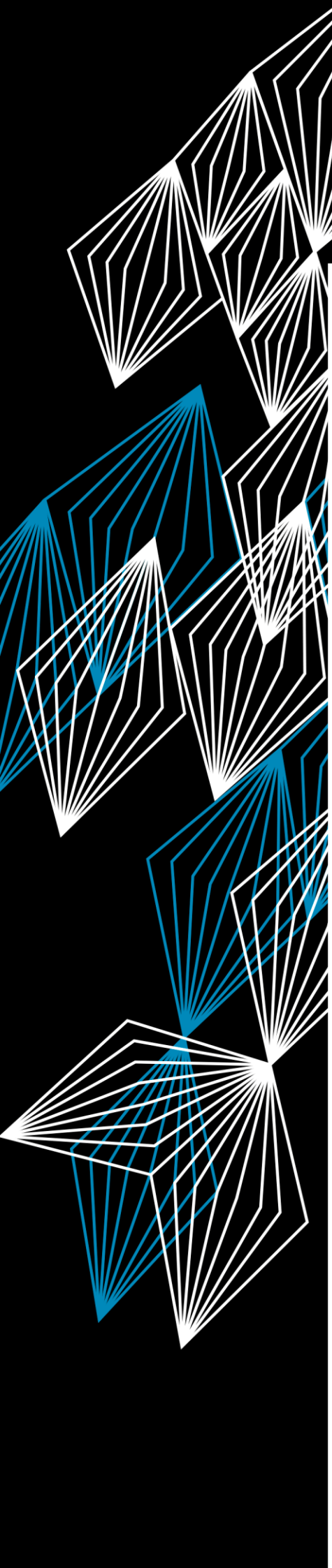


MASTER THESIS

“Dynamic product routing in a hybrid flow shop”

A case study at Machine Company X

E. H. Keppels

November, 2019

UNIVERSITY OF TWENTE.

Dynamic product routing in a hybrid flow shop

A case study at Machine Company X

E.H. Keppels

Enschede, The Netherlands, November 2019

Master thesis

Industrial Engineering and Management

Faculty of Behavioral Management and Social Sciences

Department of Industrial Engineering and Business Information Systems

University of Twente

Dr. Ir. J.M.J. Schutten
Dr. Ir. M.R.K. Mes

Machine Company X

Lead Engineer

UNIVERSITY OF TWENTE.

Preface

This thesis is the final step in completing my master Industrial Engineering and Management at the University of Twente and concludes my time as a student. During my study I have had the opportunity to gain knowledge in the research areas I found interesting, to make a group of long-lasting friends and to study abroad, which allowed me to learn more about myself. I definitely look back at my time as a student with joy, but also look forward to the time that will come; a time where I can build a career and further develop myself.

The research for this thesis is based on a case study at Machine Company X. This company also provided a workspace where I could work full time on my thesis. I have had a great time working at the company, so I would like to thank the people that were responsible for this. It was great to be part of the team. From a housewarming to bowling to a baby shower, since day one you involved me in all the activities they had as a team, so thank you to all people involved. On top of that I want to thank my supervisor within the company, you always provided great insights when I was stuck and the communication has always been very smooth.

I also want to thank Marco Schutten and Martijn Mes, my supervisors of the UT. Every meeting provided lots of useful feedback, which also meant that it rarely occurred that a meeting was finished within the time that was planned for it.

Finally, I want to thank my family, my girlfriend and friends for showing interest in my project, providing moral support and perhaps most important; providing a source of distraction and relaxation besides the thesis.

I hope you enjoy reading this thesis!

Enschede, November 2019,

Erik Keppels

Management Summary

The research of this thesis takes place at a company we call Machine Company X (MCX), a manufacturer of steel processing machines, either as stand-alone machines or complete production lines. Their product range contains drills, saws, shot blasters, plasma cutters and painting machines that can be used to process steel beams and steel plates. The machines for processing steel beams can be connected with each other with transport systems to create an automated processing line: a beamline. Beams are automatically transported on the beamline, by roller conveyors (RCs) and cross transports (CTs) according to the routing rules in MCX's software. These routing rules are the focus of this research.

The goal of this research is to evaluate the current routing rules used in the product routing of MCX, investigate possible improvements and design a solution that can be implemented together with the software department.

Figure 0.1 shows an example of a beamline layout. After machine 2 (M2), there is a split, products can go to either M3 or M4. The direction a job goes is now based on "static" rules, for example based on maximum allowed length of a beam or the number of beams in the buffer before both machines. These static rules need to be determined and configured for every individual split and are designed to create a steady flow through the system, where the goal is often to create an equal load balance over the machines. Disadvantages of these static rules are that they only consider the buffers and machines that directly follow the split and do not consider processing times, buffer occupation further in the beamline or upcoming beams. Another disadvantage is that these rules need to be configured and programmed for every beamline separately, which is a large time investment.

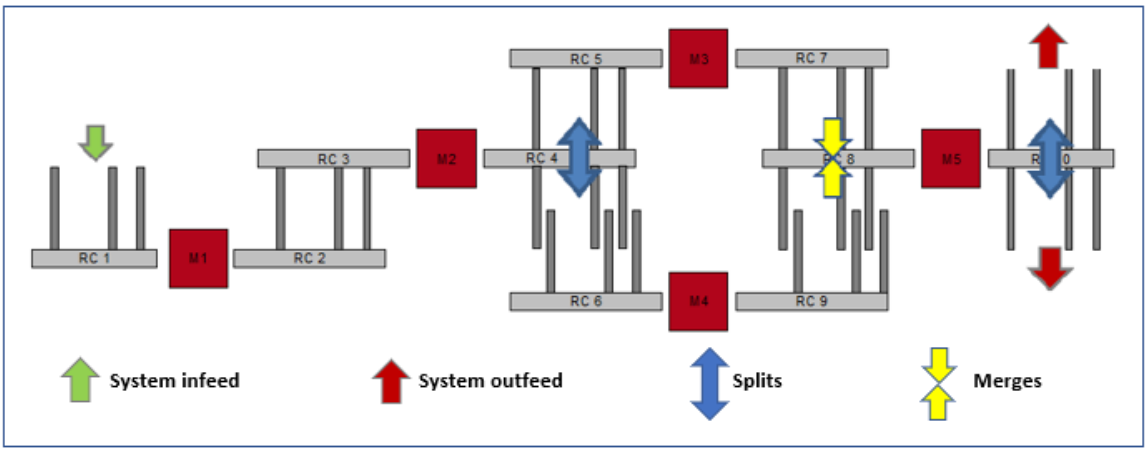


Figure 0.1: Beamline example

To tackle the mentioned problems and work towards our goal, we developed a node-based dynamic product routing that considers all feasible routes through the beamline and picks the best route, based on current occupation of the beamline. This product routing needs to work regardless of the configuration of the beamline. Literature shows that the beamline can be classified as a hybrid flow shop (HFS). Previous studies that focus on product routing in a HFS mainly focus on picking the best machines for a job and take the transport between machines for granted or simplify it to a large extend. We want a product routing that includes both the machine assignment of a job, as well as the physical path a job need traverse through the system.

We start our solution design by looking at a beamline as a network of individual node. These nodes are RC nodes and CT nodes for transport and buffering, machine nodes for performing operation, and in- and outfeed nodes that function as

start and endpoints in the beamline. To model a beamline, we connect the nodes like the different parts of a beamline are connected in practice. Figure 0.2 shows how the beamline of Figure 0.1 is modelled as network of nodes.

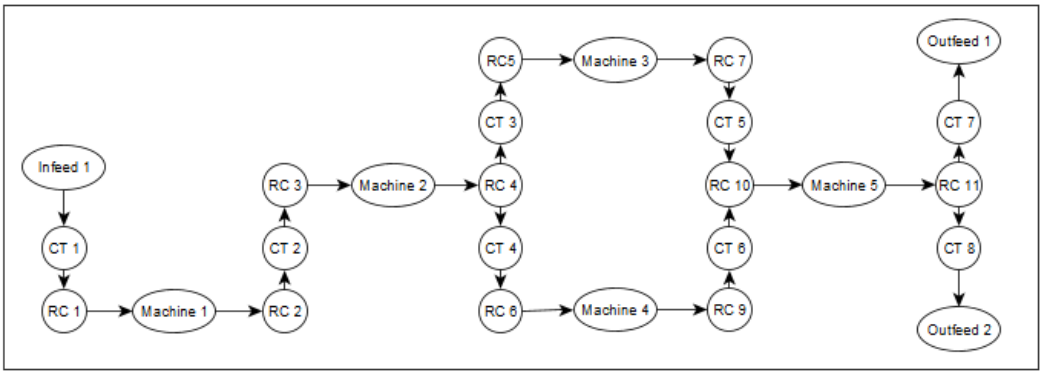


Figure 0.2: Network of nodes

Next is our definition of a route. A route consist of a physical path, which indicates all nodes on the route from an infeed node to an outfeed location, and a machine assignment, which indicates what machine on the physical path performs what operation. Jobs that arrive at the beamline have a certain infeed location, outfeed location and set of required operations. Based on these data, all viable routes through the beamline are determined per job. A route is viable for a job when it starts and ends at the right nodes and all required operations can be performed on the route. When we know the set of viable routes of a job, we score all routes based on three attributes:

- The expected throughput time (α_1), based on the expected transportation times, waiting times and processing times along the route.
- The expected relative costs (α_2), based on the required operations of a job and the machine assignment of the route.
- How well the route spreads the work over the machines (α_3), so how well every machine is being utilized. To calculate α_3, we determine the expected total processing time per machine on the beamline, when the route would be chosen. Then, we determine the variance over these processing times.

When the attribute values of all viable routes of a job are determined, we normalized them on a scale from 0 to 1, where the best value is scored as 0 and the worst as 1 (Table 0.1 gives an example). We determine the final score of a route by multiplying the attribute scores with weights. We use weights, because it allows the user of the routing rules to focus on the attribute they find important, e.g. processing costs versus machine utilization. The weights are determined with the AHP method. Multiplying every attribute score with its weight and summing the results gives the total score per route. The route with the best score is assigned to the job.

Table 0.1: Example of determining the best route

Route	Throughput time α_1	Score$_1$	Costs α_2	Score$_2$	Work Variation α_3	Score$_3$
1	1:50:04.19	0.00	4	0.00	2,066,359	0.95
2	1:53:29.48	0.29	4	0.00	2,103,271	1.00
3	1:58:18.68	0.69	6	0.33	1,698,857	0.41
4	2:02:01.47	1.00	6	0.33	1,730,021	0.46
5	2:01:43.97	0.98	10	1.00	1,415,789	0.00
6	1:57:09.97	0.59	4	0.00	2,099,486	0.99

The routing rules are strongly affected by the amount of jobs in the system and the amount of processing time of these jobs. We use dispatching rule to regulate the sequence in which nestings enter the beamline and test what effect the sequence has on the routing rules. The dispatching rules sort the nesting before the beamline on the basis of their processing times. The dispatching rules we test are:

0. Random (RAND) selects a random job. The random job is chosen with a uniform distribution.
1. Least-Work-Remaining-First (LWRF) selects the job with the least total work remaining. The total work is equal to the processing time that is required to finish all operations of the job.
2. Most-Work-Remaining-First (MWRF) selects the job with the most total work remaining.
3. Least-Work-for-Bottleneck-First (LWBF) selects the job that adds the least processing time to the current bottleneck machine.
4. Most-Work-for-Idle-Machine-First (MWIMF) selects the job that adds the most processing time to the machine that currently has the least processing time in the system.
5. Best-Work-Spread-First (BWSF) selects the job that divides the work the best over all machines in the system. The goal of this rule is to provide every machine with the same amount of processing time.

To test our routing and dispatching rules we build a simulation model in the Plant Simulation software of Siemens. Just like we model our routing solution, we build our simulation model out of individual nodes that act with each other. We test our routing rules solution on four different points:

- We test the impact of the dispatching rules.
- We check how different attribute weights of the routing rules impact the output results of a beamline.
- We test whether the impact of the routing rules is the same on a busy and a quiet beamline.
- We test how the dynamic routing rules compare to the current routing strategy.

We use four KPIs to judge the results of our tests: the average relative costs per job, average output of the beamline in tons per hour, average make span per phase and average throughput time per job. The most emphasize is put onto the average tons per hour and the average make span per phase, because these indicators are the most important to indicate the production output of a beamline.

The dispatching rules MWRF and MWIMF consistently proved to be the best performing dispatching rules, on both a busy and quiet beamline. The effect of choosing between a “bad” and “good” dispatching rule is much bigger on a quiet beamline than on a busy beamline. However, differences between the “good” dispatching rules (MWRF or MWIMF) are much smaller. Because the same dispatching rules function good on both a busy and quiet beamline, we conclude that the decision of the dispatching rule should be made on the basis of the experiments on the busy beamline.

We designed the dynamic routing rules with the idea that the user can decide the focus of the routing rules by giving weights to the different attributes. We tested if the different attribute function like they should and conclude that a focus on the costs and machine utilization attribute do function like they should. A focus on the costs attribute results in the lowest average relative production costs per job and the machine utilization focus results in the most equal spread of the work over all machines. However, a focus on the throughput time attributes does not seem to be directly results in the lowest average throughput time per job. This seems to be more linked to a low focus on machine utilization.

We also tried to find an attribute focus that would consistently result in the highest output of a beamline. Right now, every beamline performs best with a different attribute focus, so we need either a new attribute that can be used for output maximation, or try to find the optimal attribute focus per beamline. At the moment, choosing the right scenario (from the scenarios we tested) can increase the tons per hour by 1% - 1.5%, compared to scenario 1.

We comparing the dynamic routing rules to the static rules that are currently used. We compare the static rules to the basic configuration of the dynamic rules, where every attribute has the same weight, and we compare them to a total of 6 configurations, where every configuration puts weight on another attribute. We also compared the static and dynamic rules, with and without the impact of the dispatching rules. Some example results can be seen in Table 0.2. We conclude that our dynamic routing rules outperform the static rules in all comparisons. The comparisons show that the dynamic routing rules improve the performance of on busy beamlines by 0.7% to 5.1% and on quiet beamlines by 6.3% to 7.8%.

Table 0.2: Comparison Beamline 1, busy beamline (results in tons per hour)

Dispatching rule	Dynamic rules configuration	Result Static Rules	Result Dynamic Rules	Improvement
RAND (no rules)	Every attribute has equal weight	96.1	100.0	4.1%
RAND (no rules)	Best scoring scenario	96.1	101.0	5.1%
All rules	Every attribute has equal weight	96.5	100.6	4.3%
All rules	Best scoring scenario	96.5	101.7	5.4%

To conclude, we designed a dynamic routing solution and we showed that it is an improvement over the current static rules, not only based on the performance of the beamline, but it will also result on the long term in less configuration time per beamline. It offers a single standardized routing solution for every beamline, that subsequently can be tweaked by the user to focus on the attribute they find important. The challenge now is to fine-tune the attributes we use to score the routes and to find out what attribute needs to be focused, in what type of beamline, to gain the highest output in tons per hour.

Finally we have some recommendations to further improve and expand the product routing and to optimize the total production process.

- *Optimization of the routing rules:* Our goal of designing a routing solution that improves the current situation is achieved. A nest step is to optimize them. Our expectation was that a focus on the “machine utilization” attribute should lead to the highest tons per hour on the beamline, because we expected that this focus would move work away from the bottleneck machine to other machines. This only proved to be true on beamline 2. So, we suggest to either try to find a optimal attribute weight setting per beamline type, or add/change up the attributes such that there is an attribute that results on every beamline type in the highest output.
- *Nesting and production planning*: Product routing is only a part of the total production process. We also applied some dispatching rules in our solution and showed that there were some positive effect to be gained from it. Production planning can have a much bigger impact on the performance of a beamline than the dispatching rules we used, since it includes all upcoming production. We could only work with the phase that arrived at the beamline.
- *Global vs local attribute weights:* In our model we use attribute weights that apply to every job, to determine the best route. The model can be extended by allowing the user to adjust these attribute weights for every job individually. This can be useful when the global weights focus on machine utilization for example, but there is a job that needs to be processed as quickly as possible. Then the local attributes weight of that job can be set to throughput time.
- *Custom outfeed location*: The modularity of the nodes allows easy manipulation of the configuration of a beamline and the generation of new routes. To extend the solution, an attribute can be added to each node, which can indicate a node outfeed location; we already use this for machines that are capable of dividing a product. When there is a product that requires a manual removal from a RC or CT, select that node as outfeed node. A new set of routes can be generated from the infeed location of the job to the selected outfeed node and the time that is required to remove the beam can be added as extra waiting time to the node that is selected as outfeed location.

- *Global vs local machine capabilities:* We use a table *machine capabilities* to indicate which machine can perform which operations against which relative costs. Based on this table all possible routes are generated. Our suggestion is to also use *Local machine capabilities* when a certain job is not allowed to be processed by a certain machine. This can be the case when a product needs to adhere to a certain quality level for example. A more general approach would be to have different product groups, where every group has their own *machine capabilities* table. For a group that requires high quality, all operations can only be performed on the machines that deliver the highest quality. This means that the set of feasible routes of a job also depends on the job's product group.

UNIVERSITY OF TWENTE.

Table of Contents

Abbreviations

AHP	Analytical Hierarchy Process
BWSF	Best Work Spread First
CR	Criterical Rate
CT	Cross Transport
HFS	Hybrid Flow Shop
KPI	Key Performance Indicator
LWBF	Least Work for Bottleneck First
LWRF	Least Work Remaining First
MCX	Machine Company X
MADM	Multiple Attribute Management
MCDM	Multiple Criteria Decision Management
MDD	Modified Due Date
MODM	Multiple Objective Decision Management
MST	Minimum Slack Time
MU	Movable Unit
MWRF	Most Work Remaining First
RAND	RANDom dispatching rule
RC	Roller Conveyor
SPT	Shortest Processing Time
SRPT	Slack/ Remaining Processing Time

1 Introduction

This research takes place at a company we call "Machine Company X" and focuses on the product routing of the automated processing system they offer, the beamline. This chapter introduces Machine Company X in Section 1.1, the challenges they encounter with regards to their product routing in Section 1.2 and the research design, including the scope and the research questions, in Section 1.3.

1.1 Machine Company X

This research takes place at Machine Company X (MCX). MCX is active all around the world in the steel fabrication and plate processing related industries.

At MCX, they design, develop and manufacture machinery for the steel processing industry. There are two major types of steel products that can be processed with the machines of MCX: steel beams and steel plates, as shown in Figure 1.1 and Figure 1.2 respectively.

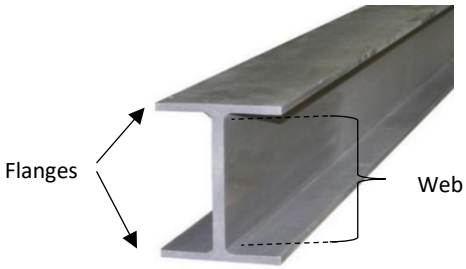


Figure 1.1: Steel beam (H-profile)

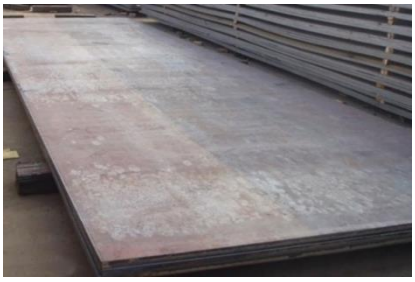


Figure 1.2: Steel plates

Chapter 2 gives a more comprehensive overview of the machines MCX manufactures and the operations these machines perform, but some examples are drills (to drill holes in plates and beams), saws (to divide beams in smaller sections) and plasma torches (to create more complex shapes on both plates and beams). Machines used for beam processing can be connected by transport systems, creating a processing line. Steel beams are automatically transported through the line on conveyor belts. Because this processing line is only available for steel beams, it is referred to as a beamline. Chapter 2 further explains the concept of a beamline.

1.2 Problem identification

This research focusses on a problem related to one of MCX projects, we call it project X. Section 1.2.1 describes what this project includes and what its goal is. Section 1.2.2 describes the issues that MCX faces with project X and determines the goal of this research.

1.2.1 Project X

The customers of MCX mostly operate in the production and processing of steel structures. The production process of these steel structures is typically divided in different phases:

1. Receive raw material (beams & plates)
2. Process beams and/or plates to create production pieces

3. Welding of plates and beams
4. Painting (often outsourced)
5. Loading on truck + delivery at building site

The goals of steel fabricators usually are:

- To deliver the right products at the right time and order to the building site.
- To keep efficiency high and cost low. (So, achieving the above against the lowest cost)

To keep the delivery reliability high, many steel fabricators create buffers between all production phases:

a. Store raw material (beams & plates) before production
b. Store half processed beams and plates before welding
c. Store semi-finished products before painting
d. Store finished product before shipping to site

Project X involves the optimization of all five phases of the steel structure production process, as well as the logistics and dependence between all phases. In this report, we are focusing on the second phase in the production process, specifically the processing of steel beams on the beamline (more thoroughly explained in Chapter 2).

With the switch to a complete system supplier, it becomes more and more important for MCX to find ways to standardize their systems in such a way that only a small amount of fine tuning is required to make a system fit the wishes of a customer. This way, MCX is able to keep delivering high quality system, while lowering the amount of effort and labor to design and program it. This standardization process has also been one of the focusses of the system engineering department the last few years.

With the beamlines, MCX has anticipated the change towards becoming a complete system supplier. A MCX beamline is a processing line for steel beams. It consists of multiple machines that are mechanically connected with roller conveyors and cross transport legs. This way, almost the entire processing is done automatically.

To control the system, it is equipped with software that includes buffer management and product routing. To give an impression on how this software currently works, here are some examples:

- In case the same operation can be performed on two different machines, the plant calculates the best option upfront, for example the machine that can perform the most required operations.
- In case two routes are possible, the plant controls decide in which direction to move the beams based on a set of configurable rules.
- In case the following beam can be taken from two different buffers, the plant controls decide which one goes first based on simple static rules. Section 2.2 further explains these rules.

The goal that MCX has in mind with this project is to investigate a dynamic routing solution that adjusts to the wishes of the customer and steers products based on the current system load.

1.2.2 Core problem at MCX

The main objective of MCX is to work towards optimization of the whole production process, as described in Section 1.2.1. This thesis focuses on one part of the total process: the beamline. Gaining full control over the behavior of this beamline is an essential step towards optimization. To make this step, we need to find out what the underlying issues are that currently withhold MCX to improve this control on the system behavior.

The beamline covers the main processing part of the customer, but there are also production processes that are not (yet) included in the beamline, like welding or painting. Part of the output of the beamline becomes input for these follow-up production steps. This means that when the beamline does not deliver enough (of the right type of) output, it becomes a bottleneck in the production process, which needs to be avoided. Right now, the production steps are not attuned to each other. Misalignments are corrected with buffers between all steps.

The beamline itself uses product routing rules to steer products the right way to make sure they visit all required machines and end up in the right place. Whenever a product reaches a point from where they can follow two different paths, the routing rules makes sure it goes the right way. Right now, these rules are static and need to be reconfigured per beamline, which can become complex. Complex rules are a problem for the software engineers of MCX. Configuring and maintaining these rules differently for each customer is a difficult and time expensive task.

Besides product routing, the production planning/dispatching also has a huge impact on how the system behaves. The sequence in which products enter the system has an impact on the occupation of both the machines and buffers. At the moment, MCX does not attune their beamline to other stages in the production process, so there has not been a need for a production planning tool

Finally, customer requirements may vary: some customers might want to minimize processing costs, maximize their output or have an order that needs to be ready for transport as soon as possible. MCX wants to deliver a system that gives the customer control over this decision. Right now, the product routing is set as soon as the beamline is installed. With the rules, MCX tries to utilize the beamline as good as possible. However, the rules are static, while the wishes or the product configuration of the customer might chance. Product can only be sent another way, by intervention of an operator. Manual intervention is something MCX wants to minimize, especially when the goal is to optimize the entire production process; one intervention might cause a chain reaction that disrupts all production steps, decreasing the performance of the system. Operators often feel the need to intervene because the rules that are used to control the system are complex and do not always function the same as the operator's instinct.

All these problems are causes of the main problem that MCX suffers from: The lack of control on the behavior of their beamline. We look at relations between the causes and find that here are three core problems that need to be solved before MXC can gain more control on the system behavior of the beamline. In the end all of the issues can be traced back to 3 core problems:

1. The current set of (product routing) rules is complex
2. The total production time that is stored in a buffer is not considered by the current routing software.
3. Steps further ahead in the production process are not taken into account

Now that the core problems are found, we look at the goals of this research. The main goals of MCX are:

- Optimizing the production process for their customers
- Minimizing the programming effort it takes per customer.

Both of these goals are related to the problem that MCX lacks the desired control over the behavior of the beamline. When they get more control on the behavior on the beamline, they can start to optimize the behavior. And when they get more control by using a clear structure or clear rules in the product routing, they can implement these routing with less programming effort per beamline. Reaching these two main goals would also achieve a lot of smaller underlying goals, like increasing the production output of the beamlines or lowering buffers. Appendix A shows an overview of all underlying goals.

1.3 Research design

In this section we discuss the research design. Section 1.3.1 determines the scope of this research. This is important because the timespan of this research is limited. In order to find sophisticated solutions and have enough time to compare and test these solutions we demarcate the focus of our research. Section 1.3.2 provides the research goal and research questions as well as an outline of what can be expected in each chapter. Finally, Section 1.3.3 determines the deliverables of this research.

1.3.1 Scope

This research takes place at MCX and focusses on the product routing of their beamline. The beamline is the processing lines, consisting of the machines and the transportation systems between the machines, that is supplied to the customer and is used to process steel beams. We look to optimize the route that products take from their infeed point on the beamline, to their outfeed point on the beamline. Because the order that products enter the beamline has such a big impact on the routing, we also include this part into our scope. We do not include the processes that take place after product leave the beamline. The customer decides at what point products need to leave the beamline, so also decides where possible the endpoint(s) of the routing will be.

1.3.2 Research questions

From Section 1.2, we derive the main research goal of this research. Achieving this goal helps us to solve the core problem of MCX.

The goal of this project is to evaluate the current routing rules used in the product routing of MCX, investigate possible improvements and design a solution that can be implemented together with the software department.

Now that the research goal is set, we formulate the research questions. Answering these questions helps reaching the goal step by step. The first step is the analysis of the current situation. How is the company currently working, what rules do they use and why do they do it this way? Knowledge to answer this question is gained from interviews with the system engineers and software engineers of MCX, who are responsible for the design and implementation of these rules. To complete the analysis of the current situation at MCX we also measure the performance of the system, using the current routing rules. In Section 1.2 we found a lot of company goals. We interview more stakeholders, find out what goals are most important to them and turn these goals into key performance indicators (KPIs) that are used to measure and compare the current situation to the situations under our new designed rules. This leads to the following research question, which is answered in Chapter 2:

Question 1: How is the product routing currently done and how well does it perform?

- *What rules are currently being used in the product routing and what is the objective of these rules?*
- *When is the product routing performing optimally according to different stakeholders?*
- *What is the current performance of the system and where is room for possible improvement?*

In Chapter 3 we look at studies that are already performed on the subject. This helps speeding up the process of designing possible solutions. To make sure that we focus the right literature, the knowledge gained from Question 1 is important, as well as discussion with the supervisors of this research.

Question 2: What literature is available that is relevant to the problem?

In Chapter 4 we use the analysis of the current situation, the earlier work related to the subject and our own insights to design solutions that achieve our research goal. This answers the following research question:

Question 3: Which routing rules/strategies can we use to solve our research question?

To compare the solutions of the previous questions and see if they are indeed an improvement over the current situation, a test model is built. It is often too expensive and difficult to perform tests with the real system, so simulation models are a good replacement tool. First, we make the choice between building a new simulation or using the inhouse simulation model of MCX. The model we use also needs to be validated before we use it to test the solutions. Chapter 5 answers the following question:

Question 4: What is a good model to test our solutions?

