rule:

- EDD: Go to sheet "EDD job list" and click on "Generate EDD job list". The EDD job list is in columns A:C.
- MS: Go to sheet "MS job list" and first click "Calculate worktime in minutes" to get the processing times in minutes per recipe in column N. Furthermore, click on "Generate MS job list" to get the EDD job list in columns A, B and D.
- RANDOM: Go to sheet "RANDOM job list" and click on "Generate RANDOM job list". The RANDOM job list is in columns A:C.

When the needed job list is made this can be copied in the file "INPUT MODEL BS-ST-SJS" in the sheet "Job list created" in the columns A:C. Following from this, in the sheet "Input job list" we have to put unique numbers from 1 onwards in column D for each job. Now the input is ready for the model in Python.

C.2 Editing/ adding measured processing times

In the file "INPUT MODEL BS-ST-SJS" in the sheet "Input data process times 2" all information is located about processing times, operator and machine assignment etc. The following information is needed to understand columns C and D, Machine and Operator respectively; **Machine** 1 = Can opener, 2 = Urschel, 3 = Holac, 4 = Pilling machine, 5 = Egg machine, 6 = Vibrating sieve, 7 = Large cutter, 8 = Small cutter, 9 and 10 = the two mixers, **Operator** 1 = Preparation operator 1, 2 = Preparation operator 2, 3 = Preparation operator 3, 4 = Runner for transport, 5 = Operator for adding wet spices, and 6 = Mixer operator. In columns E and F the duration in minutes is located.

For each ingredient or action we can link the machine and operator that can do this. For an ingredient or action with more than one option, for example, can be done by any of the preparation operators or by any of the two mixers, we make two distinct rows so that the model knows that it can select one of these options. Row 2 and 3 show that the "mixing time" action can be done by both machines 9 or 10 with operator 6. Rows 7, 8, and 9 show that the ingredient "CHICKEN ROASTED 10X10X10 BRA" can be done by any of the preparation operators thus needing three separate rows. This is done for all ingredients and actions for all recipes.

The sheet "Input successors" shows how the recipe should be done in sequence. Some operations within a recipe can be done simultaneously and some should be done in sequence. For R107 all operations should be done in sequence thus job 1,2 depends on 1,1 and so on (see columns U and V). For R110 some operations can be done simultaneously. Operation 6 and 4+5 can be done simultaneously thus operation 6 does not depend on 5 but on 3 (see columns U and V). This is done for all recipes and their operations.

The sheet "Input data other successors" shows the cleaning time on the machine per operation in column J. The machine is in column O and with this we can link the cleaning times from columns S:U to the different operations. This is done for all recipes and their operations.

Based on this logic other recipes and operations can be added and when more processing times are measured these can be altered.

C.3 Using model in Python

The programming language Python is used for running the model. We use Python Spyder as our integrated development environment. This can be downloaded via https://www.spyder-ide.org/ and is a completely free open-source platform. At the download tap the software can be downloaded for Windows. The written manual made for Johma explains the rest in more detail.