- *What type of simulation software do we use?*
- *What assumption need to be made?*
- *Does our model sufficiently represent the reality?*

In Chapter 6 we use the simulation model to test the solution that are found in Chapter 4. The outcome of these tests must show how well the solutions score on the KPIs that are defined in Chapter 2. This way, we can compare how the routing rules function on different beamlines and how they compare to the rules that MCX currently use. We answer the question:

Question 5: What is the expected performance of our solutions?

When all research questions are answered, a final conclusion can be drawn. We summarize the answers of the research questions, and discuss if the research goal has been achieved. We also discuss the contributions of our research towards both the practice and the literature, and give further recommendations on the subject and on possible follow up projects. These conclusions and recommendations can be found in Chapter 7.

1.3.3 Deliverables

At the end of this research, a certain result is expected. For this research, the desirable result is a solution (a set of rules for the product routing) that improves the current situation and can be implemented by the software department of MCX. Whether our solution is an improvement over the current situation or not is based on the how it scores on the KPIs we choose, based on the goals described in Section 1.2. So, we also deliver a set of KPIs that we consider useful to measure the performance of a beamline. The rules must describe what logic is used to steer products through the beamline and how it improves the control on the routing behavior. The rules must work on every beamline, independent of the layout. Per beamline, the configuration of the rules should also be an easy process. The ideal result would be that the solution is also implemented and tested at the end of the research.

2 Context analysis

Chapter 1 determined the challenges and research goals. This chapter gives some more context on the current situation of these challenges. Section 2.1 describes what kind of products a beamline can process, the machines that can be implemented in a beamline and how they are integrated with each other. Section 2.2 describes the rules that are currently used to steer the beamline. Section 2.3 identifies the stakeholders of the problem and via interviews we determine their perspectives on the problem. Section 2.4 converts these stakeholder perspectives into KPIs that reflect the priorities of the stakeholders. Finally, Section 2.5 concludes the context analysis.

2.1 Beamline

This section explains the concept of a beamline. Section 2.1.1 explains the types of products a beamline can process. Section 2.1.2 describes the possible machines in a beamline and what operations they can perform. Finally, section 2.1.3 explains how multiple machines are integrated into a beamline.

2.1.1 Product types

The beamlines are specifically configured to process steel beams. A common shape of steel beams is the H-profile, but beams can come in more shapes. The shape of the beam influences the operations that can be performed, as well as their processing times. Figure 2.1 displays all possible shapes. From left to right they are called H-profile, Flat-profile, Rectangular tube, U-profile, T-profile and L-profile respectively.

Figure 2.1: Beam shapes

The beams that are processed are used in the construction of large buildings. Every project requires different kind of beams with different characteristics. Even within a project there is a lot of variation. There is no standard layout with a certain length or a set number of holes a beam needs. In the most extreme case, every beam could have a different combination of length, number of holes, location of holes, required markings and coping (which is explained in Section 2.1.2). This variation in required operations per beam, results in a high variation of processing times per machine and makes it hard to have a constant product flow through the beamline, where the workload is always evenly spread between machines. One beam might require a lot of sawing and little drilling and the next requires the opposite, while a third beam requires a lot of both sawing and drilling.

To distinguish different types/configurations of products we use some terminology:

- **Job**: A single product, which requires some operations to become a finished product.
- **Nesting**: It is possible to only have one job per beam, but when jobs have the same beam type, multiple jobs can be processed from the same beam. This is called a nesting (Figure 2.2). Also, when there is just a single job in a beam, it is called a nesting. It still requires a cut at the beginning and end of the job to get it at the exact right length and create a clean cut at the beginning and end. Once all residual parts of the beam are cut off, it is seen as finished product. Cuts can be made at a saw or a plasma cutter (Section 2.1.2).

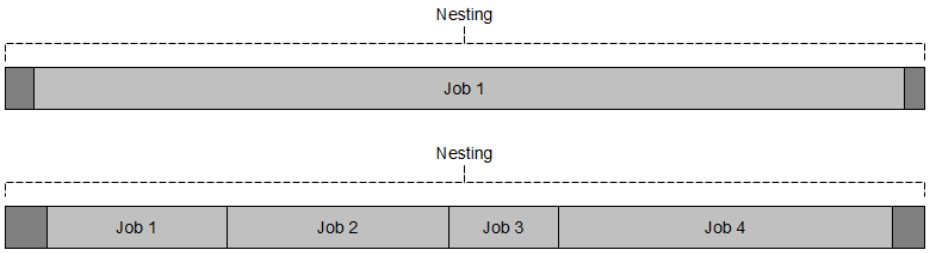


Figure 2.2: Example of a nesting

- **Beam**: When we refer to a beam, we refer to the physical beam in general. So, a beam can still contain a nesting or be a single job.
- **Batch**: Certain machines can handle multiple beams at the same time. These beams as a group is called a batch.

2.1.2 Machine descriptions

Machines

The following operations can be performed on a beamline:

- *Drilling* is the operation of drilling a hole in the beam.
- With *marking*, the machine removes a tiny layer from the surface. Usually this is used to mark lines and areas that are required during the welding process.
- *Thermal cutting* uses either plasma or oxyfuel to cut through the steel. Since it is not restricted to a drill head or saw, it has a free range of possible shapes it can cut. *Coping* is a special operation that can only be done with thermal cutting. Coping removes a corner at the end of a beam, such that multiple beams can fit into each other for example (Figure 2.3).
- With *Shot blasting* little pellets are being blasted at the beam with high pressure to clean it from rust and dirt.
- *Painting* is done by spraying paint under different angles onto the beams.
- *Sawing* is the process of dividing a beam in two.

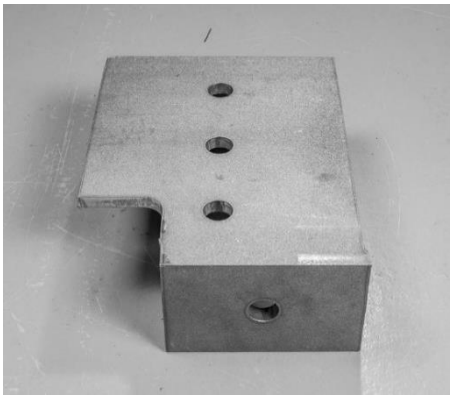


Figure 2.3: Beam with cope and holes

Table 2.1 briefly describes all machines that are relevant to this thesis. The machines are relevant when they are used in the beamlines, so we left out the machine that can only process steel plates and machines that are only sold as stand alone, not as part of the beamline. Machines often have one main function, but some are capable of multiple processes.

Table 2.1: Machine functions

Machine Type	Operation(s)	Description
Drill	Drilling, Marking	Has three drilling heads to drill both flanges and web at the same the same time, reducing processing times.
Saw	Sawing	Range of sawing machines. Different saws allow different max height of beams and sawing angles.
Marking machine	Marking	Specialized machine just for marking.
Plasma cutter	Marking, Thermal cutting, Coping, Drilling	Thermal cutting machine. Uses plasma or oxyfuel to cut any shape out of the beam. This makes it able to perform many different operations. However, due to relatively high operating costs and the amount of residue it leaves after processing, it is not preferable to perform simple tasks like marking on the plasma cutter.
Shot blaster	Shot blasting	Series of machines that blasts a stream of particle at the beams to clean them. Products can be processed in batches.

Handling equipment

Besides the machines that are used to process the steel beams, MCX also manufactures and sells equipment for the handling of steel beams. In fact, these are the best-selling items of MCX, with more than 5 km of both the roller conveyors and cross transport sold per year.

Roller conveyors

The roller conveyor (RC) (Figure 2.4) uses cylindrical transport rollers to move the beams. The distance between these rollers is important, since this determines the minimal length beams need to have. The distance between rollers determines the minimum required length for a beam to travel over a roller conveyor. Beams should be able to rest on at least two rollers at all times, otherwise they will fall between two rollers.

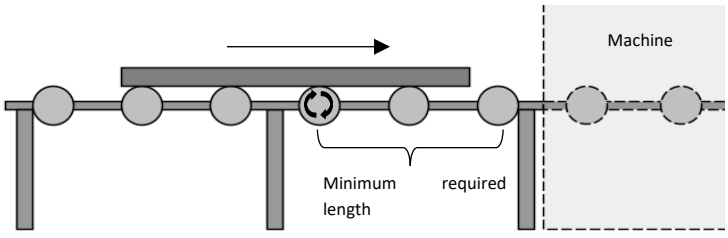


Figure 2.4: Roller Conveyor

Cross transports

The cross transports (CT) are always positioned perpendicular to the RC. They move beams on and from the RCs and also function as intermediate buffers in the processing line. Instead of moving on transport rollers, beams lie on the cross transport spokes and are either pushed by hydraulic or pneumatic collapsible notches or lifted up and moved (Figure 2.5). There are four main types of cross transports that use different methods of moving the beams, these can be found in Appendix B.

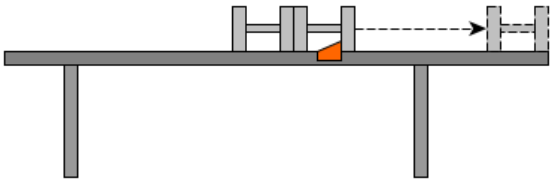


Figure 2.5: Drag dog pushing large H-profile

2.1.3 Multi system integration

All the machines that are described in Section 2.1.2 can be integrated in an automated processing line: a beamline. In a beamline the machines are connected with each other by means of roller conveyors and cross transports.

- Roller conveyors (RC) often handle the in- and output of the machines. When we look at the beamline from above and picture it on a grid with X and Y directions, the roller conveyors would move the beam in the X-direction through the system.
- Cross transports (CT) are used to move the beam in the Y-direction. They move beams between RCs and act as a buffer where beams can be stored when the upcoming machine is occupied. Cross transports can also be used as end station buffer, where beams are stored until they are picked up with a forklift or crane.

By connecting all the machines and letting them communicate with each other, MCX can deliver a fully automated system. In the ideal situation, the customer only needs to load raw beams into the system and unload the finished products at the end. This reduces manual labor, minimizes bottleneck issues and increases efficiency, while the entire process can be monitored in real time. The system uses MCX's own control software algorithms to predict and execute load-balanced material routing, reducing bottlenecks and utilizing each machine to its full potential. The control software of MCX is called VACAM.

There is no standard configuration for the beamline. Every beamline is a unique system, adjusted to the needs of the customer. Considering all possible machines that can be used in a beamline, there are a lot of possible configurations.

2.2 Product routing

Product routing can be split into two parts. First there are the routing rules that steer the product through the system and decide which machines will be visited. Second there is the buffer management which controls the movement of parts within the buffers. Section 2.2.1 explains the routing rules and Section 2.2.2 describes the buffer management.

2.2.1 Routing rules

The routing rules that are currently being used are connected to routing nodes. In a system there can be the following four types of nodes: System infeed, system outfeed, splitting and merging nodes. In VACAM, rules are linked to these nodes. Figure 2.6 shows how these nodes function.

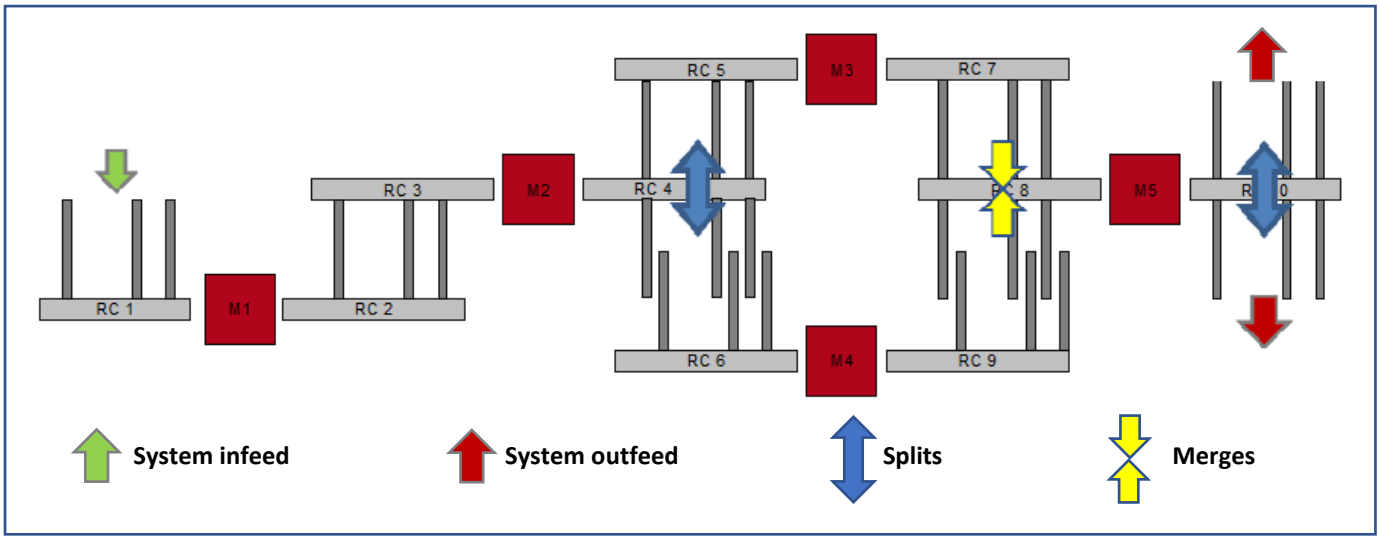


Figure 2.6: Routing nodes

System infeed: A system infeed is a location where it is possible to load material into the system. Usually, they are located on CTs at the beginning of the beamline. Per infeed location is stated which characteristics are required for a beam to be loaded at the infeed. After an operator loads a beam onto the infeed, they provide the system with the info of the beam. A beamline can have multiple infeed locations.

System outfeed: A location where it is possible to unload material from the system. Outfeed locations are usually located at the end of the beamline, but it is also possible to have outfeed locations at or just after a saw or plasma cutter. At these location beams can be cut into smaller pieces and they can become too short to travel further on the beamline, which means that there needs to be an outfeeds location.

Splits: Splits are the points in the system where the material flow is divided. Materials arriving at a split can continue in 2 or 3 directions, so the most important routing decisions are made at the split locations. When beams arrive at a split, it basically follows a checklist to decide which direction they should go. This could for example be based on required machines, minimal length of the beam or the designated outfeed of the beam. When both directions meet all points of the checklist, the direction that leads to the CT with the most free capacity is often chosen.

Merges: The points in the system where material flows come together from 2 or 3 different directions. The rules used at a merge node are used to decide which direction gets priority. The most common rule gives priority to the side with the least free capacity. So, at a merge of two CTs, the CT with the least free space gets priority. It could be that there is a merge of a buffer and an infeed. Here, the infeed usually gets priority since there is less room to store beams.

2.2.2 Buffer management

The buffers of the beamline are located on the cross transports. They can be in front of machines or at the end of the system and allow for storage when beams are not directly needed. When we take the beamline in Figure 2.6 for example, the direct buffer for machine M3 would be located on the cross transport between roller conveyors RC4 and RC5.

Within a buffer, beams cannot overtake each other. The goal of buffer management is to use the buffer space as efficiently as possible, moving beams as far to the end as possible and making sure that machines can get a new beam as soon as they are available. These rules are determined with the following objectives in mind:

- To make sure that machines run as efficiently as possible
- To optimally use the capacity of cross transports

- To create batches for the shot blaster and paint line

MCX's own control software tracks the position of each beam in the system, so it knows for example how many beams are located in each buffer. It also knows all the operations that need to be performed on a beam. However, it does not yet use or store how long a machine will be busy on a certain beam. The processing time per machine is dependent on many different variables.

2.3 Stakeholder perspectives

This research has a global research goal, described in Section 1.3, however, this goal can be interpreted differently by different stakeholders. In this section, we try to identify all stakeholders and their ideas of how the system should be functioning. All information is gathered via interviews with the concerned stakeholders.

Product manager

The product manager is in the employee within MCX who is responsible for the beamline as a 'product' that is delivered to customers. The software and hardware are seen as two different products and have two different product managers. In the end though, both have the same goal: Delivering an optimized product to the customer.

It is important that the customer gets the machines they require for their expected production and can use these machines as efficiently as possible. One focus of the product routing should be the utilization rate of the machines. This does not always mean that every machine should be performing at maximum capacity; sometimes it is more useful to focus on the utilization of the bottleneck machine and make sure that machines that should be running at all times always have a big enough buffer.

The product manager is also interested in a proper way to present the performance to the customer. Routing rules should be clear and easy to understand for the customer, such that they know how the system operates and understand why a certain performance is achieved. Then, when the customer is not satisfied with the performance, it is easier to pinpoint where things go wrong and where the configuration of the beamline can be adjusted.

Software engineers

The software engineers are responsible for the coding of the routing rules. Right now, the configuration of this is a time intensive process. Every time there is a split in the system, it needs to be explicitly stated under which conditions a beam can travel in a certain direction. The rules can be as easy as load balancing, where the beam is sent in the direction with the least number of beams in the buffer. It can also become more complex when beams need to visit certain machines and outfeeds after the split or when transport systems have a required minimum beam length. During the years, more and more of these exceptions are encountered and are added to possible characteristics of a beam that influence the decision at a split. Every beamline and every split in those beamlines have different exceptions, so configuring every split takes a lot of time.

System engineers

The system engineers seek to standardize all processes in the company, to be able to craft fitting solutions per customer with as little tweaking as possible, so the routing should be easy to configure and maintain per customer. This means that rules should not be too system specific, but applicable for every system.

Part of MCX's Project X is to include a planning tool for the customer. Right now, beams are loaded onto the beamline with a push principle in mind. An operator loads beams on the infeed of the system, then gives this as input in the software and the software knows what beam entered the system. There is no plan made upfront, so the sequence in which the beams are loaded onto the beamline fully depends on the operator. The goal is to change this into a pull system, to have a plan for the operator and show him in which sequence beams need to be loaded onto the beamline, based on the demand and the current status of the beamline. Product routing helps by predetermining the route a beam takes on the beamline and offering an expected flow time or completion time.

The routing rules of a beamline are configured before it is installed, during the 'commissioning phase'. After this, they are static and cannot be changed by the customer. One goal of system engineering is to make this more flexible for the customer, to let the customer choose between rule sets that focus on different objectives, i.e. minimizing flow times or minimize the production costs.

Customers

According to customers, the problem that causes most of the production loss is when one machine gets too much work and cannot process it fast enough. This does not only increase waiting times for that machine, but sequential machines will be standing idle, resulting in a big efficiency loss. So, it is important that workload is spread over machines, alternating products that are drill intensive with products that are saw intensive for example. It would also be preferable when machines automatically substitute for each other when possible, for example when the marking machine has too much work, the drill could take over some marking work. Right now, the operator needs to manually set when the drill is allowed to perform marking operations. Automating this decision would relieve the operator of this task and remove the variation that is created by human interaction.

For some operations, an operator is needed to complete the task. For example, the plasma cutter always needs an operator to inspect the operation, small parts need to be manually removed from the saw, when not equipped with a Short Product Removal System (SPRS), and tool changes of a drill require an operator. Right now, operators are often assigned per machine and act when necessary. This might be needed every product, but it is also possible to have long idle times between two actions. Having a product routing that can also tell the user exactly when a beam that requires an action from an operator allows for more flexible and efficient assignment of operators.

The pull principle that system engineering wants to work towards also corresponds to the wishes of the customer. The welding department often consists of a steady workforce. Every minute they are idle is a waste of money. So, they should always have enough supply to continue working. In the ideal situation products leave the beamline just in time before they are needed by the welding department. An alternative is to create a small buffer between the beamline and the welding department. To make this work, the production planning and routing on the beamline should also consider the demand of the welding department.

2.4 Production analysis

Section 2.3 shows what is important to different stakeholders. However, to use these perspectives in a measurement, we need to quantify them. We do this by creating key performance indicators (KPIs) in Section 2.4.1, based on the perspectives. When the KPIs are established, we would like to use these to measure the performance of the routing rules of a current beamline. However, the beamlines are not owned by MCX, they are sold to the customers, which makes it hard to measure the performance of these systems and gain the required data, within the timeframe and scope of this research. So, for this research we do not use actual measured performance of a beamline. What we can do, is to analyze the characteristics of all components of a beamline and simulate the current situation. In Section 2.4.2 we analyze the

machines and transport systems that can be included in a beamline. The data from this analysis is used to perform the simulation in Chapter 5. Section 2.4.3 describes the bottlenecks of the current situation.

2.4.1 KPIs

Overall, the perspectives of the stakeholders focus on three different areas: production output, operating costs of the system and usage/control of the system. The perspectives of the different stakeholders match on almost all of these areas, which can be expected, since the customers want a good product and MCX wants to be able to deliver this. Table 2.2 shows the KPIs extracted from the stakeholder perspectives.

Table 2.2: Quantitative KPIs

Quantitative	Description	Unit
Throughput time of a job	The time a job spends on the beamline	*Seconds*
Machine utilization	The percentage of time a machine is being used	
Throughput of the beamline	The number of jobs being produced per time unit	*Tons/hour*
Required configuration time for a new beamline	The time it takes for the software engineers to configure the routing rules for a new beamline	*Hours*
Operation costs	The costs of processing a job on the beamline	*Euro*
Deviation from due date	The time a job deviates from its due date, either in late or early	*Seconds*

There still remain some stakeholder wishes that cannot be transformed into a quantitative KPIs and remain qualitative. The qualitative KPIs cannot directly be used to measure the performance of the beamline but are important to keep in mind when designing the solutions. Having a solution that performs well on the quantitative KPIs for example but does not allow the customer any control over the system still contradicts the wishes of MCX. Table 2.3 shows the qualitative KPIs that are considered.

Table 2.3: Qualitative KPIs

Qualitative	
“Control” over the system	How easy is it for the customer to adjust the output of the system?
“Easy to maintain”	How easy is it for the software engineers of MCX to maintain the routing rules, for example when a beamline or its product range changes? Do the rules need to be completely redesigned or is a simple readjustment enough?
Bottleneck identification	Can bottlenecks be identified and do the rules react to this?

2.4.2 Processing time analysis

To determine the processing times of machines, we use previous time analysis of MCX. This gives us the set-up and processing times for almost all machines. Only the processing times of the saw are missing and still need to be determined. To determine these processing times, we perform our own time analysis, which can be found in Appendix C.

Table 2.4 shows the results of the time analysis. This time is divided in the set-up time, which is the total time of all activities that are independent of the product, and the processing time, which is dependent on the properties of the product. The set-up time includes for example the clamping of the product to make sure it lies in the right position, resetting the measurement systems and the selection of the right drill tool. These actions are necessary for every product that is being processed and have similar times for every product, so for simplicity they are summarized as set-up times. The shot blaster does not have a setup time, beams can just be moved from the RC into the machine without preparations like measuring or clamping.

Table 2.4: Processing Times

Machine	Set-up time (s)	Processing speed	Capable of batch processing
Saw	18,5	(Table 2.5)	No
Drill	17,5	(Table 2.6)	No
Marking machine	13	15 mm/s	No
Shot blaster	-	1,6 m/min	Yes
Plasma cutter	67,5	Coping: 35 s/cope Marking: 15 mm/s	No
Roller Conveyor	-	36 m/min	Yes

The coping time of the plasma cutter is based on the assumptions of MCX. The exact times can only be gained from the software that makes the pathing for the coping. And even then, there is variability, because material that has been cut off needs to be manually removed by an operator.

The processing time of the saw is based on the area the saw needs to go through. Figure 2.7 shows a cross section of a beam with a H-profile. The area the saw needs to go through is in that case the area of both flanges, plus the area of the web. Sawing a beam under an angle, increases the area, this is taken into account in the calculation of the area.

Table 2.5: Processing Times Saw

Profile Type	Saw time (s) X = the area (mm^2) where the saw needs to go through (Figure 2.7)
I-Profile	$0.3216 * X^{0.5973}$
U-Profile	0.0047 * X + 11.761
M-Profile	0.0061 * X + 18.051

X = Area of the flanges + Area of the web

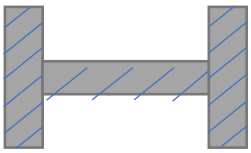


Figure 2.7: Area of a beam

Table 2.6: Processing Times Drill

Hole Diameter	Drill speed (mm/s)
[1 – 18]	0.1633
[19 – 25]	0.1866
[26 – 31]	0.1941
[32 – 40]	0.2059
[41<]	0.2259

2.4.3 Bottlenecks

According to customers the beamline itself is the bottleneck in the entire production process of a steel fabricator. Other steps in the production process, like welding, can relatively easy be expanded by hiring extra (temporary) work force. The beamline is a static system and once it is installed, expanding it is a very big investment and very uncommon for that reason. So, optimizing the use of the beamline is key in optimizing the whole production process. Within the beamline, it is hard to tell which machine the bottleneck is. This fully depends on the products that must be processed, and which machines are installed in the beamline, so it can occur that the bottleneck shifts from one machine to another when the product composition changes. From customer experiences, the machines that form the bottleneck most often are the drill and saw.

Although not a direct bottleneck of the beamline, the lack of production planning is a factor that could have a negative impact on the efficiency of the beamline. Production planning right now is often not done at product level, instead it is done per phase. A phase is a set of jobs that need to be ready at the same time. For example, a big building project is being built phase by phase, where every phase often contains 25 tons of steel. It is set at that weight, because this is the common weight a truck may transport at the same time. For the planning right now, it does not matter in what sequence jobs enter the system, as long as a phase is finished before its due date.

Routing rules are static and are set before the system is being installed, in the commissioning stage. All deviations from the ruleset require manual input. When the standard setting of a drill is to not perform any marking operations, but it needs to take over some work of the marking machine, an operator needs to manually activate and deactivate the marking function of the drill. When the routing rules need to change as a whole, because the product composition has changed for example, the software engineers of MCX need to pay a visit on site to install the new routing rules.

The routing of the beamline does also not use system data such as expected processing times or waiting times. Beams are given a route based on static constraints, like minimum required product length of transport systems, upcoming machines or the number of beams in the upcoming buffers. Because the expected processing and waiting times are not used this way, there has been no need to put much effort in calculating and storing them right now.

All machines have an infeed roller conveyor (RC) in front of the machine, which is loaded by a cross transport and delivers the product(s) to the machine. They also have an outfeed RC, which takes the product(s) from the machine and positions it such that a cross transport can pick it up. The infeed RC can only be loaded and used once the following machine is completely done with its operation and emptied by the outfeed RC (Figure 2.8). For some machines, like the saw, a measurement truck that pushes the beam needs to move back first, which means that there is some delay before the next beam can be loaded on the infeed RC. After the beam is loaded on the infeed RC, it moves towards the machine, but can

only enter the machine once the outfeed RC is emptied and standing still (Figure 2.9), otherwise, the movement of the outfeed RC might influence the measurement and positioning of the product in the machine.

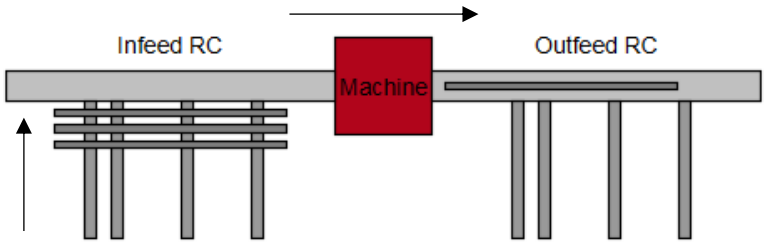


Figure 2.8: Loading Infeed RC

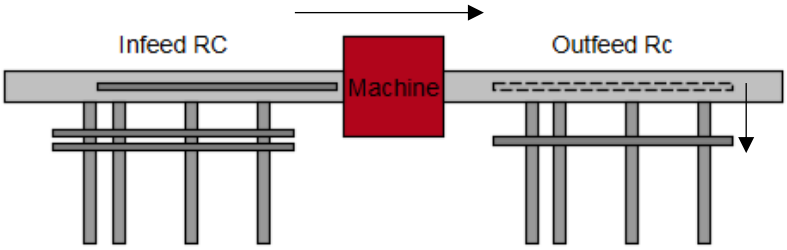


Figure 2.9: Loading Machine

Buffers are located on the CTs. They only use the First In First Out (FIFO) principle to determine the sequence in which products leave the buffer. The buffer capacity is limited and once a buffer is full, the machine previous to the buffer is not able to process anymore products, because its outfeed RC cannot be emptied. So, when for example there is too much work for the saw in the beamline, its buffer will overflow, blocking the RC before the buffer, which may as a result block the machine before it and so on. So, it is important to notice a bottleneck early on, before it causes a blockade on the beamline.

2.5 Conclusion

In this chapter, we look at the variation in types of beams that can be processed. We use different terms to clarify what type of steel beam we are talking about. A “beam” refers to a physical steel beams in general. Before a beam has reaches a saw, it contains a nesting. This nesting can consist of a single or multiple job(s). A job is a single product, which requires some operations to become a finished product. There can be lot of variation in the amount of jobs and required operations per beam. All this variation per beam makes it complex to plan and route the products in such a way that all machines are used efficiently.

This research focuses on the beamline of MCX, a processing line consisting of multiple machines that are connected with roller conveyors and cross transports. Every machine in the beamline can perform one or more of the following operations: drilling, marking, thermal cutting, shot blasting, painting, sawing. Roller conveyors are purely used for transportation, while cross transports can also be used as buffer. At a beamline, there are different routing nodes that are used for the product routing: system infeeds, system outfeeds, splits and merges. The routing is done by static routing rules per node. Besides product routing, the beamline also uses buffer management. This is used to make efficient use of the buffers.

Different stakeholders are interviewed to determine their perspective on how the beamline should perform. These perspectives are transformed into quantitative KPIs to measure the performance of the beamline and qualitative KPIs that are considered during the design of the routing rules.

Within the scope and time frame it is not possible to gather performance data of an actual installed beamline. Instead, the characteristics of all the component of a beamline are analyzed. The processing and set-up times of the machines are determined, as well as the speed of RCs. We use these times to simulate the beamline in Chapter 5.

3 Literature review

The focus of this thesis is the product routing on the beamline of MCX. How and what type of routing can be used depends a lot on the characteristics and requirements of the manufacturing process (Krajewski, Ritzman, & Malhorta, 2007). So, in Section 3.1 we define the type of manufacturing layout the beamline can be described as and what characteristics belong to that manufacturing layout. There are two major aspects that influence the behavior of a beamline: product routing and dispatching rules (as part of the buffer management). Section 3.2 describes different methods of product routing for the chosen type of manufacturing layout and Section 3.3 describes the dispatching rules. Section 3.4 explains what performance criteria are commonly used in the literature. Finally, Section 3.6 gives a conclusion on the literature that is reviewed.

3.1 Types of manufacturing layouts

The kind of routing that is required depends on the type of manufacturing process. Literature describes that there are five types of manufacturing processes: project, jobbing, batch, line and continuous processing (Hill, 1985). These processes can be grouped in two with on one side line and continuous processing and on the other side jobbing and batch processing. An example of a line processing layout would be a flow shop, while job and batch processing can be done in a job shop. Section 3.1.1 and Section 3.1.2 describe the flow shop and job shop respectively.

3.1.1 Flow shop

Figure 3.1 shows an example of how a flow shop can be set up. In a flow shop, every product requires the same processing steps. This means that machines can be set up in such a way that allows products to follow a logical path from machine to machine with minimal travel times. A flow shop also offers an easy opportunity of automatization, where transport from machine to machine is automated.

We can look at a flow shop as a process with *m* stages, where each stage k has $M(k) = 1$ machine and all products pass through all stages in chronological order, from stage 1 to stage m.

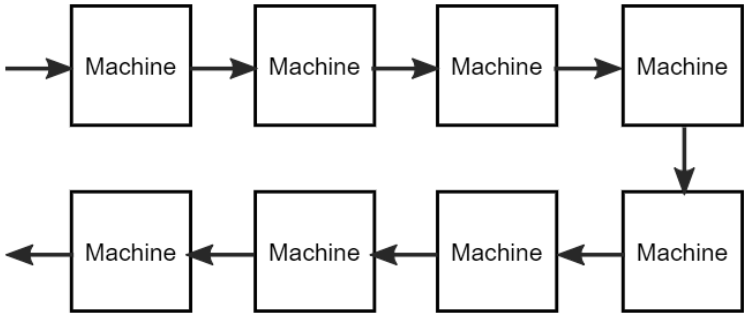


Figure 3.1: Flow shop (Owen-Hill, 2017)

3.1.2 Job shop

In contrast to a flow shop, products do not follow a standard path through all machines in a job shop. Every product might need to be processed by different machines in a different order. We can still use the same notation, in which a job shop process has m stages with $M(k) = 1$ machine per stage, but products do not step through stages chronologically. They might skip or go back stages or even revisit a stage. Figure 3.2 depicts how products could travel through a job shop.

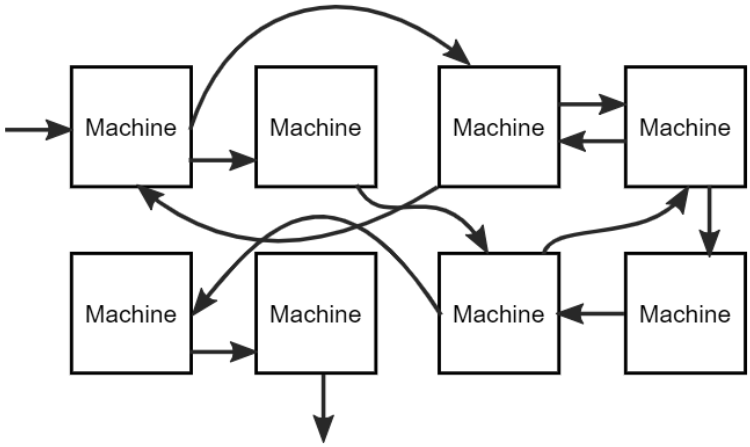


Figure 3.2: Job shop (Owen-Hill, 2017)

The beamline can be classified as a type of flow shop. Beams enter the beamline at an infeed point and from there they can only move forward through the processing line, until they reach their designated outfeed point. However, there are deviations from a normal flow shop, there can be multiple machines per stage for example. That is why we take a look at the concept of a hybrid flow shop in Section 3.1.3.

3.1.3 Hybrid flow shop

A hybrid flow shop (HFS) shows some deviation from a normal flow shop. According to Ruiz & Vázquez-Rodriguez (2010), a hybrid flow shop can be characterized by the following points:

1. The number of processing stages S is at least 2.
2. Each stage s has $M(s) \geq 1$ machines in parallel and in at least one of the stages $M(s) > 1$.
3. All jobs are processed following the same production flow: stage 1, stage 2, . . ., stage S. A job might skip any number of stages provided it is processed in at least one of them.
4. Each job j requires a processing time p_{js} in stage *s*. The operation of job *j* in stage s shall be referred to as operation o_{js}.

There has been quite some research on HFS scheduling, starting with Johnson (1954). However, a lot of this research has been performed on static flow shops that assume all jobs and machines are available at time zero (Meyyappan, Saygin, & Dagli, 2007). Also, a lot of studies assume homogeneous machines per stage, all being able to perform the same operations (Peng & Mcfarlane, 2004). Both of these assumptions do not hold for the situation at MCX.

Another difference between this research and many previous studies is the focus on product routing, instead of scheduling. At MCX, they do not know how the production planning will be at the customer that purchases their system, so how the product routing is designed should be independent of any eventual planning. That is why we review the literature that focusses mainly on routing, instead of production planning and scheduling.

Besides an HFS, there is also the concept of a flexible flow shop (FFS) (Naderi, Khalili, Taghavifard, & Roshanaei, 2008). In an FFS, jobs do not have to be processed at every stage and can ignore one or more stages. When a job does not have to be processed in stage s, it still needs to travel from stage s -1 to stage s + 1. This can either via a machine or direct means of transportation. Combining an HFS and an FFS gives a hybrid flexible flow shop (HFFS), where stages can have multiple machines and it is possible for jobs to not be processed at every stage (Naderi et al, 2008). However, Ruiz & Vázquez-Rodriguez (2010) do already include the skipping of stages in their definition of an HFS.

3.2 Routing strategies

Sanchoy & Prashanth (1997) define a route as the set of work centers or machines through which a product is processed, and the processing times at these work centers. We expand this definition by stating that it also should define the order of the work centers and the operations that need to be performed per work center. Every production facility or line has a set number of possible routes that a job can follow. Jobs can be given a route either in a static or dynamic way. In a static way of routing, jobs are given a route before they enter the production process and stick to this route until they are finished. When the set of products is known, static routing can be combined with production scheduling to create a production plan. With dynamic routing, the route of a job is updated per decision moment or split in the production process. Depending on real time data such as waiting times, breakdowns or bottleneck machines, the product can stick to its original route or change to a route that is more beneficial, for example based on the expected completion time. We look in the literature how previous studies handle these decisions.

A big advantage of an HFS is the capability of processing a product using multiple routes through the machines (Ghosh & Gaimon, 1992). With routing flexibility, the possibility of reaching the full capability of the system is greatly enhanced with an efficient alternate routing algorithm (Ozmutlu & Harmonosky, 2005). Many methodologies for alternate routing that currently exist are off-line methods based on mathematical programming. However, the fact that the routing of an HFS contains many dynamic features makes it very hard for a mathematical method to find a solution within reasonable time. This means that they become too computationally inefficient for making real-time routing decisions in an HFS (Peng & Chen, 1998). The HFS problem is proven to be strongly NP-hard and even small-sized problems, with only two stages, may require considerable computation time (Ruiz & Vázquez-Rodríguez, 2010). (Meta)heuristics are a viable option to reduce computation time, while still being able to find a near optimal result.

3.2.1 Threshold-based alternate routing

Routing decisions happen in real-time, so the time that is needed to make a decision could be extra time a machine is idle. This is why Ozmutlu & Harmonosky (2005) come up with their own simple real-time alternate machine selection criterion, named threshold-based alternate routing (TAR), in order to minimize mean flow time. TAR provides per product that enters the system a main routing, with alternative possibilities. Products are only sent to an alternative machine when the benefit in terms of waiting time of choosing that machine is greater than a certain threshold value. This waiting time is only based on the buffer for the first machine that comes next. They also make the decision not to exclude machines that suffer from a breakdown, since it could be possible they offer shorter queue times, including the expected time of a breakdown. Ozmutlu & Harmonosky (2005) also observe that system utilization levels decrease with a decrease of mean flowtime of products, creating an opportunity to increase the capacity of the system. This relation can be explained with Little's law, which states that the long-term average number of products in the system is equal to the long-term average arrival rate of products multiplied by the average time a product spends in the system (Little, 1961).

A major strength of TAR is the simplicity of its application. Only the threshold value needs to be determined. This can be done per system and only needs to be recalculated when there is a significant change in the system or strategy. What makes it less useful is the fact that it only looks at the next machine, it does not consider the rest of the route. Recalculating the threshold only at significant changes also sacrifices a lot of flexibility and responsiveness for simplicity.

Other researches show that it is useful to base the routing on shortest waiting times. Peng & Mcfarlane (2004) compared three different routing strategies: breakdown avoidance, shortest queue and availability proportion routing strategy. They used an adaptive agent-based approach to test which strategy resulted in the shortest flow time. Jobs are given one of the three routing strategies, with one third chance to get each strategy. When a job has a positive impact on the flow time, the strategy of that job gains a bigger chance to be given to following jobs. In the end, the best strategy should be assigned

to a dominant proportion of the jobs. The shortest queue strategy showed the most promising results with regard to minimizing flow times.

3.2.2 Swarm based real time routing

Meyyappan, Saygin, & Dagli (2007) base their real time product routing of the natural system of a wasp colony, in the form of a swarm based real time routing (SBR). They derived their strategy from the model of Cicirelly & Smith (2001, 2004). In this model, machines have a certain threshold to accept a product. This threshold decreases when a job can be processed by the machine or the machine is idle and increases when the queue reached a certain amount. Jobs on the other hand have a certain stimulus. This stimulus is increased, the longer a job is waiting, has high priority or is approaching its due date. When the stimulus of a job is higher than the threshold of the machine, the machine bids on the job. When two machines bid on a job, the machine with the shortest queue has a higher chance to win the bid and process the job.

Meyyappan et al. modify the model of Cicirelly & Smith in three steps. In the first step they include sequence dependent setup times in the bidding process, lowering the chance a machine gets a job when it requires additional setup time. Step two changes the model from time-driven to event-driven. Instead of updating the threshold parameters every 12 seconds they are updated every time an event happens. This approach reduces the computational intensity of the model and leads to more robust system behavior. The third step uses fuzzy rules to dynamically update a machine's threshold based on its throughput, the number of setups and average cycle time of a machine. As a result, this model outperforms previous models on these three measurement criteria and forms a more flexible and scalable approach to real-time routing.

3.2.3 Dynamic routing based on due dates

Weng, Wei, & Fujimura (2012) look at a HFS that has sequential steps. To minimize intermediate storage between the HFS and the sequential steps, jobs have a due date that is just in time for the start of the sequential steps. To minimize the storage, both the earliness and tardiness are minimized. So, the idea is to choose the route that has the forecast completion time the closest to the due date. Weng et al. designed a basic routing strategy with some possible additions.

The *basic routing strategy* (BRS) chooses the machine that in the end results in the least deviation from the due date of the job. When a job arrives at stage s, a forecast completion time will be calculated for every relevant machine at that stage. This forecasted completion time is based on the expected waiting and processing time of the machine at stage s plus the expected remaining time of the route after stage s.

Feedforward of demand change (FDC) takes into account the incoming rate of jobs. When a job arrives in stage 1 the queue time is compared with the queue time when the previous job arrived. When the difference in queue length is positive, this increased waiting time is also expected in the queue of some later (bottleneck) stages. When it is negative, the opposite is expected. This way, demand changes are noticed and reacted to early on. However, it is not certain that an increased waiting time in the early stages also occurs at later stages, so it might overemphasize demand changes.

Inclination of early machines (IEM) prioritizes the use of machines that finish the job earlier than the due date (and named these early machines) over machines that finish the job later (tardy machines). Both machines may have a same deviation from the due date, but via the early machine the job is expected to finish earlier than the due date, while via the tardy machine the job is expected to finish later than the due date. To take into account possible delays along the route, early machines are preferred over tardy machines in early stages. In the final stage this does not matter anymore, the machine that is closest to the due date will be chosen.

Consideration of time flexibility (CTF) takes into account the uncertainty in processing and transport times. This is done by moving the due date earlier and introducing an adjusted due date. At earlier stages, the route until the end is longer, so there is more uncertainty along the way. This is taken care of by adjusting the due date more at earlier stages than in later stages.

Weng et al. concluded that using the BRS in combination with all additional rules gives the best result, when minimizing both earliness and tardiness. However, using only one or two additional rules with the BRS gives only slightly worse results.

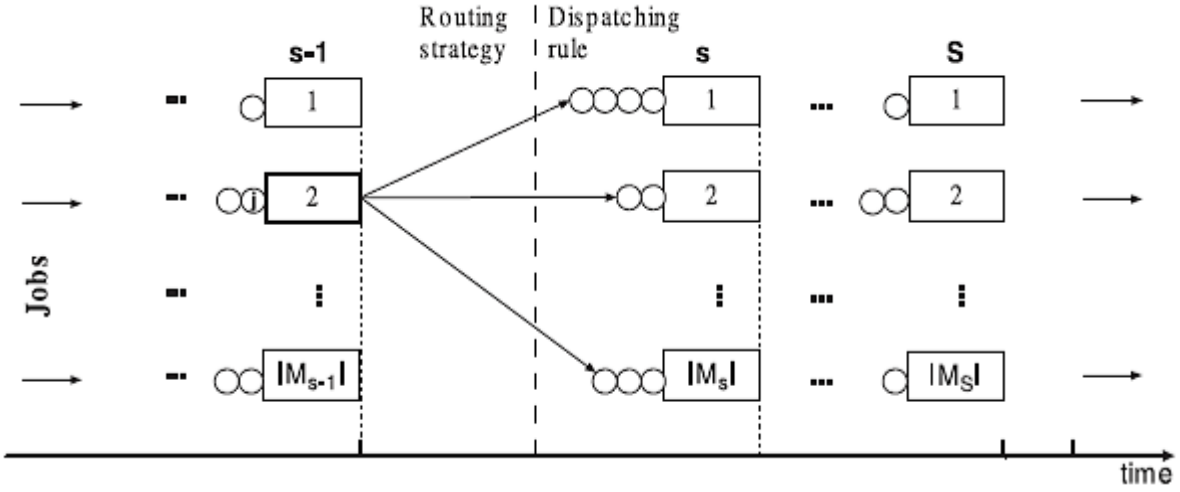


Figure 3.3: Routing and Dispatching (Weng, Wei, & Fujimura, 2012)

3.3 Dispatching rules

Besides the routing rules of the beamline, dispatching rules also influence the behavior. Figure 3.3 depicts the difference between routing and dispatching. Where routing rules decide which machine handles a job, dispatching rules decide in which sequence jobs leave a buffer, so in which sequence they are “dispatched” from the buffer. At the moment the beamline uses only the first in first out (FIFO) rule in its buffers, but there are many other possible dispatching rules.

Besides different routing strategies, Weng, Wei, & Fujimura (2012) also research the influence of different dispatching rules. Besides the FIFO rule, they look at rules that are most often used for comparison in dynamic HFS problems (Parthanadee & Buddhakulsomsiri, 2010).

- Minimum-Slack-Time (MST) selects the job with the lowest slack value, where slack is the due date substracted by the current time and the processing time of the job.
- Slack/Remaining-Processing-Time (SRPT) takes the slack value of a job and divides it by the processing time it still has in the remaining stages, then chooses the job with the lowest SPRT value.
- Earliest-Due-Date (EDD) chooses the job with the earliest due date.
- Modified-Due-Date (MDD) selects the job with the earliest due date or the completion time when the job is finished after the due date.
- Critical-Ratio (CR) compares jobs by dividing the remainting time until the due date of the job by the processing times of all stages that the job still needs to visit. The job with the mimimum value is chosen.

Weng et al. (2012) compared these dispatching rules in combination with their routing strategies and find that FIFO and SRPT showed the best results in minimizing both the earliness and tardiness. Brah & Wheeler (1998) also researched the effect of dispatching rules. Some of the rules they researched form some interesting additions to the rules that Weng et al. researched.

- Shortest-Processing-Time-First (SPT) selects the job with the shortest processing time for the upcoming operation.
- Most-Work-Remaining-First (MWRF) selects the job with the most work remaining.
- Least-Work-Remaining-First (LWRF) selects the job with the least work remaining.

Comparing the rules to a rule where jobs are selected at random, SPT shows very positive results when it comes to minimizing mean flow time and make span. MWRF is useful when make span needs to be minimized but has a negative influence on the mean flow time. Compared to MWRF, LWRF has the opposite effect on both the make span and mean flow time. Brah & Wheeler also researched the effect of FIFO and Last-In-First-Out (LIFO), but did not find a significant difference in make span or mean flow time compared to a rule that randomly selected a job. This can be explained by the way the jobs are generated in the testing simulation they used. All jobs were generated at time 0 and given a processing time from a uniform distribution. This way the simulation generates a (semi-)random set of jobs, so only selecting on which job enters first or last will still result in a random order. However, when the set of jobs is not random and enters the system in a predetermined sequence, LIFO and FIFO could have a bigger impact compared to a random dispatching rule.

3.4 Measuring performance of a flow shop

Two broad classes of measurement criteria are the cost of a flow shop (including production cost, overtime cost and inventory cost) and the performance of a flow shop (including throughput times, make span and tardiness of jobs) (Linn & Zhang, 1999). The performance criteria can be further divided in those based on flow time and those based on due dates. Linn & Zhang indicate that most researches of the HFS problem focus only on a single criterion, while in practice it is often common that multiple criteria are considered. Of the eighteen studies that Linn & Zhang study, twelve studies use the minimum make span as objective. Make span can be defined as the amount of time, from start to finish for completing a set of jobs (Pacini, Mateos, & Garino, 2013). Make span as a measurement criterium is a useful tool to check the effectiveness of a production schedule. Other studies have objectives like minimizing the maximum tardiness (Vignier, 1996), idle time of machines (Narasimhan & Panwalkar, 1984) or total flow time (C. & D., 1992). We already encountered the objective of minimizing both earliness and tardiness for JIT systems (Weng, Wei, & Fujimura, 2012). This objective is useful when there is limited buffer space between the main process and sequential processes.

3.5 Multi-Criteria Decision Making

Measuring the performance of a system based on multiple criteria and comparing the measurements of different scenarios is not as easy as it sounds, especially when the measurement criteria do not have the same measuring units or importance. To help in the decision making and comparing of different scenarios we look in the literature at some different multi-criteria decision making (MCDM) approaches. Kahraman (2008) describes two approaches to MSCM problems: multiple objective decision making (MODM) and multiple attribute decision making (MADM). MODM tries to find the best alternative by considering the tradeoffs within a set of interacting design costraints. The number of alternatives is effectively infinite, where MODM provides a mathematical framework for designing a set of decision alternatives. After that, each alternative is judged by how close it satisfies the objective(s). MADM on the other hand, combines the information of the problem's decision attributes with information from the decision maker to determine a final ranking or selection from the decision attributes. The MADM approach seems more interesting for this research, since we know our KPIs already and need to rank them to the wishes of the customer (the decision maker).

Kahraman (2008) describes 20 different MADM approaches. We are interested in an approach that can handle attributes that not measured in the same units and can be used to rank or weigh the attributes. We also want every attibute to be taken into account. Approaches like the lexicographic approach look which alternative performs the best on the most important attribute and ignore the rest of the attributes when there are no ties, so these type of approaches are not interesting for this research. We also do not want to use target values in the decision of the best alternative, which is used by approaches like TOPSIS and Distance from Target. The MADM approach that seems to fit this research the best is the Analytical Hierarchy Process (AHP). AHP lets the decision maker make pairwise comparisons between every attribute based on how much more important one attribute is than the other. These pairwise comparisons are translated into priority vectors between 0 and 1 and with a sum total of 1.

Let us give an example of how to perform the AHP. We use an example with three (m=3) attributes: α_1, α_2 and α_3. We make pair-wise comparisons between every attribute and score them according to the value interpretation in Table 3.1 (Kahraman, 2008).

Table 3.1: Value interpretation AHP

Value of α_{ij}	Interpretation
1	Objective i and j have equal importance
3	Objective i is weakly more important than objective j
5	Experience and judgment indicate that objective i is strongly more important than objective j
7	Objective i is very strongly or demonstrably more important than objective j
9	Objective i is absolutely more important than objective j
2, 4, 6, 8	Intermediate values

We fill in the pairwise comparison matrix Table 3.2 and get Table 3.3. In the comparisons, we think α_1 is demonstrably more important than α_2 and weakly more important than α_3. We also think α_3 is weakly more important than α_2.

Table 3.2: Pairwise comparison matrix

	$\boldsymbol{\alpha_1}$	$\boldsymbol{\alpha_2}$	$\boldsymbol{\alpha_3}$
$\boldsymbol{\alpha_1}$	α_{11}	α_{12}	α_{13}
$\boldsymbol{\alpha_2}$	α_{21}	α_{22}	α_{23}
$\boldsymbol{\alpha_3}$	α_{31}	α_{32}	α_{33}

Table 3.3: Pairwise comparison matrix (filled in)

	$\boldsymbol{\alpha_1}$	$\boldsymbol{\alpha_2}$	$\boldsymbol{\alpha_3}$
$\boldsymbol{\alpha_1}$	1	7	3
$\boldsymbol{\alpha_2}$	1/7	1	1/3
$\boldsymbol{\alpha_3}$	1/3	3	1

Then we derive from Table 3.3 the normalized pairwise comparison matrix (Table 3.4), where we normalize the values of every column with:

$$\bar{\alpha}_{ij} = \frac{\alpha_{ij}}{\sum_{i=1}^{m} \alpha_{ij}} \quad ,\forall\, j$$

Table 3.4: Normalized pairwise comparison matrix

	α_1	α_2	α_3
α_1	0.68	0.64	0.69
α_2	0.09	0.09	0.08
α_3	0.23	0.27	0.23

Table 3.5: Weight factors

	Weight vector
w_1	0.6687
w_2	0.0882
w_3	0.2431

Finally, the weight factors of every attribute can be calculated by taking the average per row:

$$w_i = \frac{\sum_{j=1}^{m} \bar{\alpha}_{ij}}{m} \text{ , } \forall\ i$$

It is also important to check whether the pairwise comparisons are consistent or not. When someone strongly prefers A over B, strongly prefers B over C and strongly prefers C over A, their comparisons are not very consistent, and the preferences or the compared attributes need to be reconsidered. To determine the consistency of the pairwise comparisons, we need to calculate the consistency index (CI) and the consistency ratio (CR) (Kahraman, 2008). First, compute the largest Eigen value λ_{max}. We do this by multiplying the sum per column of the pairwise comparison matrix with the weight vector; so, we multiply the sum of column 1 with weight vector 1, multiply the sum of column 2 with weight vector 2 etc. We add all these values and get the approximation of the eigenvalue.

$$\lambda_{max} = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (\alpha_{ij}) * w_j \right) = 3.011$$

Then we can calculate the consistency index (CI) with:

$$CI = \frac{\lambda_{max} - m}{m - 1} = \frac{3.011 - 3}{3 - 2} = 0.0054$$

When the pairwise comparison is completely consistent CI = 0, but small inconsistencies are tolerated. We check whether CI is low enough by calculating the consistency ratio (CR). CR compares CI to the random consistency index (RI). Table 3.6 shows the values of RI at different amounts of attributes. The values of RI are gained by calculating CI when the entries of the pairwise comparison matrix are completely random. Our pairwise comparisons are considered consistent when CR < 0.1.

$$CR = \frac{CI}{RI} = \frac{0.0054}{0.58} = 0.0093 < 0.1$$

Table 3.6: Random consistency index

m	2	3	4	5	6	7	8	9	10
RI	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.51

3.6 Conclusion

In this chapter we look at literature to help us classify the beamline of MCX and find routing strategies that previously have been researched in the past. The beamline can be classified as a hybrid flow shop (HFS), which is a flow shop where jobs

follow stages chronologically, stages can have multiple machines and jobs are able to skip stages. Most research related to the HFS deal with production scheduling, where all jobs and machines are available at time zero. Making real time routing decisions require vastly different approaches, since decision making time is crucial in a system where reducing flow time is important. Three different real-time routing strategies are reviewed.

With Threshold-based Alternate Routing (TAR), a threshold value is established for the system. Jobs follow a certain route through the system, which is their main route. Once the benefits of choosing an alternative route surpass the threshold value, the job will switch to the alternative route, instead of its main route.

The swarm based real time routing of Meyyapan et al. (2007) also uses a threshold value on machines, but this value changes dynamically with the status of its machine. On top of this, jobs have a certain stimulus that is based on their waiting time and priority for example. Once the stimulus of a job is higher than the threshold value of a machine, the machine will accept the job. This model already brings more responsiveness to changes in the status of the system.

Weng, Wei, & Fujimura (2012) researched the effect of combining routing strategy and dispatching rules. They found promising result, using a strategy that considers time flexibility and demand changes, with a dispatching rule that chose the Shortest-Processing-Time (SPT) first. The objective of their research was the minimization of both the earliness and tardiness of jobs.

We also researched what other objectives are commonly used in HFS problems. Minimizing the make span is by far the most commonly used objective. Other objectives are minimizing maximum tardiness, idle time and total flow shop. Most researches only use a single criterium objective, while in practice it is more common that multiple criteria are considered. When a Just-In-Time system is considered, minimizing both the earliness and tardiness of jobs is an interesting objective.

When multiple criteria are important in a research and these criteria are independent and unrelated to each other, it might be hard to express how much very criteria is values. That is why we looked at some multi-criteria decision-making processes, especially multi attributed decision making. The AHP approach seems the most interesting for this research, since it offers a clear way of comparing multiple attributes, by making pairwise comparison between every attribute. The weights are scales between zero and one, which makes it useful in objective functions, when multiple attributes are weighted and added.

4 Solution design

In this research we focus on the behavior of the beamline, specifically the product routing. However, there are more areas that influence the behavior of the beamline, Section 4.1 describes these other areas and how we incorporate them in this research. Section 4.2 describes how we model the product routing, including the modelling of the beamline, the routing itself and the routing of beams that are still part of a nesting. When it is clear how we model the routing, Section 4.3 explains how we choose the optimal route for the jobs and nestings. Finally, Section 4.4 gives a summarizing conclusion of the chapter.

4.1 Other solution areas

In Chapter 1 we determined that we focus on the product routing in order to get more control on the behavior of the beamline. However, with regard to the planning and logistics of the beamline, there are two more areas that influence the performance of the beamline: the nesting of beams and production scheduling.

4.1.1 Nesting

How and which jobs are nested together in the same beam has a big influence on which operations need to be performed, how much work is required per beam and how many jobs are on the beamline at the same time. So, to get to an optimal production process, the nesting of jobs cannot be ignored. However, for this research we leave the optimization of nestings out of scope and use a simple heuristic that minimizes scrap per nesting, by adding as many beams together in a nesting (see Appendix E).

The nesting of jobs also has a direct impact on the product routing. Jobs in a nesting are part of the same physical beam, so they travel together until they reach a divisor (either a saw or plasma robot), based on where the sawing operation needs to be performed. The physical path a nesting takes to the divisor we call the nesting route. Every single job in a nesting has its own operations that need to be performed. The route a nesting takes needs to provide the opportunity to perform all operations, either before or after the divisor. This also holds for the required outfeed location of a job. For example, outfeed location 1 can only be reached via saw 1 and outfeed location 2 can only be reached via saw 2. A job that needs to leave at outfeed location 1 and a job that needs to leave at outfeed location 2 cannot be nested together. We assume that these requirements are taken into account by the party that is responsible for the nesting process. It is a common process in the steel industry and normally it is done by a work preparator.

So, from a nesting perspective, the input for the product routing is a list of nestings that arrive at the beamline. For a nesting, the following characteristics are given:

- The infeed location of the nesting
- The required operations of all individual jobs
- The required outfeed location of all individual jobs
- Per saw cut, the type of cut that needs to be made, either with the saw or the plasma cutter
- The dimensions of the nesting

We assume that the length of every nesting is longer than the minimum length requirement of all transport systems necessary to reach a divisor.

4.1.2 Production scheduling

Production scheduling determines the sequence in which jobs enter the beamline. This has a direct impact on the number and type of jobs that are located on the beamline. The customers that use a beamline often process steel beams for big construction projects. Recall that these projects are divided in phases depending on the construction order and the maximum weight or dimensions that can be transported at the same time. Steel manufacturers often know when a phase is needed on the construction site, so they know when it needs to be ready. To make sure that a phase is ready in time, the jobs in the phase are processed relatively early. This is done because the expected make span of a phase is often uncertain and there is enough storage space for finished products.

The production planning of individual jobs/nestings within a phase is often based on how beams are stacked by the supplier. The operator usually just picks the beam that is easiest to grab and loads it on the beamline. This is usually the beam that lies on top of a stack. We know there are general rules where trucks are loaded with the heaviest and longest beams on the bottom. But we do not know if the line is loaded from the trucks, or if beams are restacked in the storage, and we do not know how the operator decides what beam to pick next. So, from a scheduling perspective this can be described as a random procedure. This will also be the starting scenario of our research: we start using a random sequence of jobs per phase.

In Chapter 3 we reviewed literature about dispatching rules. These rules are used to determine the sequence in which jobs leave a buffer to a next stage. We cannot use this for the buffers within the beamline, since they only work with the FIFO principle. However, we can use these rules to determine the sequence in which products enter the beamline. We know that jobs arrive at the beamline in phases, so a phase can be seen as a set of jobs that arrive simultaneously at the beamline. We can sort the jobs/nestings in these phases with dispatching rules before they enter the beamline.

The dispatching rule sorts the nestings before they enter the system, so before the nestings are influenced by the product routing. The product routing relies heavily on the number of jobs, the processing time in the system and the required operations of the jobs. With the dispatching rules we can regulate the sequence that beams enter the system, so also the amount of work that enters the system. We expect that the product routing performs most effectively when the amount of processing times in the system is evenly spread over all machines. To test this, we look at 5 different dispatching rules. We use two different dispatching rules from previous studies (MWRF and LWRF) and add three new ones (BWSF, LWBF and MWIMF). We expect that spreading out the work over all machines or choosing jobs that add the least work to the current bottleneck machine has a positive impact on the output of the system.

Dispatching rules:

0. Random (RAND) selects a random job. The random number is chosen with a uniform distribution.
1. Least-Work-Remaining-First (LWRF) selects the job with the least total work remaining. The total work is equal to the processing time that is required to finish all operations of the job.
2. Most-Work-Remaining-First (MWRF) selects the job with the most total work remaining.
3. Least-Work-for-Bottleneck-First (LWBF) selects the job that adds the least processing time to the current bottleneck machine. With this dispatching rule we try to relieve the bottleneck machine from work and first send jobs that have little work for the bottleneck machine.
4. Most-Work-for-Idle-Machine-First (MWIMF) selects the job that adds the most processing time to the machine that currently has the least processing time in the system. With this rule, we focus the machine that has the least amount of work, to make sure that all machines are able to work at all times.
5. Best-Work-Spread-First (BWSF) selects the job that divides the work the best over all machines in the system. The goal of this rule is to provide every machine with the same amount of processing time.

We do not use any dispatching rules that include the due dates. This is because the current steel processing market does not use due dates for individual jobs, but per phase. We do not have proper data available about jobs per phase and the due dates of these phases, so testing dispatching rules based on due dates would require a lot more assumptions and likely result in unreliable testing results. When more data becomes available about due dates, the following dispatching rules would be interesting to test:

- Minimum-Slack-Time (MST) selects the job with the lowest slack value, where slack is the due date substracted by the current time and the processing time of the job.
- Slack/Remaining-Processing-Time (SRPT) takes the slack value of a job, just like the MST rule, and divides it by the processing time it still has in the remaining stages, then selects the job with the lowest SRPT value.
- Modified-Due-Date (MDD) selects the job with the earliest due date or, when the job is finished after the due date, the expected completion time.
- Critical-Ratio (CR) compares jobs by dividing the remaining time until the due date of the job by the processing times of all stages that the job still needs to visit. The job with the mimimum value is chosen.

4.2 Modelling of the routing solution

Now that the assumptions of the other solution areas are clear, we focus on how we model the routing on the beamline. Because a beamline can come in many different configurations and does not fit a standard HFS from the literature, Section 4.2.1 explains how we model the beamline. After that Section 4.2.2 explains how we model the routing, which gives us all possible routes through the beamline. Finally, Section 4.2.3 explains how we determine what routes a viable per job and how this set of viable routes is updated as a job traverses through the system.

4.2.1 Modelling the beamline

Every beamline that MCX delivers is based on the production of the customer and the space limitations of the destined location, so it is very rare that two beamlines have the exact same layout. One of the goals in Chapter 1 is to simplify the configuration and maintenance of the routing strategy, so we need a routing solution that works for every beamline layout. Before we can find a generalized solution, we need to generalize the way we look at a beamline. Both Weng et al. (2012) and Meyyappan et al. (2007) model a hybrid flow shop (HFS) as a series of stages where jobs move from stage to stage, where a stage consists of one or multiple machine. We think this is a useful way to model a HFS when the machines are clearly lined up and jobs can travel from any machine in stage 1 to any machine in stage 2. That way the main function of the product routing is to pick the best machines per stage to process a job. The disadvantage is that the transportation from stage to stage very simplified or left out of scope. How a job reaches a machine in stage 2 from stage 1 and via what route is not considered.

In a beamline, the transport and buffer sections also play an important role when we pick the best route for a job. A route can exist for a large part of RCs and CTs and only a few machines. RCs and CTs determine most of the restrictions of a route with regards to dimensions of a beam and RCs have the most interaction with other parts of the beamline. So, we want to model the beamline in a way that both the machines and the transport between the machines are considered in the routing rules. We do this by modelling every component of the beamline as a node (Figure 4.1). As a result we get the following types of nodes: Infeed, Outfeed, RC, CT and Machine nodes. Every node has a different behaviour and influences the movement and the throughput time of a job differently. Modelling every part of the beamline as an individual node also makes it easy to build different beamline configurations from scratch, by simply adding and connecting nodes to a model, which is important when we design routing rules that are independent on the beamline configuration.

Machine nodes:

The machine nodes are the locations where operations can be performed on the jobs. So we define per machine node which operation(s) can be performed and the costs of performing the operation(s). In this research we make use of relative processing costs instead of the actual costs. The relative costs are determined as follows: we take the operation "Drilling" for example, the drill can drill most efficiently, so we give it a relative cost of 1. The other machine that can perform the operation "Drilling" is the plasma cutter, however the plasma cutter uses more energy and requires extra labor. Including these factors, we estimate, together with the engineers of MCX, that it is three times more expensive to make a hole compared to the drill, so it gets relative costs of 3. Besides a difference in costs, the duration of the same operation might also differ between machines. This is excluded from the costs, but we take it into account when we determine the throughput time in Section 4.3.

The ability to perform a certain operation as well as the relative costs for the operation can be summarized in one table. Table 4.1 shows the relative costs per machine of the beamline in Figure 4.1. When a machine is not capable of perfoming an operation, it does not have a relative cost for that operation.

Machines that are capable of dividing a beam in two (saws and plasma cutters) can act as outfeed point of the beamline. This is required for jobs that are separated from the nesting but are shorter than the minimum length that is required to traverse the rest of the route. These jobs need to be removed at the divisor, either manually or automatic. When a machine is capable of dividing a beam, the machine node is also labeled as a divisor and the type of cut it makes is specified. A cut made with a sawblade we call a saw-cut and one made with a plasma cutter we call a cope-cut. This difference is needed, because in most situations the plasma cutter is visited after the saw. When a product that is shorter than the minimum required handling length requires a cope, it needs to be cut by the plasma cutter, otherwise it is removed at the saw and will never reach the plasma robot.

Table 4.1: Machine Capabilities

	Shot Blasting	Marking	Drilling	Sawing	Coping
Shot Blaster	1	-	-	-	-
Marking Machine	-	1	-	-	-
Drill	-	1	1	-	-
Saw 1	-	-	-	1	-
Saw 2	-	-	-	1	-
Plasma Cutter	-	5	3	5	1

Roller Conveyor (RC) nodes

The RC nodes are responsible for the transportation of beams. We divide the RC nodes in three different types:

- Infeed RC, is located in front of a machine and feeds beams to the machine.
- Outfeed RC, is located after a machine and takes beams from the machine.
- Transfer RC, is not located adjacent to a machine is and dedicated to transporting beams.

Section 2.4 already explains the interaction between the RC and machine; when an in-/outfeed is blocked and when a next beam can be loaded onto them. Another important aspect of the RC is the amount of predecessors and successors they have, since RC nodes can be a merge, a split or even both.

When a RC node is defined as a split, everytime a beam enters the node, we determine the best route for the beam (Section 4.3 explains how we do this). Then, the beam is transported to the correct successor. When a RC is defined as a merge, everytime the RC is ready to receive a next beam, a priority rule is used to determine from which predecessor the next beam is taken (Section 4.3 also explains the priority rules of merges).

The combination of infeed RC – machine – outfeed RC could be seen as one big 'node' with the combined attributes of the three individual parts, which would reduce the amount of nodes and possibly simplify the model. However, because of the amount of interaction and configurations of the RCs we choose to keep these nodes separately. This way, the possibilty of a split or merge is exclusively linked to RC nodes and performing operations is linked to machine nodes. It also creates a clear distinction between different nodes, which results in a more modular solution and makes it easier and clearer to configure a new model from scratch. Every node can be modeled as it is in real life, with its own functions and attributes.

Furthermore, we need to know the following attributes per RC node:

- Minimum and maximum allowed length of beams
- The speed at which the RC can move the beams

Cross Transport (CT) nodes

CTs function as buffers within the beamline. We model them as FIFO buffers, since beams cannot pass each other on the CT. To take into account that beams need to be moved from one side of the buffer to the other, beams have a minimum waiting time inside the buffer, based on the length of the buffer. Per CT node, the following information needs to be stored:

- Jobs located in the buffer and the sequence in which they entered the buffer
- Minimum and maximum allowed length of beams
- Capacity of the CT in meters, based on the length of the buffer

Infeed nodes

An infeed node represents the point from where the jobs are loaded onto the beamline and always acts as the starting point of a possible route through the beamline. It can be seen as a (small) buffer in front of the beamline where the operator picks the beams from and loads them onto the first cross transport.

Outfeed nodes:

The outfeed nodes function as exit points of the beamline, so once a job arrives at an outfeed node they leave the beamline.

By creating connections between the nodes, we create a directed graph. From this directed graph we can derive all possible paths that lead from a start point to an end point and because we know all nodes on paths and the attributes of each node, we know all attributes per path. In Section 4.2.2 we use this to create all possible routes through a beamline.

4.2.2 Modelling the routing

First it is important to know how we define a route. Our definition of a route consists of two parts: The physical path a job takes, and decision of which machine performs which operation.

1. ***Physical path:*** All the nodes a job visits from the start point of the route until the endpoint of the route.
   - A route always starts at an infeed node. This is the point where a job enters the beamline

- The route ends at a node where it is possible for a job to leave the system. This is either at an outfeed node or a machine node that is labeled as a divisor. We need to add routes that stop at the divisor because it is possible that individual jobs do not have the minimum required length that is needed to continue after the divisor.

2. ***Machine assignment:*** Defines which operation is performed at which machine-node.
   - A job is only allowed to follow a route when all required operations of the job can be fulfilled on the route.
   - When an operation is assigned to a machine, the assumption in this research is that all work of that operation type is performed at that machine (for example: it is not possible to drill a portion of the required holes with the drill and make the rest with the plasma cutter). Exception to this are the cuts that need to be made on a nesting. For every job within a nesting it is decided whether its cuts need to be made at the saw or the plasma cutter.

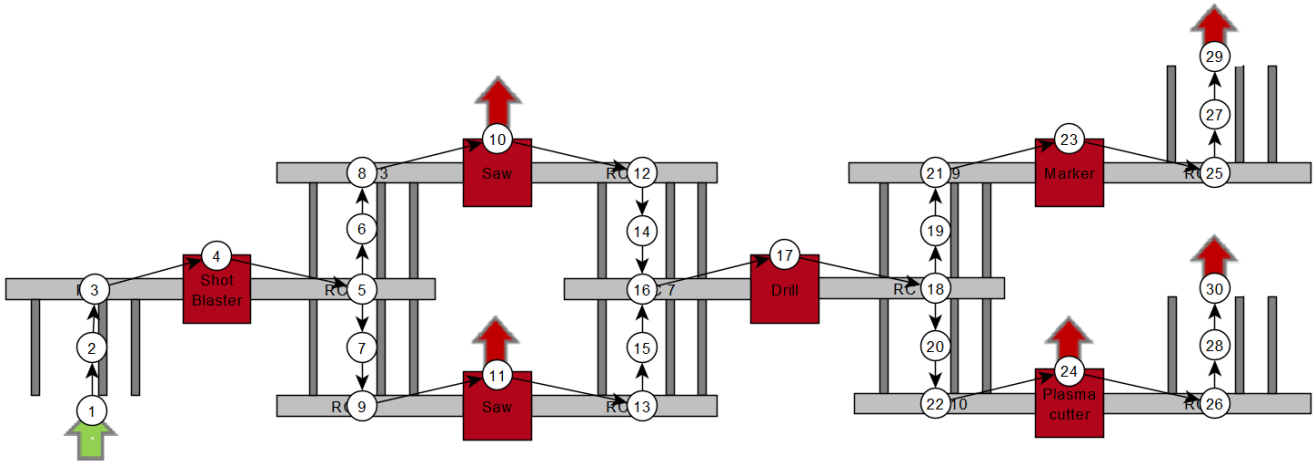


Figure 4.1: Beamline as graph

Using this definition of a route, we can determine all possible routes on a beamline. When we take the beamline of Figure 4.1 as an example, we are able to generate 38 different routes. To give an example of how the routes are defined, Table 4.2 (on page 52) shows half of the 38 routes (we only consider the routes via the top saw, the other routes are the same but then via the bottom saw). The columns of the table are as follows:

Column 1 (Route): Indicates the route number.

Column 2 (Infeed): Indicates the infeed location of the route. A route can only be started from an infeed node.

Column 3 (Outfeed): Indicates the Outfeed location of the route. The outfeed location of the route can be an outfeed node or a "divisor" node (a saw or plasma cutter). Products leave at a divisor node when they are too short to travel any further.

Column 4 (Nesting Route): Before a job reaches the saw, it is part of a nesting. The nesting route is the physical path the nesting takes to the first divisor it encounters on the route. This helps in determining which routes are viable per job, since every job in a nesting needs to have the same nesting route. Section 4.3.4 explains this concept.

Column 5 (Physical path): The physical path consists of all the nodes a job will visit. Multiple routes can have the same physical path, because operations might be performed on different machines. A Physical path always starts at an infeed and ends at an exit point, which can be either an outfeed node or a divisor. In the case of the beamline in Figure 4.1 a route

can end at the saws, at the plasma cutter or at one of the two outfeeds. In Table 4.2 we only look at the routes via the top saw, so the physical paths are: infeed – saw, infeed – plasma cutter, infeed – outfeed 1 and infeed – outfeed 2.

Columns 6 - 10 (Operations): Indicates which machine will perform which operation. An empty cell means that the operation cannot be performed on that route, so when a job requires that operation it cannot follow the route. It can also be that a job does not have any holes that need to be drilled, but the route has the drill assigned to drill the holes. This is no problem. When the job arrives at the drill in that case, it will just continue through the machine, without being processed.

Column 11 (Divisor): Indicates the type of divisor. This can be either 'saw' when the cuts are made by the saw, or 'cope' when the cuts are made by the plasma cutter

Column 12 (OutType): Indicates at what type of node the outfeed is located. It can be either outfeed, saw or cope, as a route can end at an outfeed node, saw or plasma cutter respectively.

Column 13 (Min Nesting Length): Indicates the minimum required length to reach the divisor.

Column 14 (Min Job Length): Indicates the minimum required length from the divisor to the outfeed. When the divisor functions as outfeed this column is empty.

Column 15 (Max Length): Indicates the maximum possible length to follow the entire route.

Table 4.2: Route examples

Route	IN	OUT	Nesting Route	Physical path	Sawing	Drilling	Marking	Shot-Blasting	Coping	Divisor	OutType	Min Nesting Length	Min Job Length	Max Length
1	In 1	Plasma Cutter	1	1	Saw 1	Drill	Drill	Shot Blaster	Plasma Cutter	Saw	Cope	2.5	1.6	25
2	In 1	Plasma Cutter	1	1	Saw 1	Drill	Plasma Cutter	Shot Blaster	Plasma Cutter	Saw	Cope	2.5	1.6	25
3	In 1	Plasma Cutter	1	1	Saw 1	Plasma Cutter	Drill	Shot Blaster	Plasma Cutter	Saw	Cope	2.5	1.6	25
4	In 1	Plasma Cutter	1	1	Saw 1	Plasma Cutter	Plasma Cutter	Shot Blaster	Plasma Cutter	Saw	Cope	2.5	1.6	25
5	In 1	Plasma Cutter	1	1	Plasma Cutter	Drill	Drill	Shot Blaster	Plasma Cutter	Cope	Cope	2.5		25
6	In 1	Plasma Cutter	1	1	Plasma Cutter	Drill	Plasma Cutter	Shot Blaster	Plasma Cutter	Cope	Cope	2.5		25
7	In 1	Plasma Cutter	1	1	Plasma Cutter	Plasma Cutter	Drill	Shot Blaster	Plasma Cutter	Cope	Cope	2.5		25
8	In 1	Plasma Cutter	1	1	Plasma Cutter	Plasma Cutter	Plasma Cutter	Shot Blaster	Plasma Cutter	Cope	Cope	2.5		25
9	In 1	Out 2	1	2	Saw 1	Drill	Drill	Shot Blaster	Plasma Cutter	Saw	Outfeed	2.5	1.6	25
10	In 1	Out 2	1	2	Saw 1	Drill	Plasma Cutter	Shot Blaster	Plasma Cutter	Saw	Outfeed	2.5	1.6	25
11	In 1	Out 2	1	2	Saw 1	Plasma Cutter	Drill	Shot Blaster	Plasma Cutter	Saw	Outfeed	2.5	1.6	25
12	In 1	Out 2	1	2	Saw 1	Plasma Cutter	Plasma Cutter	Shot Blaster	Plasma Cutter	Saw	Outfeed	2.5	1.6	25
13	In 1	Out 2	1	2	Plasma Cutter	Drill	Drill	Shot Blaster	Plasma Cutter	Cope	Outfeed	2.5	1.6	25
14	In 1	Out 2	1	2	Plasma Cutter	Drill	Plasma Cutter	Shot Blaster	Plasma Cutter	Cope	Outfeed	2.5	1.6	25
15	In 1	Out 2	1	2	Plasma Cutter	Plasma Cutter	Drill	Shot Blaster	Plasma Cutter	Cope	Outfeed	2.5	1.6	25
16	In 1	Out 2	1	2	Plasma Cutter	Plasma Cutter	Plasma Cutter	Shot Blaster	Plasma Cutter	Cope	Outfeed	2.5	1.6	25
17	In 1	Saw 1	1	3	Saw 1			Shot Blaster		Saw	Saw	2.5		25
18	In 1	Out 1	1	4	Saw 1	Drill	Drill	Shot Blaster		Saw	Outfeed	2.5	1.6	25
19	In 1	Out 1	1	4	Saw 1	Drill	Marker	Shot Blaster		Saw	Outfeed	2.5	1.6	25

4.2.3 Update viable routes

A job can only follow a route when all requirements of the job can be fulfilled. A route that fulfills all requirements, we call a viable route. A viable route needs to fulfill the following requirements of a job:

- The route must start at the infeed location of the job or the corresponding nesting
- The route must lead to the outfeed location of the job
- All required operations of the job need to be processed by a machine
- The length of the job needs to lie between the minimum and maximum length restrictions of the route

In reality it is possible to perform some of the operations manually, after a job leaves the beamline. However, to create a fully automated process we assume that all operations need to be performed on the beamline.

Once a job has entered the system its route can still be changed; the route can only be changed at nodes that are defined as a split node. Split nodes are nodes that have multiple successors. The only type of node that can be defined as split nodes are roller conveyors (RCs).

As a job moves through the beamline, the number of viable routes becomes smaller. There are a number of times the set of viable routes gets updated.

1. After a split, a job has been sent in a certain direction. All routes that go in a different direction at that split can be removed.
2. After a job has gone through a machine, it is possible that the job has been processed by the machine. For example, a job leaves the marking machine which just processed all marking work on that job. In this case, all routes that offer a different machine for the marking operation can be removed from the viable routes.
   The same principle is true when a job was not processed by the marking machine. In that case, all routes that assigned the marking machine for the marking operation can be removed from the viable routes of that job.

4.3 Selecting the best route for a job

Now that we determined the set of routes a job can follow, we explain how we select the best route from the set. One of the goals in Chapter 1 was to give more overview to the customer. The perspectives of the stakeholders in Chapter 2 also showed that it is desirable to give the customer control over the output of the beamline, by letting the customer choose the focus of the routing, based on a couple of preset attributes. Section 4.3.2 explains the attributes that are used and how they are put into practice. The attributes are put into an objective function in Section 4.3.3. But first, Section 4.3.1 explains how the total processing time of a job is determined, since this is an important measurement in the calculations of Section 4.3.2.

4.3.1 Determining the total processing time of a job

The processing time of a job plays an important role in two of the three attributes that we use to score a route in Section 4.3.2. So, it is important to understand how the processing time of a job is determined.

- **Processing time in the machine**: This includes the time that is required to complete all actions in the machine, e.g. measuring the beam, clamping it in place and performing the required operations.
- **Travel time inside the machine**: When a machine is located on the route of a job, it needs to travel through it, also when the beam is not getting processed by the machine. This is why we separate this from the processing time of the machine. This travel time is relatively short compared to the processing time, but cannot be ignored. When a job only travels though the machine, the machine is still occupied during that duration. When many

products in a row all just travel through the machine and we do not count this travel time towards the processing time of the machine, it seems like there is no work for the machine, while in reality the machine is busy for several minutes just to move beams.
- **Travel time on the infeed RC**: We also include the time a beam needs to travel on the infeed RC, because the second beam in the buffer has to wait from the moment the first beam is loaded onto the infeed RC until it leaves the machine. The travel time on the infeed RC dependents on three factors: the length of the beam, the roller speed of the RC and the distance the beam needs to travel on the RC for the front of the beam to reach the machine.

To determine the total processing time of a nesting, we take the sum of the machine processing times of all jobs in the nesting and the travel time based on the length of the nesting.

The determined total processing times are used in the calculations of the best route of a beam. These calculations are updated as the beam travels through the beamline, so we need to store the processing times. We do this in three ways:

- **Processing Times per Beam**: The processing times are stored per beam. They are stored on a beam level, because the travel times are based on the length of the total beam. So, as long as a job is part of the nesting, its processing time is equal to that of the nesting. Once it gets separated from the nesting its processing time is based on the individual job.
- **Total Time in System**: Once a beam enters the beamline, its processing times per machine are stored. When the route of the beam is changed or when an operation is performed, the processing times are updated. With this we use a similar approach as Weng, Wei, & Fujimura (2012) did with their “feedforward of demand change”-rule. This way we know how much work is expected per machine in the beamline, which can be used to see how work is distributed over the different machines and maybe predict upcoming waiting times.
- **Processing Times per Node**: We also store the processing time per node. When a beam enters a node, its processing times for every machine are stored in that node. This way we know how the expected work per machine is divided over the nodes. This is used to determine how much work is in front and behind a certain job. For example, when we want to calculate the expected waiting time of a job, we are interested in the processing times in the nodes that are still to come on the route of the job.

4.3.2 Scoring of the routes

To select the best route for a job, we give a score to every feasible route of a job. We score the routes on three attributes: the throughput time of a job, processing costs of a job and the utilization of each machine in the beamline. These attributes are based on the KPIs from Chapter 2. We choose these three attributes because we can determine them per individual job, and they show three different types of characteristics of both the job and the beamline. Table 4.3 shows how we translate the attributes into data we can calculate. Appendix D gives a more elaborate explanation of how we calculate the attributes.

Attribute	Data used to calculate a score for a route	Indicated as
Throughput time of a job	Expected throughput time of a job. Per route, we calculate the expected time until the job reaches its outfeed location. We use the assumptions that a lower throughput time of a job is always better than a longer throughput time.	α_1
Processing costs of a job	Expected relative processing costs of a job. On a route, every operation is assigned to a certain machine. With a table like Table 4.1 we determine the relative costs of performing every operation with the assigned machines. Finally, for this attribute, we sum the relative costs of all machines. We prefer low relative processing costs.	α_2
Machine utilization of the system	Expected work spread over all machines. * When the best route for a job needs to be determined, there is already a certain amount of processing time per machine in the system. Depending on the route, the job adds processing time to each relevant machine. With this attribute we determine how skewed the processing times are distributed over the machines, by calculating the variance over the processing times of all machines. A low variance means that every machine has roughly the same amount of processing time in the system. We assume that a low variance reduces the chance of having one big bottleneck, while the rest of the machines have no work, and think it has a positive effect on the total throughput of a beamline.	α_3

Table 4.3: Required data for route scoring

***Comment**
We use the variance in order to find the route that spreads the work the best over the machines. This is sufficient because the processing time of an operation does not change a lot when it is performed on different machines. In this comment we also describe a scenario where the processing time of an operation does change a lot depending on the machine it is performed on. We also suggest a solution what to do in this case. Table 4.4 shows two scenarios. ***Scenario 1:*** *the marking machine and the drill can process a beam in 300 seconds, while the plasma cutter takes 1000 seconds to process the same beam (*Figure 4.3) Selecting the plasma cutter would lower the variance in the system the most compared to the other machines, however it would also add a lot more to the total processing time in the system. This is negative for the output of the system, since it adds extra processing time, without adding extra output. So, in this scenario we also need to include the impact on the mean processing time in the system. The impact is more noticeably when the amount of processing time in the system is relatively low and the amount that is added is relatively high. Using a method that includes both the percentage difference of the mean and the variance (or standard deviation) would solve this. With a weight factor (γ), the emphasis can be shifted more towards the difference in mean processing

time for example, since the variance reacts more dramatically than the average to value changes. The weight factor can be used to counteract the stronger reaction of the variance.

$$Machine\ Utilization\ value = \gamma * (\%\ diff.Mean) + (1 - \gamma) * (\%\ diff.Variance)$$

Scenario 2: *All three machine can process a beam in 300 seconds (*Figure 4.2*)*

Here, all machines add the same amount to the mean processing time in the system, so our machine selection can be purely based on how the variance impacts the variance between processing times.

Table 4.4: Machine utilization in different scenarios

Begin state of the system			**Scenario 1** Job processed by:				**Scenario 2** Job processed by:		
			Marking machine	Drill	Plasma cutter		Marking machine	Drill	Plasma cutter
	Proc. time		Proc. time	Proc. time	Proc. time		Proc. time	Proc. time	Proc. time
Marking machine	2000		2300	2000	2000		2300	2000	2000
Drill	1000		1000	1300	1000		1000	1300	1000
Plasma cutter	500		500	500	1500		500	500	800
Mean	1166.7		1266.7	1266.7	1500		1266.7	1266.7	1266.7
% diff	-		+ 9%	+ 9%	+29%		+ 9%	+ 9%	+ 9%
Var	388889		575556	375556	166667		575556	375556	275556
% diff	-		+48%	-3%	-57%		+48%	-3%	-29%
St Dev	623.6		758.7	612.8	408.2		758.7	612.8	524.9
% diff	-		+22%	-2%	-35%		+22%	-2%	-16%

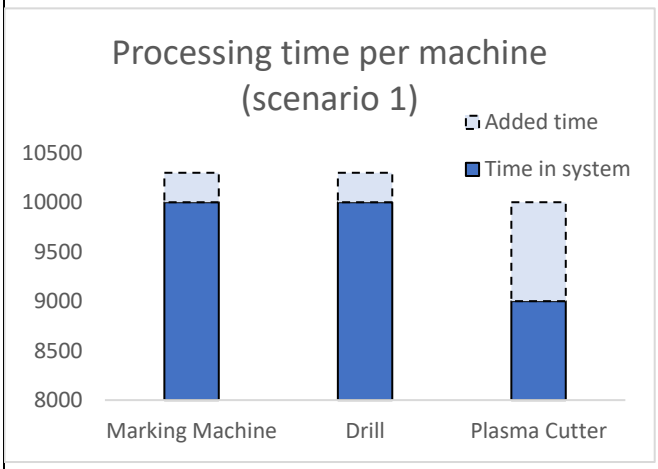


Figure 4.3: Machine utilization scenario 1

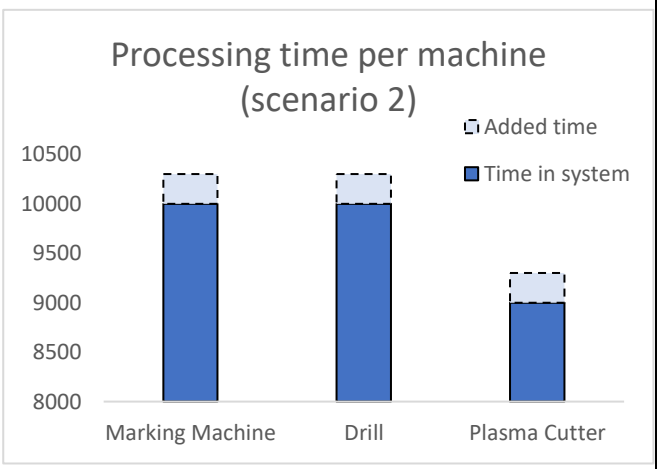


Figure 4.2:: Machine utilization scenario 2

When the α values are calculated for all feasible routes of job j, the α values are in different measuring units and on very different scales. α_1 is measured in throughput time, α_2 in relative costs and α_3 in variance of the processing time in the system. To use all of these units in the same function to compare the different routes, we express all α on a same scale, which we call a score. The attributes need to be scaled in a way it clearly expresses how well a route scores on α_1, α_2 and α_3. We do this by using normalization. Assume job j has R feasible routes it can follow. For every route (r = 1, 2, …, R), calculate α_{1r}, α_{2r} and α_{3r}. After that we normalize the values to have a score between 0 and 1, using (1). We want to minimize all three attributes, so the α of the best route scores 0 and the worst α scores 1.

Table 4.5 shows an example where a job has 14 feasible jobs. These jobs are scored on all three attributes and finally gets a total score, which is explained in Section 4.3.3.

$$Score_{1r} = \frac{\alpha_{1r} - \alpha_{1max}}{\alpha_{1max} - \alpha_{1min}} \qquad (1),$$

Where

$$\alpha_{1_{max}} = max(\alpha_{11}, \alpha_{12}, \dots, \alpha_{1R})$$

$$\alpha_{1_{\min}} = \min(\alpha_{11}, \alpha_{12}, \dots, \alpha_{1R})$$

Another way to scale the score would be to standardize the values instead of normalization. The difference is that the values will be scored on a normal scale, with a mean of 0 and a standard deviation of 1. The advantage of using standardization is that it handles outliers better than normalization. With normalization an outlier would cramp together the “normal” values more than standardization, since it does not scale the values on a set range and instead looks at the deviation from the mean value. We looked at different sets of routes and scaled them in both ways. In the end, both methods do not influence how well a value is ranked, the worst value gets the worst score and the best value gets the best score. In the next section we add weights to each score, but this also shows no difference between using standardization or normalization. We choose to use normalization, because this shows every attribute on a same scale from 0 to 1. When every attribute is given a weight, like we do in the next section, the sum of the score is also scaled between 0 and 1, which gives a clear overview of the ranking of the scores.

Table 4.5: Scoring Attributes

Route	Throughput time α_1	Score$_1$	Costs α_2	Score$_2$	Work Variation α_3	Score$_3$	Total Score
17	1:50:04.19	0.00	4	0.00	2,066,359	0.95	0,68
18	1:53:29.48	0.29	4	0.00	2,103,271	1.00	0,78
45	1:53:44.68	0.31	4	0.00	2,062,573	0.94	0,74
48	1:58:18.68	0.69	6	0.33	1,698,857	0.41	0,46
49	2:02:01.47	1.00	6	0.33	1,730,021	0.46	0,55
50	2:01:43.97	0.98	10	1.00	1,415,789	0.00	0,27
70	1:57:09.97	0.59	4	0.00	2,099,486	0.99	0,83
81	1:53:44.68	0.31	4	0.00	2,066,359	0.95	0,74
82	1:57:09.97	0.59	4	0.00	2,103,271	1.00	0,84
83	1:58:17.47	0.69	8	0.67	1,754,550	0.49	0,54
84	1:58:18.68	0.69	6	0.33	1,702,643	0.42	0,46
85	2:02:01.47	1.00	6	0.33	1,733,807	0.46	0,56
86	2:01:43.97	0.98	10	1.00	1,419,574	0.01	0,28

There is a downside to scaling our values between 0 and 1, it over-exaggerates the difference between the scores of two routes when the differences are very small. For example: we need to choose between two routes, route 1 has a throughput time of one hour, while route 2 has a throughput time of one hour and one second. There is only a minor difference between the routes, however, route 1 gets the best score of 0 and route 2 get the worst score of 1. Standardization does also not solve this, since it would exaggerate the values the same way, just project them on the standard normal scale instead of a 0 to 1 scale.

One way to treat this problem is to scale all values that show a minor difference to the best value to zero. This way, we would make no difference between the best scoring route and routes that score within a certain percentage from the best route. The disadvantage of this is when the choice is made between two routes, where one route is slightly better on all attributes. We want to choose the better route, but with the proposed method, both routes are seen as equal, which is not desirable.

In the search of a way to solve our scaling problem, we return to the literature and find the concept of global scaling (Monat, 2009). Right now, we scale the values of an attribute between the worst and best score, which is called local scaling. With global scaling, we would first determine an overall best and worst possible value for an attribute (based on historic data, aspiration or imagination) and remap these to 0 and 1 on a global scale. When the maximum and minimum of the global scale are established, can be score based on this scale. To get useful results, it is essential to set the right boundaries of the global scale and there are many options to determine these boundaries. The most logical method to set the boundaries is to use an aspirational global scale, which sets the scale based on the best and worst values that could realistically be achieved, within the available means or capabilities. Table 4.6 shows how we can achieve the worst and best value per attribute, given that we stay within our means. Because our means include the feasible routes a job can follow and the current occupation of the beamline, the worst and best values actually match the boundaries of the local scale. In our opinion it would not make sense to use global scaling methods that do not consider the available means and create boundaries that surpass the values that are actually achievable (like imagined global scaling or universal global scaling). This means that global scaling is not a clear benefit over local scaling for us.

We cannot seem to find an optimal way to solve the over exaggeration of the minor differences, so accept it when it occurs. We want to state that this is not per se a major problem, since it only occurs when all feasible routes score relatively close to each other on an attribute.

Table 4.6: Aspirational global scale

Attribute	Worst achievable value	Best achievable value
Throughput time	The route with the longest throughput time	The route with the shortest throughput time
Costs	Perform every operation with the most expensive machine option	Perform every operation with the cheapest machine option
Machine Utilization	Perform every operation that increases the variance in the beamline the most.	Perform every operation that increases the variance in the beamline the least.

4.3.3 Objective function

Not all customers view every attribute as important. When a customer has a lot of overcapacity, they might want to focus on producing against minimal costs since a maximum machine utilization is not as important, whereas a customer with a

very busy beamline wants to focus on machine utilization or a fast throughput time. To deal with this, we use weights that increase or decrease the impact an attribute has on the total score of a route.

For a customer to pick a weight for every attribute, without any reference point, can be a hard task. So, to help in this process, we use the analytical hierarchy process (AHP). Asking the customer to make pair wise comparisons between every attribute according to the AHP comparison scale from 1 to 9 makes it a lot easier for the customer to translate their attribute preferences into weights. Table 4.8 shows an example of the translation of pair wise comparisons into weight factors. w_1, w_2 and w_3 indicate the weight factor for attribute 1, attribute 2 and attribute 3 respectively.

Table 4.7: AHP value interpretation

Value of α_{ij}	Interpretation
1	Objective i and j have equal importance
3	Objective i is weakly more important than objective j
5	Experience and judgment indicate that objective i is strongly more important than objective j
7	Objective i is very strongly or demonstrably more important than objective j
9	Objective i is absolutely more important than objective j
2, 4, 6, 8	Intermediate values

Table 4.8: AHP Example

	Throughput time	Costs	Machine Utilization
Throughput time	1	3	1/5
Costs	1/3	1	1/7
Machine Utilization	5	7	1

	Weight factor
w_1	19.3%
w_2	8.3%
w_3	72.4%

When the weights are determined, we multiply the score with the corresponding weight for every attribute ($i = 1, 2, 3$) and get.

$$\sum_i Score_i * w_i$$

Using the example weights of Table 4.8, we can determine the total score of every route in

Table 4.5. We want to minimize the score, so when we would rank the routes from lowest to highest score, we find that route 50 scores the best in this scenario and will be chosen for this job.

4.3.4 Routing of a nesting

Before jobs visit a divisor, either a saw or a plasma cutter, jobs are part of a nesting. This means that all jobs travel along the same route until the first divisor, after that jobs might be split from the nesting and continue as individual jobs or remain joined in a nesting and continue to another divisor together. In Section 4.1 we already explained that we only nest jobs together that have the same infeed location and can complete all of their required operations on at least one route.

So, until the nesting has reached the divisor, its route needs to be optimized for all jobs in the nesting. Different jobs in the nesting might have a different optimal route. In this case, a choice needs to be made on which path is chosen. The operations that are performed on the jobs however, still need to be determined per job. There might be two jobs in a nesting, where job 1 will visit a plasma cutter after the saw and the job 2 follows another route. When the nesting visits a drill, job 1 will have the option to make its holes with the drill or at the plasma cutter, where job 2 has no choice and needs to have the holes drilled. Figure 4.4 illustrates this principle.

By design, there will never be multiple saws on the same route and per route there will also be maximum of two machines where the cuts can be made: at the saw or at the plasma cutter. We call the physical path that a nesting takes to the first divisor it encounters on its route, the nesting route. We use the first divisor on the route, because this is the part of the route that every job in the nesting is guaranteed to travel together. When jobs are still joined in a nesting after the first divisor, it is guaranteed that they all will go to the second divisor together. Since there is only one second divisor in the route, it is not necessary to check if all jobs follow the same nesting route to the second divisor.

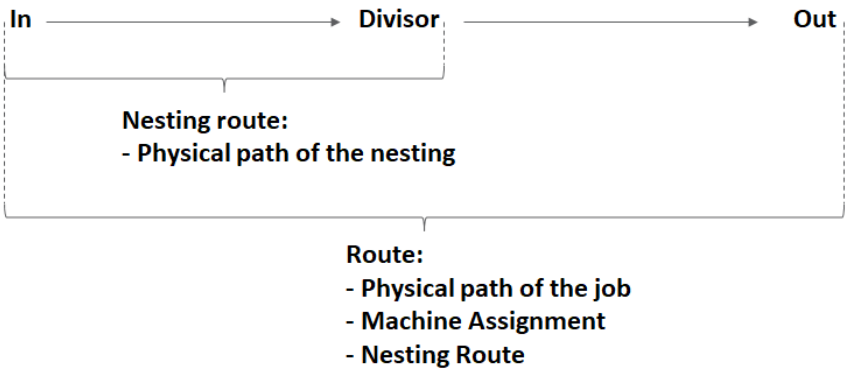


Figure 4.4: Route with out on an outfeed node

To give an example of the process to find the best nesting route, we look at three jobs that are nested together. First, we determine all viable routes per job and check the corresponding nesting routes (Table 4.9). Then we check which nesting routes are viable for every job in the nesting (in the example these are nesting routes 2 and 3) and remove the routes that do not follow this nesting route (

Table 4.10). The remaining routes get a score and the best route per job is selected. Let us color the best route per job green in this case and the best route of any different nesting route light green (Table 4.11). If all jobs have the same nesting route as best option, we pick it as the nesting route for the nesting. When jobs have different best nesting routes, we add the scores per job for every nesting route and the nesting route with the best total score is considered the best nesting route (Table 4.12).

By calculating the best nesting route this way, we consider all nesting routes instead of just the one that is the best for most jobs for example. It is possible one nesting route is good for 5 jobs in a nesting but very bad for the other 3, while another nesting route might be slightly worse for the 5 jobs, but a lot better for the other 3. Our way of choosing the best nesting route tries to find a route that is on average best for all jobs.

For the jobs that had a best route with a different nesting route to the optimal nesting route, we determine a new best route. We select this route from a set of viable routes that follow the chosen nesting route.

Table 4.9: Determining best nesting route (1)

Job 1		**Job 2**		**Job 3**	
Route	Nesting Route	Route	Nesting Route	Route	Nesting Route
1	1	3	2	1	1
3	2	5	3	2	2
4	2	6	3	6	3
5	3	9	5	7	4

Table 4.10: Determining best nesting route (2)

Job 1		**Job 2**		**Job 3**	
Route	Nesting Route	Route	Nesting Route	Route	Nesting Route
3	2	3	2	2	2
4	2	5	3	6	3
5	3	6	3		

Table 4.11: Determining best nesting route (3)

Job 1			**Job 2**			**Job 3**		
Route	Nesting Route	Score	Route	Nesting Route	Score	Route	Nesting Route	Score
3	2	1.1	3	2	2	2	2	2
4	2	1.2	5	3	1.2	6	3	1
5	3	1.5	6	3	1			

Table 4.12: Determining best nesting route (4)

Nesting Route	**Job 1**	**Job 2**	**Job 3**	**Total Score**
2	1.1	2	2	5.1
3	1.5	1	1	3.5

Priority at merges

At a merge, two or more paths come together. This can be a merge of two CTs onto a RC (Figure 4.5), a merge of a CT and a RC/machine (Figure 4.6) or a combination of the two, where two CTs and a RC merge. To determine which side of the merge has priority, we look to minimize the chance of a standstill of the nodes. A standstill happens when a machine is blocked when it should be processing a beam. In Chapter 2, we explained that a machine can only be loaded with a new beam, when its outfeed RC is free. As soon as the outfeed RC is occupied when a new beam can be loaded into the machine, the machine is standing still unnecessary and production is lost.

So, the way to prevent a standstill is to make sure that an outfeed RC is always able to transfer its beam to a successor. This means that it is preferable that there is always storage space available in every CT. For a merge of two CTs onto a RC, this means that priority is given to the CT that has less free space remaining. For a merge of a CT and a RC, the RC has always priority, since the RC has the biggest chance of holding up a machine. For a merge of two CTs and a RC, the RC also has priority, then the CT with the least free capacity.

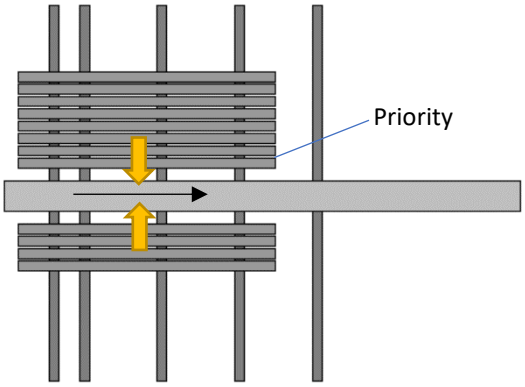


Figure 4.5: Merge of two CTs

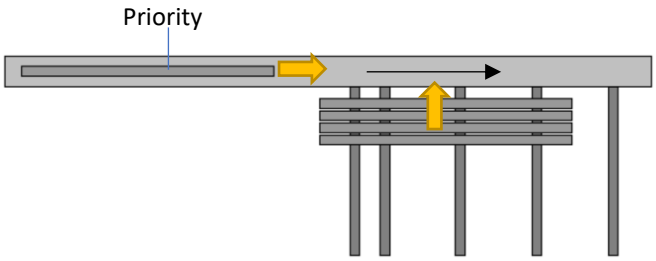


Figure 4.6: Merge of RC and CT

This solution has the advantage that it becomes easier to predict when a beam will leave the CT. When there is no supply of new beams and we take the example in Figure 4.5, the upper CT will be drained first, until the free capacity is equal to the other CT. From that moment, a beam is taken from both CTs alternately. This means that, when a beam enters the CT, it needs to wait at least the amount of time it takes to empty both buffers.

We choose to base the merge priority on free capacity (in meters) over methods that look at the amount of processing time in the buffers, because a lack of buffer space is what causes a standstill. Processing times do not represent the physical contents very well, a buffer might be completely filled with jobs that require very little work, while the other buffer might contain a few beams that require a lot of work.

When the free capacity on both sides of the merge is close to equal, it might be interesting to look at the processing time of the first beam of both buffers. It might be preferable to choose the beam with the least amount of processing time, to increase the speed of draining the buffers, or to choose the beam with the longest processing time when the route after the merge is very busy and needs some time to deplete. These considerations are not taken into account in our merges but could be part of another research.

4.4 Conclusion

Besides product routing, we considered two other areas that influence the behavior of the beamline: nesting and production planning. To stay within the boundaries of our scope, we exclude the optimization of the nestings and leave that for another research project. We do consider the production planning, to determine whether it can support the routing rules we design. We expect that the routing rules are heavily influenced by the distribution of the work that enters the beamline. We assume that the routing rules can be the most effective when the work is spread evenly over all machines, so we use different dispatching heuristics to test this.

In Section 4.2 we showed how we designed our solution. First, we model the beamline as a network of nodes. Every part of the beamline is a defined as a type of node with certain attributes. The nodes are divided into RC nodes, CT nodes, machine nodes, infeed nodes and outfeed nodes. In machine nodes certain operations can be performed against a relative cost, RC nodes are responsible for transportation, CT nodes function as buffers within the beamline and in- and outfeed nodes function as entry and exit points respectively. By connecting the nodes, like the different parts would be connected in reality, we can derive all the possible routes through the beamline. Since we know all the attributes of the individual nodes, we also know the attributes of every route, like the possible operations that can be performed on a route and the minimum and maximum required beam length.

When a beam enters the beamline or arrives at a split node the best route is (re)calculated. Every route is scored on three different attributes: The expected throughput time, the expected relative processing costs and how well the work is divided over all machines in the beamline. Because we want to give some control to the user of the routing rules, the user can apply weight factors to each attribute. These weight factors are gained by using a analytical hierarchy process, where the user compares every attribute pairwise on how much more important one is over the other. Within a nesting, it can occur that not every job has the same best nesting route. To decide with nesting route is taken, we first determine all routes that are candidates to be the best route. Then, we determine the total score of each candidate route, by adding the score of every product on the concerned route and select the candidate that has the best total score.

Merges within the beamline have their own priority rules. When two CTs merge, the CT with the least amount of free capacity gets priority. When a CT merges with a RC or a machine, the RC or machine always gets the priority, since the chance on a standstill is the greatest when a beam lies longer than necessary on a RC or in a machine.

5 Test design

In this chapter we design and build a simulation model to test the routing and dispatching rules we designed in Chapter 4. To test our solutions, we create a simulation model in Tecnomatix Plant Simulation, a program made by Siemens. We use Plant Simulation because it offers good software to create and perform a discrete even simulation. With continuous simulation, every movement is simulated, where discrete event simulation only simulates events that change in the state of the model and skips the time in between these events, which saves a lot of time in the simulation process. In our model, some examples of discrete events are the arrival of new beams at the infeed node, the movement of a beam from one node to another or finishing the processing of a beam in a machine node. Another advantage of Plant Simulation is our familiarity with the software, so our learning curve is shorter, and we can focus on testing our solution instead of learning the software.

Besides choosing the software, it is important to understand why we use simulation and what we want to achieve with our model. With the simulation model, we try to simulate the behavior of the beamline, so that we can test our routing rules in the model instead of a real beamline. This allows us to keep track of the processing statistics and analyze how our routing rules function in different scenarios. Section 5.1 describes the model we use and Section 5.2 shows how we validate this model. Section 5.3 describes the different configuration of input variables we want to test. Section 5.4 explains the experiment design, including the number of replications and run length of the experiments that we perform. Finally, Section 5.5 summarizes all sections of this chapter.

5.1 The model

Plant Simulation uses object-oriented hierarchical modeling (Siemens, 2019), which allows the user to use pre-defined objects to build a model. These objects are arranged into different libraries, based on the function of the object. This section uses general terms to describe the conceptual model where our simulation model is based on.

Three things are important with the model: all nodes need to function like they do in real life and the interaction between nodes need to be match real life, the model needs to get the right infeed of products and the model needs to keep track of the data we need to judge our solutions. Section 5.1.1 explains how we generate the jobs and nestings that are fed into the beamline. In Section 5.1.2 we explain how we model the nodes, Section 5.1.3 describes the interaction between nodes and Section 5.1.4 explains how we combine the nodes and create different beamline configurations. The model also uses a lot of tables to keep track of input, process and output statistics. We summarizes these tables in appendix G and reference to these tables when required for clarification.

5.1.1 Job generation

We recall the distinction between the different states a job can occur in and what we need to know about it to simulate it in Table 5.1.

Table 5.1: Beam nomenclature

State	Description	Required input
Job	A single product, which requires some operations to become a finished product.	- Dimensions (length, width, height) - Profile - Profile Code - Phase - # of holes and their dimensions - Total marking length - # of copes - Weight
Nesting	A beam before it has visited a divisor. A nesting can contain a single job or multiple jobs. In general, beams enter the system as nestings, because they always need to visit a divisor. We need to know what jobs the nesting contains and if these jobs need to be separated at the plasma cutter or not. (This is explained later in this section)	- Maximum length (24 meters for I-Profiles, 18 meters for U-Profiles, 12 meters for M-Profiles) - The jobs it contains - The type of cuts in the nesting
Beam	When we refer to a beam, we refer to the physical beam in general. So, a beam can still contain a nesting or be a single job. This has no impact on the simulation but makes it clear when we refer to something as a beam.	N/A
Batch	Certain machines can handle multiple beams at the same time. These beams as a group is called a batch.	- Maximum width of a batch - Maximum height difference between beams in the batch

To generate the jobs, we use a product database that is provided by MCX. This product database is based on a year's worth of production of one of MCX customers and contains a lot of project variety, which is expected at users of the beamline. We use this database because is a complete overview of typical products for MCX customer in Europe. It contains a lot of different type of buildings and it covers a long time spam. It includes the dimensions per job, required operations and corresponding project numbers.

Every project contains a certain amount of jobs. These jobs are not produced all at once, but are often produced per phase. From our database we cannot derive which job belongs to which phase, so we divide every project into phases ourselves (as shown in Figure 5.1). For this research, we assume that a phase contains a set of jobs with a cumulative weight of maximum 25 tons. We add jobs to a phase until it would surpass 25 tons, then we assign the next jobs to the next phase etc. We use 25 tons because this is the maximum weight of the trucks that transport the beams. So, we assume that every phase is a delivery of a full truck load.

We divide every project into phases and this results in a product database that contains 655 phases in total. From now on, we do not look at the jobs per project anymore but consider the jobs per phase. We do it this way, because we feed our model with phases instead of projects. In real life it is possible that multiple projects are being processed simultaneously, which means that it is not necessary that the phases of one project need to be finished before the phases of the next project can start.

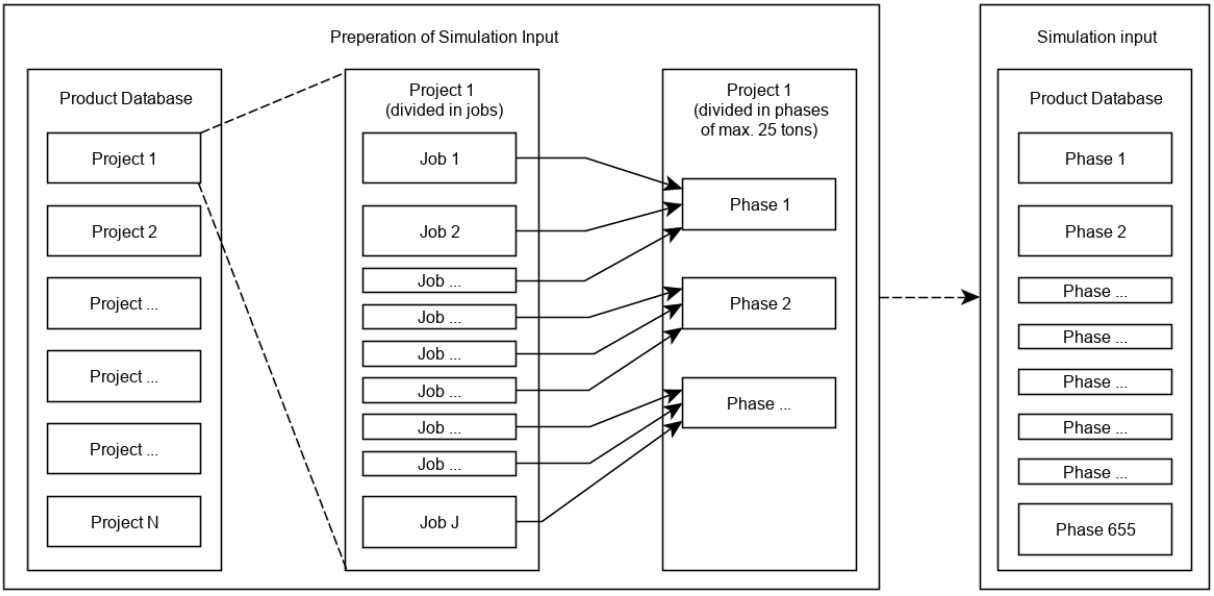


Figure 5.1: Preparation of Simulation Input

Now that we divided our product database into phases, we can use it to feed our model. However, these phases still consist of jobs, while the input of the beamline consists of nestings. When the beamline triggers the generation of a new phase, a (random) phase will be picked from the database. Then the jobs in this phase are put into nestings (Figure 5.2). Our assumption is that jobs can only be nested with jobs from the same phase. We do this by looping through the jobs of a phase. For the first job, we always create a new nesting and add the job to that nesting. For the rest of the jobs, we loop through all nesting of the phase and add it as soon as we find a match. When a job does not fit into any nesting, we create a new one. A job fits into a nesting when 1) it has the same profile 2) the length of the nesting does not exceed the maximum length when we add the job. The maximum length of a nesting depends on the profile type and equals 24, 18 and 12 meters for I-profiles, U-profiles and M-profiles respectively. The way we nest our jobs comes close to what happens in reality, where the goal is often to minimize waste, which is achieved by cutting as many jobs from one nesting. A more elaborated example of the nesting process can be found in Appendix D. After the nestings are created, they are stored in

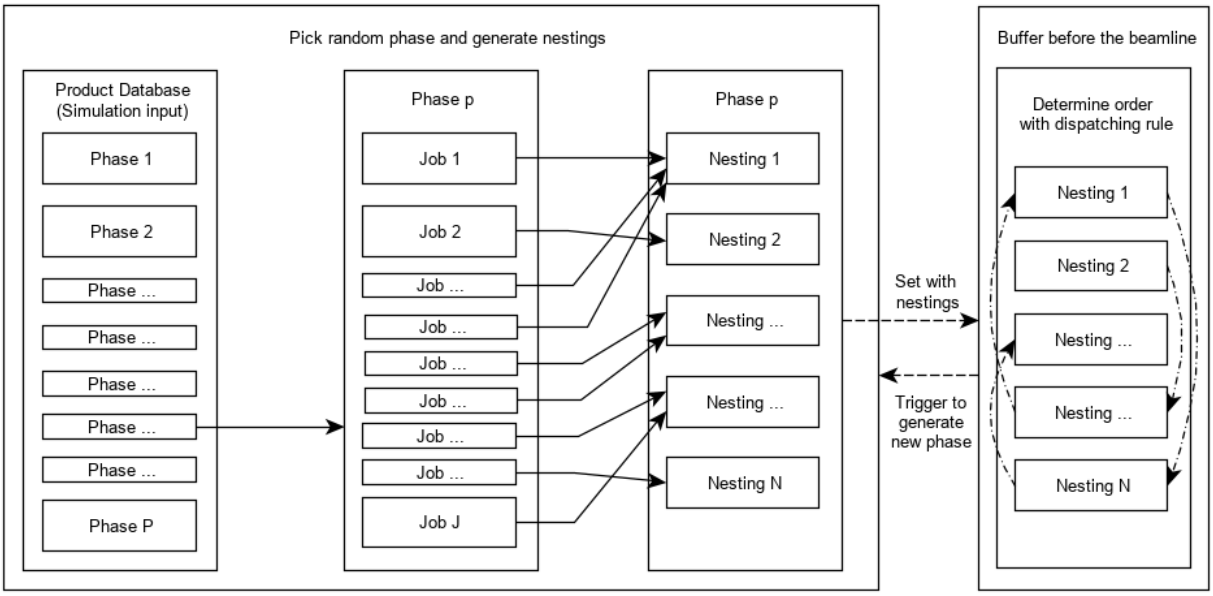


Figure 5.2: Generation of Simulation Input

a buffer in front of the corresponding infeed of the nesting. Every infeed has its own table. Here they are sorted according to the dispatching rule.

Before the nestings enter the beamline, we determine which cut needs to be made at what kind of divisor, a saw or a plasma cutter. We try to perform as many cuts as possible with saw(s). Sawing is the only operation that has a set machine that cannot be changed. The main reason for this is that cuts that are created by a saw or a plasma cutter have different widths. This width is incorporated in the rest of the design of the beam, the placement of the holes etc. When the holes are already drilled and need to be 10 millimeter from the beginning of the beam, this difference in cut width between the saw and plasma cutter makes a big difference.

Another reason is that some beams require not just a straight cut, but a curved cut or a cut that consists of multiple angles. These "special cuts" can only be performed by the plasma cutter. However, this also means that the products that are connected to these special product also need to be cut off at the plasma cutter. We call cuts that need to be performed at the plasma cutter "Cope" cuts. Cut that are set as "Cope" cannot be changed to a saw cut. Take the nesting in Figure 5.3 for example; job 1 has normal cuts and can be cut off from the nesting at the saw. After that, the cut between job 2 and 3 is a special cut, so it cannot be cut off at the saw and we set it as "Cope". We assume that the cut between job 3 and 4 also is a regular cut again and can be performed at the saw and the cut between job 4 and 5 is a special cut and need to be performed at the plasma cutter again. This way, we assign every cut at a certain type of machine. Recall that every beam also has a required cut at the beginning and end of the beam to remove excess material.

Figure 5.3: Nesting

5.1.2 Nodes

Chapter 4 explained the function of all types of nodes and what information is required per node. In this section we explain how we translate this into a conceptual model. Figure 5.4 gives an idea how the modeled nodes look like in our model.

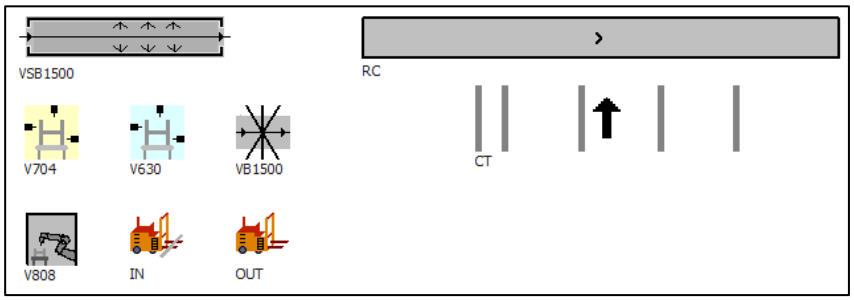


Figure 5.4: Simulation nodes

Before we discuss every node individually, we explain some function that are applicable for every node. All nodes are connected to at least one other node. Whenever a beam enters a node, we update the location of the beam, we update the destination where the beam is supposed to go next and we keep track of the expected processing times per machine that are located in the node. When a beam leaves a node these expected processing times are updated again.

RC Node

To model a RC, we use a conveyor object, which moves beams like a conveyor belt. In reality the speed of the RC is dependent on the weight of the beam(s) it transports, with a maximum speed of 1 m/s, when the beams are very light, and a minimum of 0.3 m/s for very heavy beams. In the model we set the *Speed* of the *Line* at 0.6 m/s as a steady speed, to simplify the model. Setting the speed at a set amount does impact the travel time of the beams, however it does not impact the behavior of the product routing, which we want to test and compare. The choice of 0.6 m/s (slightly below the average between 0.3 m/s and 1 m/s) is the result of discussions with the engineers at MCX.

The dimensions of the RC are set according to the blueprints of the beamline and we add two attributes to the node to indicate the maximum and minimum length that is required for a beam to travel over the RC. When the RC has multiple predecessors or successors, we indicate it as merge or split respectively. When a beam enters a RC that is indicated as split, the best route for the beam is determined. When a RC with a merge is able to receive a new beam a merge rule is applied to choose from which predecessor the next beam will be taken.

We also indicate the function of a RC: as infeed of a machine, outfeed of a machine or as transfer. This info is relevant for the interaction between the RCs and the machine. We describe this interaction in Section 5.1.3.

CT Node

We model a CT node as a queue. Arriving beams join the end of the queue and the beam at the front of the queue is the first one to leave. This results in a First-In-First-Out buffer, which corresponds with how a CT function in real life. In the model we keep track of the beams that are in a CT and in what order they arrived.

Just as with the RC nodes, we indicate the maximum and minimum length requirements of the beam. We also keep track of the free capacity of the CT. At the beginning of the simulation the beamline is empty and this attribute is set to the maximum capacity, which is determined with the blueprints of the beamline.

In reality, the CT is not a static queue, where beams are immediately available when they enter the CT. When a beam enters an empty CT, it needs to be shoved to the other end of the CT, which takes approximately 30 seconds. We simplify this in the model by forcing beams to stay least 30 seconds in the buffer. This mainly impacts the buffer time of beams that enter an empty buffer. When a beam enters a non-empty buffer, their waiting time is already longer than 30 seconds most of the time.

Machine Node

The drill, marking machine, saw and plasma cutter can only process one beam at the time. A beam needs to be fully processed and moved away before the next beam can enter the machine. This is why we model them with a *SingleProc* object, which is capable of processing one product at the time. The shot blaster is capable of processing multiple batches of nestings simultaneously, so we model it with a *ParallelProc* object, which is capable of processing multiple products at the same time. Section 5.1.3 further describes how the total process per machine looks like and how it is modelled. Besides the difference of a SingleProc and ParallelProc, all machines get the same attributes. Per machine node we keep track of the CT(s) that is/are preceding the infeed RC of the machine, because these function as buffers directly in front of the machine. Whenever the machine can receive a new beam, the beam is picked from these buffers

We also label machines that are capable of dividing a nesting into beams as a divisor. For the saw, we also indicate whether it contains a short product removal system (SPRS) or not. When the saw does have a SPRS, a product is automatically transported away from the saw and dropped into a collection bin or on the floor. The required time to remove a short product from the saw is set at 30 seconds. Without SPRS, an operator needs to remove short products with a crane, which takes about 120 seconds. During the time that a short product is being removed from a divisor, the divisor is stopped.

Which operation can be performed by which machine against what relative costs is stored in a table like Table 5.2.

Table 5.2: Machine Capabilities

	Shot Blasting	Marking	Drilling	Sawing	Coping
Shot Blaster	1	-	-	-	-
Marking Machine	-	1	-	-	-
Drill	-	1	1	-	-
Saw 1	-	-	-	1	-
Saw 2	-	-	-	1	-
Plasma Cutter	-	5	3	5	1

Every time a beam leaves a machine node, the total processing times in the system are updated as well as the operations that are still required on the beam.

Infeed Node

Just like the CT node, the infeed node is also modelled as a queue. However, its capacity is set as infinite and it does not interact directly with the rest of the nodes. The infeed node functions as queue before the beamline where beams wait until they can be loaded onto the beamline. This means that the nestings in the infeed node do not yet contribute to the total processing time in the system. However, we do keep track of all nestings in the queue together with their expected processing times of the nestings, since this data is used by the dispatching rule to determine the sequence in which the nestings will leave the infeed. Every time a new phase of nestings enters the infeed node, we the nestings according to the given dispatching rule. The current dispatching rule is indicated as a global variable in the simulation model.

Outfeed Node

The outfeed node is modelled as a drain object. A drain simply removes a beam from the model. Before it is removed, we save all relevant statistics of the beam, such as the throughput time and the route it took.

5.1.3 Interaction between nodes

There is a lot of interaction involve between nodes. Nodes often need to wait before they can pass on a beam to their successor and moving a beam from node to node takes time. In this section we will explain the interaction between different nodes and how we model this.

RC → CT → RC

When a beam reaches its end position on a RC node, we check the destination of the beam and check whether it is possible to move the beam there. When the destination is a CT, the beam needs to be shoved from the RC to the CT and place in the right spot. This process of moving the beam takes about 24 seconds (the pushing mechanism needs to move to the

right spot behind the beam, then push the beam from the RC to the right spot on the CT). During the transfer, both nodes cannot interact with any other nodes or beams. As soon as a beam is moved, the nodes can interact again and we check if there are beams waiting to be loaded onto the RC in question. If so, we start the process of moving that beam.

Moving from a CT to a RC also takes about 24 seconds, and involves some extra logic. In reality a CT contains one master leg. The master leg is the leg of the CT that contains the sensor that notices where a beam lies, so a beam always need to lie on the master leg and at least one other leg. In the simulation, we assume the master leg always connects with the RC at twenty percent of the total RC length. Because a beam always needs to touch the master leg, we assume that twenty percent of the beam lies before the master leg. Figure 5.5 shows that the position of a beam that has been moved from the CT to the RC. It is position such that 20 percent of the beam lies before the master leg position.

After a beam has been transferred from a CT, we check if there is a beam waiting to be loaded on the CT.

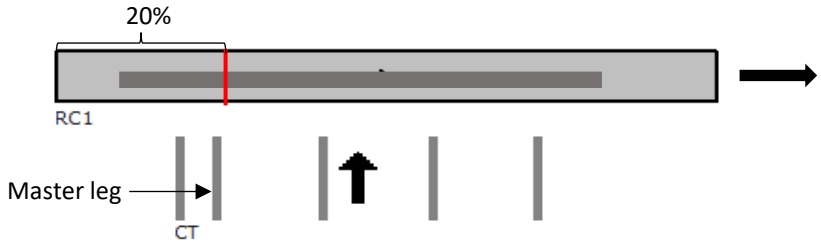


Figure 5.5: RC > CT > RC

RC → Machine → RC

Marking machine and drill

The marking machine and drill are very similar in behavior, so we treat them the same. The interaction between the nodes works as follows:

- A beam can only enter a infeed RC when the infeed RC and succeeding machine are both free.
- A beam can only enter a machine when the machine and the succeeding outfeed RC are both free, otherwise it needs to wait before the machine (Figure 5.6).
- A beam can only transfer from a machine to a outfeed RC when the outfeed RC is free

When a beam reaches the end of the infeed RC, we transfers the beam to the machine node when possible. After the processing time, the beam is transferred to the outfeed RC. In reality, the beam needs to move through the machine, which is why we also include the travel time inside the machine in the processing time. When a beam leaves the machine, we check whether a beam can be loaded onto the infeed RC. It is not possible a beam is occupying the infeed RC, while the machine is occupied.

The processing times of the marking machine and drill are calculated as follows:

$$Processing\ time\ marking\ machine\ = \frac{Beam\ length}{0.6\ m/s} + \frac{Total\ marking\ length}{15\ mm\ /\ s} + 13$$

$$Processing\ time\ drill = \frac{Beam\ length}{0.6\ m/s} + Processing\ time\ to\ drill\ every\ hole^{*} + \#\ different\ diameters * 40 + 17.5$$

* The processing time to drill every hole is calculated, using the data from

Table 2.6

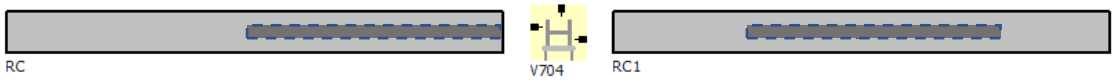


Figure 5.6: RC to Marking machine to RC

Saw

The saw uses the same rules as the marking machine and drill but has some additional rules. When a job is cut off a nesting and part of the nesting is still in the saw, there are two options:

- When the saw is also the outfeed location of the job (when the job is short), the job needs to be removed from the saw before it can continue sawing. This removal time is 30 seconds when the saw has a SPRS and 120 seconds otherwise. So, when the job leaves the beamline via the saw, the machine is stopped during the removal time and continues processing afterwards.
- When the saw is not the outfeed location of the job, it continues to the outfeed RC. The saw is stopped until the job has left the outfeed RC Figure 5.7.

When the last part of a nesting is processed and leaves the machine, a next beam can enter the infeed RC. However, unlike the drill and marking machine, the infeed RC cannot be loaded immediately. The saw works with a measuring truck that pushes a beam through the saw. This measuring truck first needs to move back to its starting position, so that it can push the next beam through the saw. The measuring truck takes 20 seconds to move back, so we check whether a new beam can be loaded onto the infeed RC 20 seconds after a nesting is completely processed by the saw.

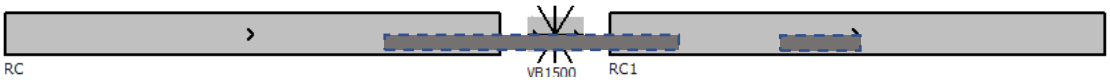


Figure 5.7: RC to Saw to RC

A jobs processing time for the saw depends on the jobs position in the nesting. The time per cut is calculated with the formulas we determined in Chapter 2. However, when we look at the example in Figure 5.8, the processing time of job 5 is a lot more than just the time that is needed for the cuts of job 5. All jobs before it need to be cut and transported away via the outfeed RC or the SPRS. During this transport the saw is stopped, so it is added to time a nesting is occupying the saw.

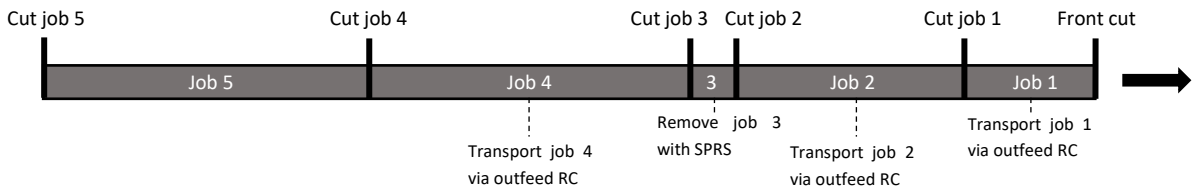


Figure 5.8: Relative processing time in a nesting

$$Processing\ time\ saw\ i^{th}\ job\ in\ the\ nesting$$
$$= cut\ time * front\ cut + \sum_{j=1}^{i} cut\ time\ for\ j^{th}\ job\ in\ the\ nesting$$
$$+ \sum_{j=1}^{i-1} transport\ time\ \ for\ j^{th}\ job\ in\ the\ nesting$$

When a job is removed with the SPRS, the transport time is easy to determine, namely the SPRS time of 30 seconds. However, when the job is transported via the outfeed RC, we determine the transport time the same as we would determine the transport time on a RC. Plus, we add the required time to transfer a beam from a RC to a CT, to make sure a beam is really removed from the RC. The processing time of the whole nesting is equal to the relative processing time of the last job in the nesting.

Shot blaster

The shot blaster processes batches of beams by slowly moving them through the machine, while the beams are blasted with a stream of course material. Inside the shot blaster, the beams move with a speed of 2 meter per minute or 0.03 m/s. The processing time of a batch is based on the length of the batch (to move from position 1 to 2 in Figure 5.9) plus the length of the shot blaster itself, which is 8.3 meter (from position 2 to 3 in Figure 5.9). As soon a batch reaches position 2, the infeed RC is free and a new batch can be loaded onto the infeed RC, so after (Batch length / 2 m/min) we check if there is a batch waiting to be loaded onto the infeed RC. When the batch is placed on the infeed RC, it can start moving towards the shot blaster It can also enter the shot blaster while the previous batch is still (partly) in the shot blaster. This way it is possible that the shot blaster processes multiple batches at the same time.

$$Processing\ time\ Shot\ Blaster\ = \frac{Batch\ length + 8.3}{0.03}$$

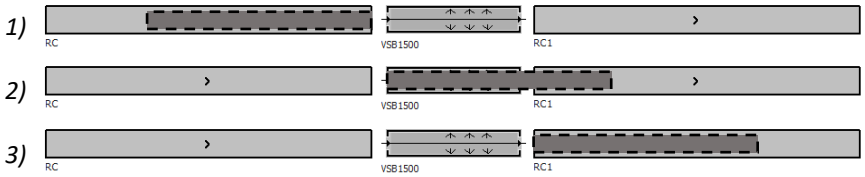


Figure 5.9: Shot Blaster

A batch can only leave the shot blaster when the outfeed RC is free. When a batch reaches the end of the shot blaster, while the outfeed RC is still occupied the shot blaster is stopped, as well as the infeed RC to prevent collisions.

Plasma cutter

The plasma cutter is a combination of the drill/ marking machine and the saw. When it just needs to process a single job, it functions the same as the drill and marking machine: a job arrives at the machine, when it leaves the machine a new job can enter the infeed RC and when the outfeed RC is clear, the new job can enter the machine. The processing time is based on the number of copes and holes need to be cut out, the marking length and the travel time through the machine.

$$Processing\ time\ Plasma\ cutter\ (job)$$
$$= (\#\ of\ copes + \#\ of\ holes) * 35 + \frac{Total\ marking\ length}{15\ mm\ /\ s} + \frac{Product\ length}{0.6}$$

When a nesting enters the plasma cutter and the individual jobs need to be divided, the plasma cutter functions like the saw. The difference is that the processing time per job is extended with the other operation the plasma cutter might need to perform, like cutting some holes or add some markings. Short products are always manually removed, not with an SRPS, so this time is also larger.

$$Relative\ processing\ time\ Plasma\ cutter\ (i^{th}\ job\ in\ the\ nesting) = \sum_{j=1}^{i} (cut\ time\ for\ j^{th}\ job\ in\ the\ nesting + prossecing\ time\ Plasma\ cutter\ (job)) + \sum_{j=1}^{i-1} transport\ time\ \ for\ j^{th}\ job\ in\ the\ nesting$$

5.1.4 The beamlines

We use three different beamline configurations, based on real beamline layouts, to test our solutions. Appendix F shows the layout of the beamlines. We choose these three configurations because they contain the most possible routes. When a beamline contains a lot of different routes, product routing becomes more interesting, because it becomes less clear what the best route is. The layout is also very different per beamline: Beamline 1 and 2 contain the same amount of machines, but have 104 and 67 possible routes respectively, making it interesting whether the number of routes has an impact on our routing rules. Beamline has an extra saw and drill compared to the other two beamlines, increasing the total production capacity of the line. Note that all three beamlines start with a shot blaster. This is a common feature, because it is more efficient to shot blast nestings instead of individual jobs.

We create the beamlines by simply connecting the nodes from Section 5.1.1. Plant Simulation automatically recognizes what the direct predecessor and successor of every node is and adds them as attribute to the node. We use this to automatically derive all possible routes in the beamline. Recall that a route consists of a physical path and the machine assignment on that physical path. So, first we determine all physical paths through the beamline; a physical path contains all nodes from an infeed location to an outfeed location. Then, we determine per physical path which machines are available and what operations they will perform. Let us take an example of a physical path that contains a marking machine and a drill. On this path, there are two possible routes, one where the marking is done on the marking machine and the drilling on the drill, and one where both the marking and the drilling are performed on the drill.

Beamline 1 and 2 contain multiple outfeed location, however, from our product database we do not know to which outfeed location a job needs to go. To simulate this, we first determine what percentage of the jobs needs to go to which outfeed node. This percentage is determined together with experts from MCX. Then, when the jobs are generated they get assigned an outfeed based on these percentages.

Beamline 3 contains multiple infeed nodes. For this we use the same principle as with the multiple outfeed nodes: we assign every job to an infeed node, based on a predetermined percentage.

5.2 Validation of the model

We do not have production data of a real beamline that shows the information like the actual processing times, when jobs enter the beamline, when every operation is being performed or when the jobs leave the beamline. So, we cannot compare the output of our simulation model with the output of a real system. One way to validate our model is validating the

functionality of all individual nodes. When all components do what they must do, we can be confident that the model represents the reality in a proper fashion. We sit together with the system engineers of MCX to check per node type of the beamline whether it does what it should do. The most important thing to check is whether the simplifications we implemented in the simulation do not remove any vital functions of a node, alter the function of the node too much or alter the processing times. The interactions that were explained in Section 5.1 have all been designed with the input of the employees of MCX. Because of their inclusion in the design of the model, many aspects of the model were already validated during the design of the model, so we will only explain the big simplifications per node.

Infeed node: The infeed node itself does not need any validation, it functions as the start node of routes. However, we do need to validate the way and speed that jobs are loaded onto the beamline. In our simulation, we add jobs to the beamline whenever the CT after the infeed node has free capacity. This means that the first CT is always full and jobs are always available. The beamline will never need to wait on jobs. In reality this is often the goal of the beamline users, to always have beams ready to go, to avoid that machine are waiting because there is nothing to process. Of course, it might always happen that beams are loaded too late or not in the right order, but we simplify this by assuming that the operators load the beamline correctly and in time.

Another part of the infeed that needs to be validated are the jobs we feed into our model. We use a product database from MCX, so the jobs themselves are actual jobs that have been processed in the past.

Outfeed node: The outfeed node simulates the end of the beamline, where beams are taken off the beamline to go to following production steps like welding. In our model, we assume that as soon a beams is available at the outfeed node, it is taken off the beamline. This means that there is always an operator at the outfeed location to pick up the beam and that there is always demand for the beams. The customer perspective in chapter 2 showed us that the beamline often

RC: We simplify the RC node by setting the speed at a standard speed of 0.6 m/s. We already discussed this in Section 5.1.2 where we explained how we came to this speed and why.

CT: A big simplification of the CT is the placement of the master leg. We assume that the master leg is always positioned at 20% from the beginning of the RC and a beam is always positioned in a way that 20% of the beam lies in front of the master leg and 80% lies after the master leg. This has an influence on the distance (and time) a beam needs to travel on the RC to reach the end of the RC. We use the same argument as we used with the speed of the RC: this simplification means that the travel times on the RC do not exactly match the times they travel in reality, but it does not affect the behavior of our routing rules. All viable routes of a job are influence by it in the same way, so the routing rules picks between equal routes.

Another simplification is the dwell time of 30 seconds, which determines that a beam needs to stay at least 30 seconds in a buffer, before it can move on. This dwell time simulates the situation where a beam enters an empty buffer and first needs to be transported to the other side of the CT, before it can be transferred to its successor. In reality this time depends on the length of the CT and the weight of the beam. But again, this time does not influence the behavior of our routing rules, only the minimum buffer time of a beam. It also only affects beams that enter an empty CT. Beams that enter an already occupied CT (almost) always have to wait at least 30 seconds, since the transfer time of a beam from the CT to the subsequent RC is already 20 seconds.

Machine nodes: There are two important aspects that need to be validated: the interaction of the machines with the RCs and the processing times of the machines. We calculated the processing times of the beams in a similar fashion as MCX does in their time calculations, so these times are approved by MCX.

For some jobs, operators are required. The shot blaster, marking machine and the drill can process their operations fully autonomously. However, when a saw does not have an automatic short product removal system, short products need to

be removed manually, which requires an operator. The same holds for the plasma cutter; short products need to be removed manually and the operations often require an observer to make sure everything is processed right. For the simulation, we assume that the number of operators is high enough, such that there is no waiting time for operators and operations can always continue.

Working shifts: In the simulation, we let the production run from 6:00 to 22:00. Between 22:00 and 6:00, the line keeps running unmanned to process the beams that are still on the beamline. Because the beamline is a fully automated processing line, this is possible and also happens in reality. The only factor that needs to be taken into account is that the beams that are still on the beamline cannot contain any operations that require an operator.

After discussing the functionality of all individual nodes with the engineers of MCX, we are confident that the nodes act like they would do in reality. Because every node is modelled properly, we assume that we can properly model a complete beamline. We conclude from this section that, despite all of the simplifications, our simulation model does not differ too much from what happens in reality. We can use our simulation model and be confident that it mimics the reality in a proper fashion. With a properly validated model we can attach more value to the results of our simulation.

5.3 Test configurations

Now that we build our model, we explain what we want to test with it. First, it is important to know what we want to achieve with the tests. We want to test the following things:

- We want to test the impact of the dispatching rules on the KPIs.
- The impact of the different routing rules scenarios on the KPIs.
- We want to test whether the impact of the routing rules is the same on a busy and a quiet beamline.
- We want to test how the dynamic routing rules compare to the current routing strategy.

The focus of the routing rules is based on the weights in our objective function, w_1, w_2 and w_3. These weights are based on the pairwise comparison matrix. In reality, the matrix would be filled in by the user of the beamline. For the tests we try to mimic four different scenarios. Table 5.4 shows an overview of the different scenarios. In the scenarios 2, 3 and 4 the customer prefers one attribute over the other. We want to analyze if there is a different result when the customer purely focusses on one attribute (scenario 2b, 3b and 4b) or when they have a strong preference for one attribute, but also have a slight preference of a second attribute (scenario 2a, 3a and 4a). Scenario 1 can be seen as the basic configuration of our routing rules, where the weights do not influence the score of the routes and every attribute is valued equally. We recall the value interpretation of the AHP method in Table 5.3.

Table 5.3: Value interpretation AHP

Value of α_{ij}	Interpretation
1	Objective i and j have equal importance
3	Objective i is weakly more important than objective j
5	Experience and judgment indicate that objective i is strongly more important than objective j
7	Objective i is very strongly or demonstrably more important than objective j
9	Objective i is absolutely more important than objective j
2, 4, 6, 8	Intermediate values

Table 5.4: Scenario descriptions

Scenario	Focus	Pair-wise comparison matrix	
1	Equal focus	Table 5.5	The customer is indifferent of the attributes. All attributes have equal importance.
2a	Throughput time	Table 5.6	The customer's main focus is throughput time, but also prefers machine utilization over costs.
2b	Throughput time	Table 5.7	The customer only focusses on throughput time.
3a	Costs	Table 5.9	The customer's main focus is costs, but also prefers machine utilization over throughput time.
3b	Costs	Table 5.8	The customer only focusses on costs.
4a	Machine Utilization	Table 5.11	The customer's main focus is machine utilization, with a slight focus on costs.
4b	Machine Utilization	Table 5.10	The customer only focusses on machine utilization.

Table 5.5: Scenario 1

	α_1	α_2	α_3
α_1	1	1	1
α_2	1	1	1
α_3	1	1	1

w_1 = 33% **(Throughput Time)**

w_2 = 33% **(Costs)**

w_3 = 34% **(Machine Utilization)**

Table 5.6: Scenario 2a

	α_1	α_2	α_3
α_1	1	7	3
α_2	1/7	1	1/5
α_3	1/3	5	1

w_1 = 67%

w_2 = 9%

w_3 = 24%

Table 5.7: Scenario 2b

	α_1	α_2	α_3
α_1	1	9	9
α_2	1/9	1	1
α_3	1/9	1	1

w_1 = 82%

w_2 = 9%

w_3 = 9%

Table 5.9: Scenario 3a

	α_1	α_2	α_3
α_1	1	1/7	1/3
α_2	7	1	3
α_3	3	1/3	1

w_1 = 9%

w_2 = 67%

w_3 = 24%

Table 5.8: Scenario 3b

	α_1	α_2	α_3
α_1	1	1/9	1
α_2	9	1	9
α_3	1	1/9	1

w_1 = 9%

w_2 = 82%

w_3 = 9%

Table 5.11: Scenario 4a

	α_1	α_2	α_3
α_1	1	1/3	1/7
α_2	3	1	1/3
α_3	7	3	1

w_1 = 9%

w_2 = 24%

w_3 = 67%

Table 5.10: Scenario 4b

	α_1	α_2	α_3
α_1	1	1	1/9
α_2	1	1	1/9
α_3	9	9	1

w_1 = 9%

w_2 = 9%

w_3 = 82%

We want to test the impact of the dispatching rules.

In Chapter 4, we determined 6 dispatching rules: one random dispatching rule, 2 rules from literature and 3 rules of our own. To recall, the dispatching rules are:

0. Random (RAND) selects a random job. The random job is chosen with a uniform distribution.
1. Least-Work-Remaining-First (LWRF) selects the job with the least work remaining.
2. Most-Work-Remaining-First (MWRF) selects the job with the most work remaining.
3. Least-Work-for-Bottleneck-First (LWBF) selects the job with the least work for the current bottleneck.
4. Most-Work-for-Idle-Machine-First (MWIMF) selects the job with the most work for the machine with least work.
5. Best-Work-Spread-First (BWSF) selects the job that divides the work the best over all machines in the system. The goal of this rule is to provide every machine with the same amount of work.

Besides the dispatching rules, we also want to test the 7 different scenarios on 3 different beamlines, both in a busy and quiet state. Running a single experiment takes quite some time, so we want to select a few dispatching rules that we can use to test every scenario with. We do this by testing all dispatching rules with scenario 1 and select the rules that perform the best. Ideally we want to test one off-line rule (dispatching rule 1 or 2), and one on-line dispatching rule (dispatching rule 3, 4 or 5). The off-line rules do not consider the beam that are on the beamline, which makes them easier to use in production planning for example. The online rules do take into account the beams on the beamline, so the loading sequence of the beams can change every time a single job updates it route. This asks for a lot of calculation, but also attention of the operators who load the beams onto the beamline and need to check the updated production planning every time. So, when an on-line rule has only a slight benefit over an off-line rule, it might still be preferred to use an off-line rule to have more clarity in the production planning. So after the effect of the dispatching rules is tested, we combine them with the routing rules and test these simultaneously.

Table 5.12: Experiments dispatching rules

	Scenario	Dispatching rule	Experiment
Testing dispatching rules	1	RAND	1
		LWRF	2
		MWRF	3
		LWBF	4
		MWIMF	5
		BWSF	6

Comment
Recall that most beamlines start with a shot blaster, which has a relative long processing time, but is able to process batches of beams. When we started testing the dispatching rules we discovered that it is important to create large batches in the CT before the shot blaster, so that it can process multiple nestings simultaneously. Otherwise the shot blaster becomes the bottleneck due to its relative long processing time. Since the shot blaster is often located at the beginning of the line, we can prevent it from becoming the bottleneck by making efficient batches in the infeed. To make batches, we first sort the nestings with the dispatching rule. After that, we select the best nesting according to the dispatching rule and try to add as many nesting to it to create a batch, by cycling through all nestings from best to worst. When the batch is full, we select the next best nesting that is not yet in a batch and again try to add as many nestings as possible. We try to add as many beams per batch according to the standards of MCX.

We want to check how different routing rule scenarios impact the output results of the beamline and whether the impact of the routing rules is the same on different beamlines.

In total we have seven different scenarios. We already test scenario 1 when we test our dispatching rules. For the other scenarios, we want to test them with the random dispatching rule to get a benchmark result, generated with a random sequence of jobs, and with the on-line and off-line dispatching rules that showed the most promising results in the previous test. We do this to test the impact the routing rules and test whether the dispatching rules can further increase this impact. Table 5.13 shows all the different experiments. We perform these experiments on all three beamlines. These tests have two goals. We want to know if the different scenarios impact the beamline like we expect. For example, scenario 3a and 3b focus on minimizing the costs, so we expect that the experiment with these scenarios result in the lowest average costs per job. The second goal is that we want to know if one of the scenarios always leads to a higher production output compared to the other scenarios. We expect that scenarios 4a or 4b results in the highest tons per hour. These scenarios focus on spreading the work equally over all machine, so in theory it should reassign beams from to the bottleneck machine towards other machines. This means that the waiting time in front of the bottleneck machine reduces, the throughput time per job decreases and more jobs can be processed with the same total capacity.

Table 5.13: Experiments per beamline

	Scenario	Dispatching rule	Experiment
Testing Scenarios	2a	RAND	7
		Offline rule	8
		Online rule	9
	2b	RAND	10
		Offline rule	11
		Online rule	12
	3a	RAND	13
		Offline rule	14
		Online rule	15
	3b	RAND	16
		Offline rule	17
		Online rule	18
	4a	RAND	19
		Offline rule	20
		Online rule	21
	4b	RAND	22
		Offline rule	23
		Online rule	24

We want to test whether the impact of the routing rules scenarios is the same on a busy and a quiet beamline.

To test whether there is a difference between a busy and a quiet beamline, we perform all experiments again on all three beamlines. The only difference is that we will only add one phase per day to the beamline, instead of adding beams when

possible. This means that the number of beams on the beamline will be very low. We expect that the routing rules will have a smaller impact, because there are less beams in the system that are getting influenced by every routing decision.

We want to test how the dynamic routing rules compare to the standard routing that is used at the moment.

The last test, and maybe the most important one, is to also test the results of the simulation when we apply the static rules that MCX uses right now. When the results of our dynamic rules are worse than the static rules, is it worth it to implement the dynamic rules, just to save time in the configuration time of the rules? We only test the static rules on beamline 1 and 2. This is purely because of time constraints in our research, but when there is a clear difference between the dynamic and static rules on beamline 1 and 2, we expect that a similar difference occurs on beamline 3. Appendix H describes the static rules that MCX uses per beamline. We test the static rules with all our dispatching rules and don't look at the different scenarios, since these are based on the dynamic routing rules.

5.4 Experimental Design

Before we start performing our tests, we need to explain some settings of the experiments we will perform with our simulation model, like the length of one simulation run and the amount of replication we perform per test.

Run Length:

We do not set an end time of our simulation. We want to run our simulation until all products from our product database are processed. The product database contains a year worth of production, but this production was processed on a smaller beamline than we perform our tests with. We also do not know in how many shifts the total production was produced and how long the actual uptime was during the year of production. So, we expect the simulation run to be much shorter than 365 days. Test runs show that beamline 1 and beamline 2 can process the entire product database in around 70 days, while beamline 3 can process it within 50 days. So, we do not have a set run length, but a set number of products we want to produce.

Warm up time:

We do not use a warmup time, we start with an empty system, which is also a common scenario, at the start of a new day, when the line has been emptied during the night. This means we can use all data from our simulation and do not need to ignore data during a warm up time.

Cooling down:

We stop the simulation when all jobs from the product database have been processed. There is no cooling down included.

Number of replications:

In the simulation, we generate jobs from our product database. We do this by picking random phases from the database and feeding them to the beamline. This results in a random sequence of phases. When we look at the results, we know how our routing rules function in this sequence of phases, but we cannot say with confidence that the results will be the same in other scenarios. Instead we want to be able to say with a certain level of confidence (β) that the results will actually be within a range from the results of the simulation we performed. To do this, we perform multiple replications of the same experiment, where we change the sequence of the phases that enter the system per replication. Every extra replication we perform will change the confidence interval of our results. We determine a confidence interval using the mean and standard deviation and the equation (2). We base our confidence interval on a confidence level of $\beta = 95\%$.

$$\left[\bar{X} - t_{n-1,1-\frac{\beta}{2}} * \frac{s}{\sqrt{n}} \, , \, \bar{X} + t_{n-1\,,1-\frac{\beta}{2}} * \frac{s}{\sqrt{n}}\right] \qquad (2)$$

Where,

$\bar{X}$ = the sample mean of all replications

n = the number of replications

β = the confidence level

$t_{n-1,1-\frac{\beta}{2}}$ = the t-value of the students t distribution

s = the standard deviation of the sample

To know when we performed enough replications to say that our confidence interval is small enough, we use the relative error (γ). This relative error indicates how wide our confidence interval is compared to the sample mean. The smaller the interval, the more value it adds to the simulation study, because we can be confident that the results will lie within a small range of values. We use equation (3) to check how many replications we need to perform until the confidence interval is small enough. We use a relative error of 5%.

$$\frac{t_{n-1,1-\frac{\beta}{2}} * \frac{s}{\sqrt{n}}}{\bar{X}} < \gamma' \qquad (3)$$

Where,

(γ'): $\frac{\gamma}{1+\gamma} = 0.048$

We determine the required number of replications with multiple configurations of our input variables, because a certain number of replications might be enough for one configuration, but not for another. We determined the required number of replications for three different configurations, all using different input configurations. Figure 5.10 shows the results of the first configuration (where we use dispatching rule RAND and routing rule scenario 1), while the results of the other configuration can be found in Appendix I. Figure 5.10 shows that with 4 replications the relative error is small enough. The other configurations only required 3 replications to drop below the acceptable relative error of 5%. So, we will use 4 replications per experiment.

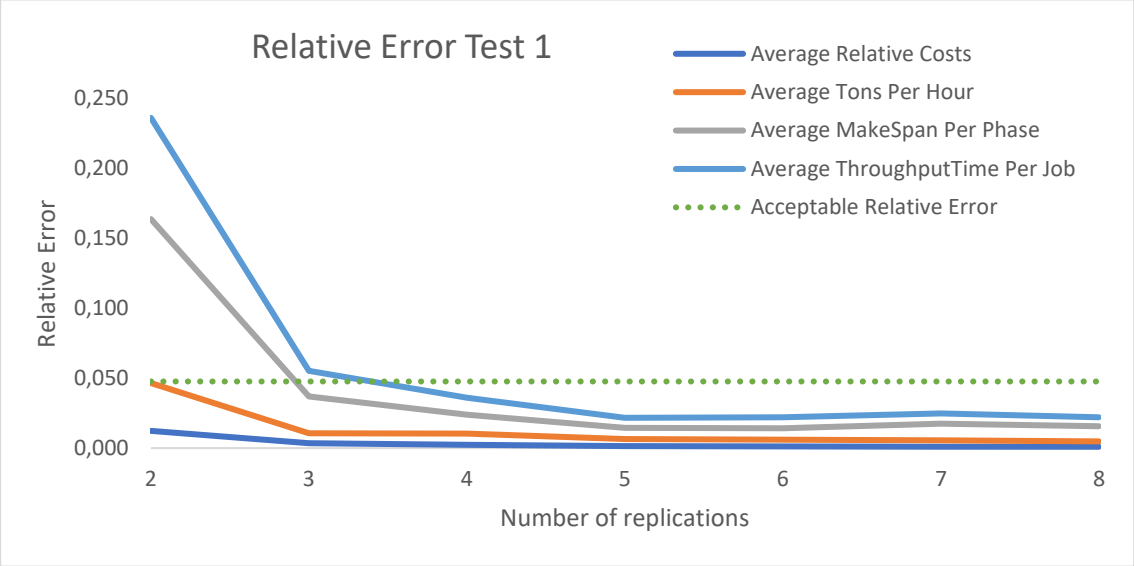


Figure 5.10: Relative error test 1

5.5 Conclusion

To test the routing rules we designed in Chapter 4, we build a simulation model in the Plant Simulation software of Siemens. Just like the nodes we use to model our beamline, Plant Simulation uses preprogrammed, adjustable building blocks that can be used to build a model. So, for all nodes, we fill in the required input parameters we determined in Chapter 4 and adjust the building block so that they match the functionality of our nodes. Then we can connect these building blocks/nodes and create three different beamlines. When the beamlines are built, it is important that the interactions between the nodes follow the correct logic. We programmed this logic in methods, e.g. the logic when to send a next product to a machine or when to determine a new best route for a job. To generate the jobs for our model, we use a product database of MCX. We divide this database into phases and enter these phases in a random sequence into the model. We validated the simulation model by discussing with the engineers of MCX whether the simplification we made to the nodes did not alter the functionality of the nodes too much. Because it is a model to test the behavior of the routing rules and not a model to calculate the expected throughput times with a hundred percent accuracy, the behavior of the beamline resembles the reality well enough.

With our simulation, we want to test four different things: 1) the impact of the dispatching rules, 2) how different entries of the pairwise comparison matrix impact the output results of the beamline and whether the impact of the routing rules is the same on different beamlines, 3) whether the impact of the routing rules is the same on a busy and a quiet beamline and 4) how the dynamic routing rules compare to the standard routing that is used at the moment.

Finally, we set the parameters of our experiments. We do not have a set run length, but let the simulation run until all jobs from the product database are processed. We do not have a warm-up or cooling down period. Per experiment we perform four replications, where we change the random number that determines the sequence in which the phases enter the beamline. The number of replications has been determined, using a confidence level of 95% and a relative error of 5%.

6 Results

In this chapter, we analyze the results of our experiments. First in Section 6.1 we explain what KPIs we look at when we judge the results of our tests. Then, in Section 6.2 we analyze the results of the test we performed. We use the KPIs of Section 6.1 to analyze our results, but also go into more detail when this is necessary. Finally, in Section 6.3 we give a summarizing conclusion on this chapter.

Before we discuss the results of our tests we want to address the fact that all experiments have been performed with the same routing approach. We modelled the beamlines like Chapter 5 explained. Because we model every beamline as a collection of connected nodes, we can apply our routing solution from Chapter 4. Due to the connections between the nodes we can generate all possible physical paths. From the physical paths we can generate all possible routes through the beamline, because we know what machine nodes (and possible operations) are available per physical path. We then use the set of possible routes with our routing rules to determine the best route for each nesting/job.

All values in this chapter are scaled, in order to keep the real values confidential. The base values are the experiment results using scenario 1 and dispatching rule RAND on a busy beamline. We set these values at 100 and scale the rest of the values from all other experiment such that their percentage difference stays the same. When the base value was 20 and another value was 22, the base value becomes 100 and the other value becomes 22/20*100 = 110.

6.1 Measurement KPIs

In Chapter 2, we defined KPIs that are important to the stakeholder. We use these KPIs and define KPIs that we can use to measure the results of our test. Table 6.1 shows these measurement KPIs.

Table 6.1: Measurement KPIs

KPI	Explanation
Average relative costs per job	This KPI speaks for itself, it measures the average relative costs to produce a job. It is the average taken over all jobs that have been processed.
Average output of the beamline in tons per hour	In the end, the output of the beamline is the KPI most customers look at. More output means more production and means that the customer can make more money. The output in the steel processing market is often measured in tons per hour, so we will also use this unit. We calculate the average tons per hour by dividing the total amount of tons that are produced by the total amount of hours that the beamline has been processing.
Average make span per phase	The make span per phase is the time between the first job of the phase entering the beamline and the last job of the phase leaving the beamline. We take the average make span over all phases that have been produced.
Average throughput time per job	The throughput time per job says something about how fast individual jobs go through the beamline. The throughput time starts when a job enters the first CT and ends when it leaves the beamline. We take the average over all jobs that have been produced.

6.2 Analysis of the results

To keep this section readable, we put the tables with all results in Appendix J and refer to these tables or show parts of the table when they are required for clarification. With our simulation, want to test four things:

- The impact of the dispatching rules on the KPIs.
- The impact of the different routing rules scenarios on the KPIs.
- Whether the impact of the routing rules is the same on a busy and a quiet beamline.
- How the dynamic routing rules compare to the static routing that is used at the moment.

We first discuss the impact of the dispatching rules in Section 6.2.1. In Section 6.2.2. we analyze the effect of the different scenarios of our routing rules. Here, we use the model as we described in Chapter 5. In Section 6.2.3. we adjust the model slightly by only generating one phase per day. This way we can compare the effect of our routing rules in a busy and a quiet beamline. After we analyzed our own routing rules and dispatching rules, we still need to compare whether the rules developed in this research are actually an improvement over the static rules that MCX uses at the moment. Section 6.2.4 shows the results of this comparison.

6.2.1 Dispatching rules

We test the dispatching rules, using scenario 1, which puts an equal focus on all attributes. Looking at the results in Appendix J.1, we see that the dispatching rules do not impact the costs, so we leave this KPI out of the comparison. The other KPIs do change. In the end, steel manufacturers care the most for a high output of the beamline and a short make span of phases, so these are the KPIs we use to compare the dispatching rules. Table 6.2 shows the test results of our tests.

Table 6.2: Results test dispatching rules

Scenario	Dispatching Rule	Average tons per hour			Average make span per phase (s)		
		Beamline 1	Beamline 2	Beamline 3	Beamline 1	Beamline 2	Beamline 3
1	0 (RAND)	100.0	100.0	100.0	100.0	100.0	100.0
1	1 (LWRF)	100.2	99.5	99.6	103.3	104.1	98.2
1	2 (MWRF)	100.0	100.1	99.7	104.1	100.1	95.1
1	3 (LWBF)	100.4	100.3	100.1	103.4	105.5	98.5
1	4 (MWIMF)	100.6	100.5	100.1	103.3	102.2	97.0
1	5 (BWSF)	100.2	99.8	100.3	103.1	107.5	100.5

Table 6.3: Best dispatching rules

	Average tons per hour			Average make span per phase		
	Range	Best off-line rule	Best on-line rule	Range	Best off-line rule	Best on-line rule
Beamline 1	[100 – 100.6]	1 (LWRF)	4 (MWIMF)	[100 – 104.1]	1 (LWRF)	5 (BWSF)
Beamline 2	[99.5 – 100.5]	2 (MWRF)	4 (MWIMF)	[100 – 107.5]	2 (MWRF)	4 (MWIMF)
Beamline 3	[99.6 – 100.3]	2 (MWRF)	5 (BWSF)	[95.1 – 100.5]	2 (MWRF)	4 (MWIMF)

In Chapter 5 we stated that we want to test our routing rules with the RAND dispatching rule, an off-line dispatching rule and an on-line dispatching rule. Table 6.3 shows that the dispatching rules 2 (MWRF) and 4 (MWIMF) are the most obvious choices to use in our future tests, since these rules show the best results in most experiments. We also see that the impact of the dispatching rules are much bigger on the make span per phase than on the tons per hour. The gap between the best and worst value for the tons per hour is less than 1% for all three beamlines, while the gap for the make span ranges are

4%, 7.5% and 5.5% for beamline 1, 2 and 3 respectively. Table 6.2 also shows that the RAND dispatching rule performs very well on beamline 2 and has average results on the other two beamlines. The small gap between the worst and best value of the average tons per hour, together with the fact that the RAND dispatching rule scores average to good, could mean that the impact of the dispatching rules is rather small. However, we cannot yet draw conclusion, without looking at the effect of combining dispatching rules with the different scenarios of our routing rules.

6.2.2 Routing rules

We test the performance of the routing rules using the scenarios from Chapter 5. We test the different scenarios in combination with the three dispatching rules we chose in Section 6.2.1. Table 6.6 shows the experiment results of all scenarios in combination with the RAND dispatching rule. Table 6.5 and Table 6.6 show the results of using the scenarios in combination with MWRF and MWIMF respectively. The results are all from experiments on Beamline 1, the results of the other two beamlines can be found in Appendix J.1.

Table 6.4: Results Beamline 1, using dispatching rule 0 (RAND)

Scenario	Disp. Rule	Avg. Costs Per Job		Avg Tons Per Hour		Avg Make Span Per Phase (s)		Avg Throughput Time Per Job (s)	
1	0	100,0	-	100,0	-	100,0	-	100,0	-
2a	0	100,6	0.6%	101,0	1.0%	99,2	-0.8%	101,1	1.1%
2b	0	99,1	-0.9%	101,0	1.0%	94,0	-6.0%	95,8	-4.2%
3a	0	98,8	-1.2%	99,9	-0.1%	99,6	-0.4%	100,1	0.1%
3b	0	98,8	-1.2%	100,5	0.5%	95,2	-4.8%	95,8	-4.2%
4a	0	101,2	1.2%	98,9	-1.1%	102,9	2.9%	102,1	2.1%
4b	0	101,2	1.2%	100,1	0.1%	102,8	2.8%	102,3	2.3%

Table 6.5: Results Beamline 1, using dispatching rule 2 (MWRF)

Scenario	Disp. Rule	Avg. Costs Per Job		Avg Tons Per Hour		Avg Make Span Per Phase (s)		Avg Throughput Time Per Job (s)	
1	2	100,0	-	100,0	-	104,1	-	106,2	-
2a	2	100,6	0.6%	100,8	0.8%	102,2	-1.8%	105,5	-0.7%
2b	2	99,1	-0.9%	101,0	1.0%	99,5	-4.4%	103,9	-2.2%
3a	2	98,8	-1.2%	98,9	-1.1%	103,8	-0.3%	107,0	0.7%
3b	2	98,8	-1.2%	100,4	0.4%	100,8	-3.2%	103,7	-2.3%
4a	2	101,2	1.2%	98,7	-1.3%	108,4	4.1%	109,3	2.9%
4b	2	101,2	1.2%	99,3	-0.7%	106,3	2.1%	108,3	2.0%

Table 6.6: Results Beamline 1, using dispatching rule 4 (MWIMF)

Scenario	Disp. Rule	Avg. Costs Per Job		Avg Tons Per Hour		Avg Make Span Per Phase (s)		Avg Throughput Time Per Job (s)	
1	4	100,0	-	100,6	-	103,3	-	104,0	-
2a	4	100,6	0.6%	101,3	0.7%	99,3	-3.9%	101,3	-2.6%
2b	4	99,1	-0.9%	101,7	1.1%	97,4	-5.7%	100,9	-2.9%
3a	4	98,8	-1.2%	99,2	-1.4%	99,9	-3.2%	101,5	-2.4%
3b	4	98,8	-1.2%	101,0	0.4%	98,0	-5.1%	100,0	-3.8%
4a	4	101,2	1.2%	98,6	-2.0%	107,5	4.1%	107,5	3.4%
4b	4	101,2	1.2%	99,1	-1.5%	105,4	2.1%	105,5	1.5%

Impact of different scenarios

We want to test if every scenario impacts the performance of the beamline like we expect.

Lowest costs

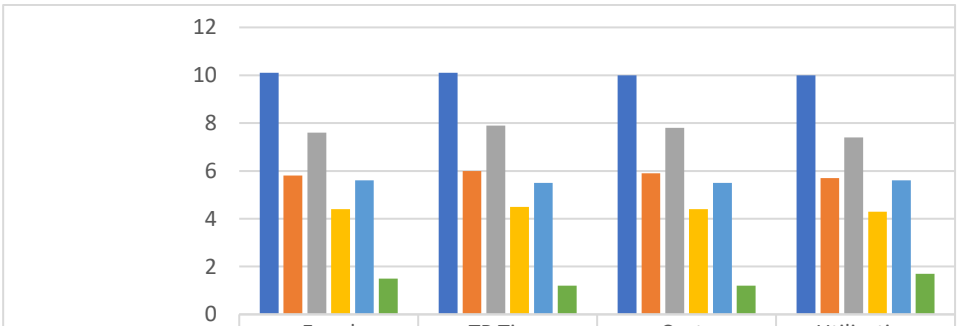
Table 6.6, Table 6.5 and Table 6.6 show that for all three dispatching rules, scenarios 3a and 3b result in the lowest costs. These scenarios focus on the "Costs" attribute, so this matches our expectation. This conclusion also holds when we look at the results of beamline 2 and 3 in appendix J.1.

Most equal work spread

We cannot directly derive from Table 6.6, Table 6.5 and Table 6.6 which scenario spreads the work most equally over all machines. To do this we look at the production output per machine. We compare scenarios 1, 2b, 3b and 4b with each other and refer to the scenarios as "Equal", "TP Time" (short for throughput time), "Costs" and "Utilization" respectively, based on their focus. We choose these scenarios because they are focused purely on one attribute, thus should have the biggest impact.

We look how the different scenarios influence the average tons per hour per machine (Figure 6.1). The figures of the analysis of other two beamlines can be found in Appendix H. Figure 6.1 shows that with the "Utilization" focus, the tons/hour is distributed most equally over the marking machine, drill and plasma cutter, so the plasma cutter does take over work from the drill and marking machine. Table 6.7 also indicates that the "utilization" focus results in the lowest standard deviation over the production per machine. This means that every machine produces a more equal amount of tons and the work is more equally spread over the machines. From this we conclude that focusing on the "machine utilization" attribute (scenario 4a and 4b) does spread the work most equal over all machines, like it should.

The tons/hour of the machines with the "Equal" focus scenario are in between the rates of the other focusses, so the "Equal" focus can be used as a good middle ground of costs and machine utilization, since scenario 4a and 4b are also the most expensive scenarios. (This is also true for the other two beamlines)



	Equal	TP Time	Costs	Utilization
Shot blaster	10,1	10,1	10	10
Marking machine	5,8	6	5,9	5,7
Drill	7,6	7,9	7,8	7,4
Saw 1	4,4	4,5	4,4	4,3
Saw 2	5,6	5,5	5,5	5,6
Plasma cutter	1,5	1,2	1,2	1,7

Figure 6.1: Average tons per hour per machine on beamline 1

Table 6.7: Average tons per hour (statistics)

	Equal	TP Time	Costs	Utilization
Average	5.83	5.87	5.80	5.78
St. Dev.	2.65	2.76	2.73	2.55

Fastest throughput time

Finally , scenario 2a and 2b focus on the "Throughput time" attribute, so we expect that these scenarios result in the fastest throughput time per job. Table 6.6, Table 6.5 and Table 6.6 show that scenario 2b is indeed amongst the scenarios with the fastest average throughput time per job, but so is scenario 3b, which is actually faster per dispatching rule. Appendix J.1 shows that for beamline 3, scenarios 2 and 3 also result in the faster throughput times. These scenarios have in common that they both put minimum weight to the utilization attribute, which means they do not try to use every machine equally. Scenario 4a and 4b result in the longest throughput times on beamline 1 and 3, which further establishes that focusing on equal machine utilization is negative for throughput time.

Scenario for the best output

It would be ideal to have a single scenario that would maximize the output on every beamline. However, Table 6.6 and the tables in Appendix J.1 show that for each beamline another scenario results in the highest tons per hour. For beamline 1 the most tons per hour is gained when we focus on throughput time, for beamline 2 it is gained when we focus on machine utilization and for beamline 3 we gain the highest tons per hour when we focus on costs. Choosing the right scenario (from the scenarios we tested) can increase the tons per hour by 1% - 1.5%, compared to scenario 1.

We look at the results in Table 6.6 and the analysis in Figure 6.1 and try to conclude why the "Throughput time" focus results in the highest output on beamline 1. The "Throughput time" focus has relative low tons/hour on the plasma cutter and high tons/hour on the marking machine and drill. This means that marking and drilling operations are (mainly) done on the marking machine and drill (This is also true for "Throughput time" focus the other two beamlines).

To determine what type of focus results in the highest output per beamline, we need to consider the layout of the beamline. Both beamline 1 and 3 contain an outfeed node that is only reachable via a plasma cutter. Many routes come together and go through the same machine to reach an outfeed, which creates a chokepoint. At the start of the day, when there are few jobs in the system it might be useful for the plasma cutter to take over some work, to spread out the processing time. However, when more jobs arrive at the beamline and more jobs need to go through the chokepoint, that extra added processing time becomes extra waiting time for all these jobs. So it seems that focusing on machine utilization has a negative effect on the total production, when the beamline has a chokepoint. Beamline 2 does not have a such a chokepoint because there are many routes that go around the plasma cutter. Here it is useful to swap some work from a marking machine or drill to the plasma cutter, since it lowers the utilization of these machines and adds it a machine that can be avoided.

So, a good extension to the dynamic routing rules we developed in this research would be an attribute (or rework the "utilization" attribute for example), such that a focus on that attribute would always result in the highest tons per hour. This concept is further explained in Chapter 7.

6.2.3 Busy vs quiet beamline

In the previous tests, we tested the dynamic routing rules on beamlines that are almost always are filled with beams. Now, we test the dynamic routing rules and dispatching rules when only one phase per day is loaded onto the beamline. We do this to see if it is useful to select different scenarios and dispatching rules, based on how busy the beamline is.

Dispatching rules

We start again by testing the combination of scenario 1 with all dispatching rules and Table 6.8 shows the results. We see that the RAND, MWRF and MWIMF rules again result in the highest tons/hour and shortest make span per. Compared to the busy beamline, the absolute tons per hour are lower because the beamline is only processing 1 phase at the time, where, on a busy beamline, the beamline processes multiple phases parallel. Table 6.9 and Table 6.10 show that the gap between the best and worst performing dispatching rule is bigger regarding tons per hour, but smaller regarding the average make span per phase. The tables also show that on a quiet beamline, an off-line rule (MWRF) outperforms the best on-line rule (MWIMF). This is caused by the fact that the on-line rule bases its sequence on the data in the beamline. In a quiet beamline there is not much data to work with, so the on-line rule does not perform as well as on a busy beamline.

Table 6.8: Results quiet beamline

Scenario	Dispatching Rule	Average tons per hour			Average make span per phase (s)		
		Beamline 1	Beamline 2	Beamline 3	Beamline 1	Beamline 2	Beamline 3
1	0 (RAND)	85.7	88.3	75.2	32.0	35.6	23.1
1	1 (LWRF)	83.3	85.1	73.3	32.9	37.1	23.6
1	2 (MWRF)	85.6	89.1	76.3	32.8	36.2	23.1
1	3 (LWBF)	83.9	86.0	73.5	33.0	36.9	23.5
1	4 (MWIMF)	85.4	88.1	75.3	32.6	36.2	23.3
1	5 (BWSF)	83.4	85.5	73.1	33.0	37.3	23.7

Table 6.9: Best dispatching rules quiet beamline

	Average tons per hour				Average make span per phase			
	Range	Range Gap	Best off-line rule	Best on-line rule	Range	Range Gap	Best off-line rule	Best on-line rule
Beamline 1	[83.3 – 85.7]	2.9%	2 (MWRF)	4 (MWIMF)	[32– 33]	3.1%	2 (MWRF)	4 (MWIMF)
Beamline 2	[85.1 – 89.1]	6.3%	2 (MWRF)	4 (MWIMF)	[35.6– 37.3]	4.8%	2 (MWRF)	4 (MWIMF)
Beamline 3	[73.1 – 76.3]	4.3%	2 (MWRF)	4 (MWIMF)	[23.1 – 23.7]	2.4%	2 (MWRF)	4 (MWIMF)

Table 6.10: Best dispatching rules busy beamline

	Average tons per hour				Average make span per phase			
	Range	Range Gap	Best off-line rule	Best on-line rule	Range	Range Gap	Best off-line rule	Best on-line rule
Beamline 1	[100 – 100.6]	0.6%	1 (LWRF)	4 (MWIMF)	[100 – 104.1]	4.1%	1 (LWRF)	5 (BWSF)
Beamline 2	[99.5 – 100.5]	1.5%	2 (MWRF)	4 (MWIMF)	[100 – 107.5]	7.5%	2 (MWRF)	4 (MWIMF)
Beamline 3	[99.6 – 100.3]	0.8%	2 (MWRF)	5 (BWSF)	[95.1 – 100.5]	5.7%	2 (MWRF)	4 (MWIMF)

Routing rules scenarios

We use the RAND dispatching rule and the two rules from Table 6.9 and perform the same experiments as we did in 6.2.2., Table 6.11, Table 6.12 and Table 6.13 show the results of these experiments. The tables show that the same scenarios score well on the same beamlines, but the differences between the different scenarios are much smaller. Within scenarios, the differences between the three dispatching rules are also very small. Since the differences on a quiet beamline are smaller and the same dispatching rules and routing scenarios perform well compared to a busy beamline, we recommend to make the decision of which dispatching rule and routing scenario to apply, based on experiments with a busy beamline.

Table 6.11: Results beamline 1 (quiet), using dispatching rule 0

Scenario	Disp. Rule	Avg. Costs Per Job		Avg Tons Per Hour		Avg Make Span Per Phase (s)		Avg Throughput Time Per Job (s)	
1	0	100.3	-	85.7	-	32.0	-	26.3	-
2a	0	100.9	0.6%	86.3	0.7%	31.8	-0.9%	26.1	-0.8%
2b	0	99.4	-0.9%	86.1	0.5%	31.8	-0.8%	26.1	-0.7%
3a	0	99.1	-1.2%	86.2	0.6%	31.7	-1.0%	26.3	-0.1%
3b	0	99.1	-1.2%	86.3	0.7%	31.7	-1.1%	26.2	-0.5%
4a	0	106.4	6.1%	83.1	-3.1%	33.3	4.0%	26.9	2.3%
4b	0	106.7	6.4%	83.0	-3.2%	33.4	4.1%	27.0	2.4%

Table 6.12: Results beamline 1 (quiet), using dispatching rule 2

Scenario	Disp. Rule	Avg. Costs Per Job		Avg Tons Per Hour		Avg Make Span Per Phase (s)		Avg Throughput Time Per Job (s)	
1	2	100.3	-	85.6	-	32.8	-	27.8	-
2a	2	100.6	0.3%	86.2	0.7%	32.5	-0.8%	27.7	-0.5%
2b	2	99.4	-0.9%	86.3	0.8%	32.4	-1.0%	27.6	-0.9%
3a	2	99.1	-1.2%	86.0	0.5%	32.5	-0.8%	27.8	-0.3%
3b	2	99.1	-1.2%	86.1	0.6%	32.5	-0.8%	27.7	-0.6%
4a	2	107.0	6.7%	83.1	-2.9%	34.1	4.0%	28.4	1.9%
4b	2	107.0	6.7%	83.2	-2.8%	34.1	4.1%	28.4	2.0%

Table 6.13: Results beamline 1 (quiet), using dispatching rule 4

Scenario	Disp. Rule	Avg. Costs Per Job		Avg Tons Per Hour		Avg Make Span Per Phase (s)		Avg Throughput Time Per Job (s)	
1	4	100.3	-	85.4	-	32.6	-	26.8	-
2a	4	100.9	0.6%	86.0	0.7%	32.4	-0.6%	26.7	-0.5%
2b	4	99.1	-1.2%	86.1	0.8%	32.3	-1.0%	26.5	-1.3%
3a	4	99.1	-1.2%	85.6	0.2%	32.5	-0.6%	27.1	0.9%
3b	4	99.1	-1.2%	84.9	-0.6%	32.7	0.3%	27.9	4.0%
4a	4	107.0	6.7%	82.9	-3.0%	34.2	4.6%	27.1	0.9%
4b	4	107.6	7.3%	82.7	-3.2%	34.2	4.7%	27.4	2.0%

6.2.4 Dynamic vs static rules

We apply the static rules from Appendix H in our model and test them in combination with our dispatching rules. Table 6.14 and Table 6.15 show the results of these tests, for beamline 1 and 2 respectively.

Table 6.14: Beamline 1, static rules

Scen.	Disp. Rule	Avg Tons Per Hour	Avg Make Span Per Phase (s)	Avg Throughput Time Per Job (s)	Avg Tons Per Hour	Avg Make Span Per Phase (s)	Avg Throughput Time Per Job (s)
		Busy beamline			Quiet beamline		
Static	RAND	96,1	107,5	96,8	80,0	34,7	27,7
Static	LWRF	96,5	111,0	100,3	78,5	35,0	27,3
Static	MWRF	95,9	109,3	99,9	79,1	35,9	29,4
Static	LWBF	95,8	107,5	97,2	79,2	35,0	26,9
Static	MWIMF	96,2	110,5	99,6	79,8	35,2	28,4
Static	BWSF	96,4	110,8	98,6	78,1	35,3	28,1

Table 6.15: Beamline 2, static rules

Scen.	Disp. Rule	Avg Tons Per Hour	Avg Make Span Per Phase (s)	Avg Throughput Time Per Job (s)	Avg Tons Per Hour	Avg Make Span Per Phase (s)	Avg Throughput Time Per Job (s)
		Busy beamline			Quiet beamline		
Static	RAND	99,3	124,0	120,4	83,0	40,7	34,2
Static	LWRF	100,0	128,4	123,9	81,0	41,4	34,7
Static	MWRF	100,8	126,7	124,6	83,9	41,2	35,0
Static	LWBF	99,8	125,1	121,7	82,7	40,8	34,6
Static	MWIMF	100,1	128,8	125,8	82,9	40,9	34,9
Static	BWSF	99,5	129,6	124,7	81,4	41,5	35,5

We compare the results from the static rules to the results from the dynamic rules. Table 6.16 shows the four comparisons we want to make. For the first two comparisons, we set the dispatching rule as RAND, to mimic the use of no dispatching rule. In comparison 1 we compare the static rules to scenario 1 of the dynamic rules, because scenario 1 acts like the standard configuration of the dynamic routing. We want to know if the dynamic rules outperform the static rules without tweaking the attribute weights. In scenario 2 we compare the static rules to the best result we achieved with the dynamic routing. In scenario 3 and 4, we compare the same scenarios as we did in scenario 1 and 2 respectively, but consider all dispatching rules. The comparison results can be found in Table 6.17 to Table 6.20. The comparisons are all made using the tons/hour attribute, because this is ultimately the KPI that steel manufacturers care the most about.

Table 6.16: Comparisons

Comparison	Dispatching rule	Static Rule	Dynamic Rule
1	RAND	Static rules	Scenario 1
2	RAND	Static rules	All scenarios
3	All rules	Static rules	Scenario 1
4	All rules	Static rules	All scenarios

Table 6.17: Comparison Beamline 1, busy beamline (results in tons per hour)

Comparison	Static Rules	Dynamic Rules	Improvement
1	96,1	100,0	4.1%
2	96,1	101,0	5.1%
3	96,5	100,6	4.3%
4	96,5	101,7	5.4%

Table 6.18: Comparison Beamline 1, quiet beamline (results in tons per hour)

Comparison	Static Rules	Dynamic Rules	Improvement
1	80,0	85,7	7.1%
2	80,0	86,3	7.8%
3	80,0	85,7	7.1%
4	80,0	86,3	7.8%

Table 6.19: Comparison Beamline 2, busy beamline (results in tons per hour)

Comparison	Static Rules	Dynamic Rules	Improvement
1	99,3	100,0	0.7%
2	99,3	101,4	2.1%
3	100,8	100,5	-0.3%
4	100,8	101,5	0.7%

Table 6.20: Comparison Beamline 2, quiet beamline (results in tons per hour)

Comparison	Static Rules	Dynamic Rules	Improvement
1	83,0	88,3	6.4%
2	83,0	88,3	6.4%
3	83,9	89,1	6.3%
4	83,9	89,1	6.3%

From Table 6.17 to Table 6.20 we conclude that our dynamic routing rules outperform the static rules of MCX in all comparisons, except in comparison 3 on beamline 2, but this is only a different of -0.3%. All other comparisons show that the dynamic routing rules improve the performance of on busy beamlines by 0.7% to 5.1% and on quiet beamlines by 6.3% to 7.8%.

These improvement are based on the scenarios we determined in Chapter 5. We did not yet try to find an optimal weight configuration that would result in the maximum possible output of the beamline. This optimization step is something for further research. Together with the other advantages we mentioned, we conclude that our dynamic routing rules are an improvement over the static rules.

6.3 Conclusion

The node-based dynamic routing rules, as defined in Chapter 4, function on all three beamlines we used to experiment, without adjusting the rules. This means that the routing rules can be applied to different beamline layouts, without extra configuration. This is already a big improvement over the static routing MCX uses at the moment, which requires a lot of configuration per beamline.

To judge the results of our tests, we use four KPIs: the average relative costs per job, average output of the beamline in tons per hour, average make span per phase and average throughput time per job. The most emphasize is put onto the average tons per hour and the average make span per phase, because these indicators are the most important to indicate the production output of a beamline.

Dispatching rules

We have tested our dispatching rules in combination with routing scenario 1 and with the static rules of MCX. We did this on all three beamlines, both in a busy and quiet state. The dispatching rules MWRF and MWIMF consistently proved to be the best performing dispatching rules. The effect of choosing between a “bad” and “good” dispatching rule is much bigger on a quiet beamline than on a busy beamline. However, once one of the “good” dispatching rules (MWRF or MWIMF) is chosen, the differences between them are very small. Because the same dispatching rules function good on both a busy and quiet beamline, we conclude that the decision of the dispatching rule should be made on the basis of the experiments on the busy beamline.

Effect of the different routing rule scenarios

We designed the dynamic routing rules with the idea that the user can decide the focus of the routing rules by giving weights to the different attributes. We tested if the different attribute function like they should and conclude that a focus on the costs and machine utilization attribute do function like they should. A focus on the costs attribute results in the lowest average relative production costs per job and the machine utilization focus results in the most equal spread of the work over all machines. However, a focus on the throughput time attributes does not seem to be directly results in the lowest average throughput time per job. This seems to be more linked to a low focus on machine utilization.

We also tried to find an attribute focus that would consistently result in the highest output of a beamline. Right now, every beamline performs best with a different attribute focus, choosing the right scenario (from the scenarios we tested) can increase the tons per hour by 1% - 1.5%, compared to scenario 1. It would be ideal to have a single attribute that can be focused in order to gain maximum output, but to achieve this an attribute need to be added or a current attribute need to be altered. Another option is to try to find the optimal attribute weights per beamline.

Static vs dynamic routing rules

Finally, we compared the performance of our node-based dynamic routing rules to the beamline performance that can be achieved with the static rules of MCX. The dynamic rules outperformed the static rules on both beamlines we used in the comparison, in both a busy and quiet state. The comparisons show that the dynamic routing rules improve the performance of on busy beamlines by 0.7% to 5.1% and on quiet beamlines by 6.3% to 7.8%, compared to the static rules. Together with the gains in configuration time and optimization possibilities, we conclude that the dynamic routing rules are an improvement over the current static routing rules.

7 Conclusions and recommendations

This chapter summarizes the conclusions we drew during this research and explains how to move on. Sections 7.1 provides answers to the research question that came up in the first chapter, summarizes the conclusions of each chapter and discusses whether we achieved the research goal we set in Chapter 1. Section 7.2 explains the contribution of this thesis towards both the literature and practice. Finally, Section 7.3 provides suggestions for further research and expansions of both the routing rules and the way of testing the rules.

7.1 Conclusions

We recall the goal of this research:

The goal of this project is to evaluate the current routing rules used in the product routing of MCX, investigate possible improvements and design a solution that can be implemented together with the software department.

We looked at how the current routing rules function and interviewed important stakeholders, to discuss the pros and cons of the routing rules. We concluded that the limits of the current routing rules are that they are static, do not adapt to the contents of the beamline and need to be configured and programmed for every beamline individually. This does not fit in the current vision of MCX of producing their beamline as a complete system supplier. The beamline should be quickly configurable with standard routing rules that are applicable regardless of the beamline layout. On top of this, the current product routing mainly bases its decision making on the CTs that directly follow the RC where the routing decision is made. It does not take into account the occupation of the route that comes afterwards or the beams that still have to come and are influenced by the decision that is made.

In the literature, we did not find a way of product routing that is directly applicable to the beamlines of MCX. That is why we designed our own product routing solution. The first step in our solution is modeling the beamline as a network of nodes (Figure 7.1). A beamline contains infeed and outfeed points, RCs, CTs and machines, and can all be seen as individual nodes that together, when they are connected, form a beamline. This way of modeling the beamline allows us to model every possible beamline configuration, simply be connecting the nodes as they would be in reality. From the network of nodes, we can derive all possible routes that start in an infeed node and end in a possible outfeed location. Outfeed locations can be outfeed nodes, but also machines that are able to divide jobs (saws and plasma cutters).

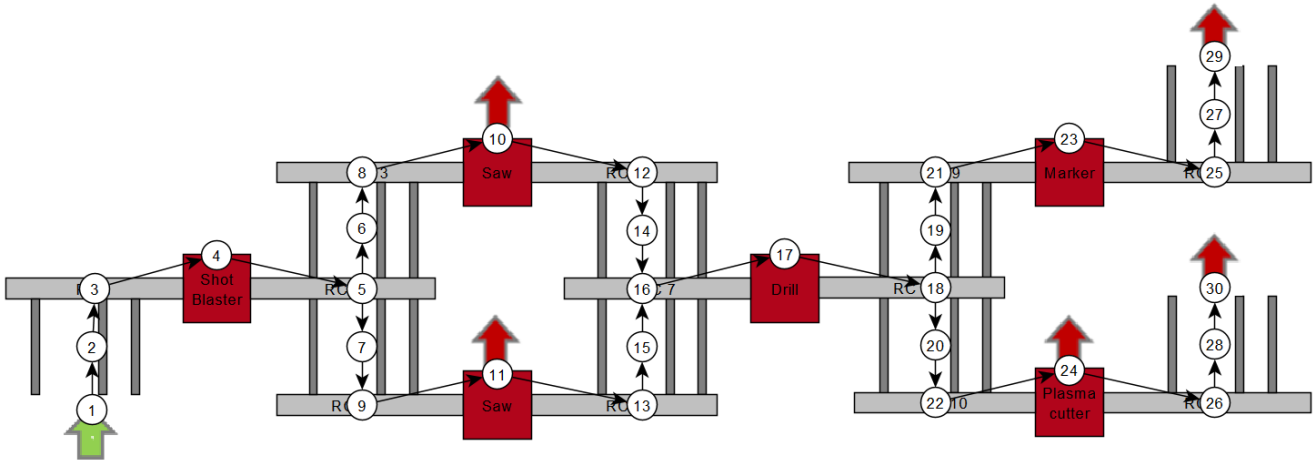


Figure 7.1: Beamline as network of nodes

Besides the physical path, a route also contains the machine assignment, so it determines which operation is performed at which machine. So, the combination of every physical path and possible machine assignments on these physical path create all routes through a beamline.

To pick the best route for a job, every route is scored on three different attributes: The expected throughput time, the expected relative processing costs and how well the work is divided over all machines in the beamline. The user of the beamline can apply weight factors to each attribute. These weight factors are gained by using the AHP method, where the user makes pairwise comparisons between every attribute on how much more they value one over the other. Within a nesting, it can occur that not every job has the same best nesting route. To decide with nesting route is taken, we first determine all routes that are candidates to be the best route. Then, we determine the total score of each candidate route, by adding the score of every product on the concerned route and select the candidate that has the best total score.

Besides the product routing, we also looked at the effect of dispatching rules, to look if they can further support the routing rules. These dispatching rules are:

0. Random (RAND) selects a random job. The random number is chosen with a uniform distribution.
1. Least-Work-Remaining-First (LWRF) selects the job with the least work remaining.
2. Most-Work-Remaining-First (MWRF) selects the job with the most work remaining.
3. Least-Work-for-Bottleneck-First (LWBF) selects the job with the least work for the current bottleneck.
4. Most-Work-for-Idle-Machine-First (MWIMF) selects the job with the most work for the machine with least work.
5. Best-Work-Spread-First (BWSF) selects the job that divides the work the best over all machines in the system. The goal of this rule is to provide every machine in the beamline with the same amount of work.

We tested our routing and dispatching rules with a simulation model we build in Plant Simulation. To test the routing rules, we created seven different scenarios, in which the attribute weights were different in every scenarios: 2 scenarios that focus on the “Throughput time” attribute, 2 that focus on the “Costs” attribute, 2 that focus on the “Machine utilization” attribute and one that has an “Equal” focus on all three attributes.

We tested the impact of the dispatching rules.

We test the dispatching rules on three beamlines, both in a busy and quiet state. The dispatching rules MWRF and MWIMF consistently proved to be the best performing dispatching rules. The effect of choosing between a “bad” and “good” dispatching rule is much bigger on a quiet beamline than on a busy beamline. However, once one of the “good” dispatching rules (MWRF or MWIMF) is chosen, the differences between them are very small, when they are used in combination with the same scenario. Because the same dispatching rules function good on both a busy and quiet beamline, we conclude that the decision of the dispatching rule should be made on the basis of the experiments on the busy beamline.

We tested the effect of the different routing rule scenarios.

We designed the dynamic routing rules with the idea that the user can decide the focus of the routing rules by giving weights to the different attributes. We tested if the different attribute function like they should and conclude that a focus on the costs and machine utilization attribute do function like they should. A focus on the costs attribute results in the lowest average relative production costs per job and the machine utilization focus results in the most equal spread of the work over all machines. However, a focus on the throughput time attributes does not seem to be directly results in the lowest average throughput time per job. This seems to be more linked to a low focus on machine utilization.

We also tried to find an attribute focus that would consistently result in the highest output of a beamline. Right now, every beamline performs best with a different attribute focus, so we need either a new attribute that can be used for output maximation, or try to find the optimal attribute focus per beamline. At the moment, choosing the right scenario (from the scenarios we tested) can increase the tons per hour by 1% - 1.5%, compared to scenario 1.

We tested the performance of the static vs dynamic routing rules

We compared the performance of our node-based dynamic routing rules to the performance that can be achieved with the static rules of MCX. The comparisons show that the dynamic routing rules improve the performance of the beamlines 0.7% to 5.1% on a busy beamline and 6.3% to 7.8% on a quiet beamline. We recall that this is achieved with one set of routing rules that has not yet been optimized, versus the static rules that consist of individual sets per beamline. Even setting the dynamic rules to a "equal" focus, which weights every attribute the same and could be seen as the basic configuration, outperforms the static rules. Together with the gains in configuration time and optimization possibilities, we conclude that the dynamic routing rules are an improvement over the current static routing rules.

Another advantage of having dynamic rules that are applicable on all beamlines, is that they can be developed further to increase their effectiveness. With the static rules, the rules need to be optimized per beamline and MCX needs to invest time and effort to just optimize one beamline at the time. With the dynamic rules, MCX can invest the same amount of time and effort in optimizing the dynamic rules and this investment will affect all beamlines that use the dynamic routing rules.

As final conclusion, we designed a dynamic routing solution and we showed that it is an improvement over the current static rules of MCX, not only based on the performance of the beamline, but it will also result on the long term in less configuration time per beamline. It offers a single standardized routing solution for every beamline, that subsequently can be tweaked by the user to focus on the attribute they find important. The challenge now is to fine-tune the attributes we use to score the routes and to find out what attribute needs to be focused, in what type of beamline, to gain the highest output in tons per hour.

7.2 Contribution to literature and practice

Literature

The focus of this thesis has been to design a dynamic product routing for the beamline of MCX. However, we think that the way we designed the routing rules, based on nodes, can be an addition to the current literature. In Chapter 3, we looked in the literature for methods that were used in previous studies regarding product routing in hybrid flow shops. The majority of these studies look at the routing as a stage to stage decision making process. When a job is finished in stage 1, it needs to choose which machine it will go to in stage 2. This requires the flow shop to be dividable into stages and simplifies the transportation between stages. In flow shops like the beamline of MCX, it is not obvious what could qualify as a stage in these type of models and the transport between the stages is not that easy to simplify. Considering Chapter 5, we already simplified our routing a lot, but it still plays an important role in the routing.

Our approach of solving the product routing problem with a nodes based routing has not been done before, as far as we know. We think it is a useful alternative of looking at product routing, because it is applicable on many different hybrid flow shops. When a hybrid flow shop can be divided into stages, it can also be divided into nodes, which means that our node-based routing solution can be applied. Because the existing routing solutions did not fit the beamline of MCX, we did not compare our routing solution to existing routing solutions in the literature. So, we cannot say how well our solution performs in relation to existing methods.

Another aspect of our product routing that is not common in the literature is the use of multiple attributes to determine the best route. Usually product routing is based on a single attribute: fastest throughput time or least deviation from the due date. Using multiple attributes and combining it with the AHP method to generates weights per attribute offers a way to include multiple focus areas in the product routing. We proved that varying the weights per attribute actually result in a different performance of the beamline.

Practice

From a practice point of view, our routing solution offers a new way of routing for MCX. Out solution moves away from the static rules that MCX uses at the moment and takes a dynamic approach. For MCX, the static rules are a very big time investment for each new beamline, as well as a lot of maintenance. Implementing the dynamic routing rules would require an initial time investment to program the routing rules, but afterwards it would reduce the configuration time of new beamlines a lot and the maintenance only need to be performed on one set of dynamic routing rules, instead the static rules of each individual beamline.

One of the goals we stated in Chapter 1 was that the routes should be easy to configure. To achieve this goal, we designed the rules with the philosophy that they should function on every beamline, independent of the configuration of the beamline. Modelling every beamline as a network of nodes makes it possible to quickly generate all possible route through the system and apply our routing rules. On top of this, the routing rules let the user focus on three different attributes: the throughput time, costs and machine utilization. Because we use the AHP method to determine the weights per attribute, the number of attributes can easily be adjusted. Adding an extra attribute to the routing rules only require a bigger pair-wise comparison matrix in the AHP method.

The tables that are used in the product routing show information like utilization rates of machine, amount of processing time per machine or the route that a certain job follows. These tables are used in the calculations, but they can also be presented to the user of the beamline, to give an status overview of the beamline. This gives extra clarity for the customer, which helps in understanding of why a job takes a certain route. When the customer understands why decision are made, it can be easier to convince them that the routing functions properly. With the static rules, the customer needs to thrust MCX that the static rules are performing as good as possible, which can be a harder bargain.

At MCX, we look at variable machine assignment, which is not a common concept. Often a machine is designed to perform a certain operation and cannot easily take over operations of other machines. This would vastly reduce the number of possible routes, since a route is purely based on its physical path in this scenario. It should still be stated which operation can be performed per route, but this is not variable anymore per physical path.

7.3 Recommendations/ Extensions

Optimization of the routing rules

We designed our node-based dynamic routing rules and showed that they are an improvement over the current static routing rules of MCX. We tested our routing rules on the basis of seven different scenarios, in which we varied the attribute weights of the routing rules. Our first expectation was that a focus on the “machine utilization” attribute should lead to the highest tons per hour on the beamline, because we expected that this focus would move work away from the bottleneck machine to other machines. However, this hypothesis only proved to be true on beamline 2. So, we suggest to test the routing rules on more different layouts of beamlines and examine when a certain attribute focus results in a high production output. In Chapter 6, we already explained that the “machine utilization” focus is negative for the production output on beamlines with a chokepoint.

Custom outfeed location

The modularity of the nodes allows easy manipulation of the configuration of a beamline and the generation of new routes. Sometimes a job is too long to travel through part of the beamline. In this case, the job needs to be manually removed from the beamline, which is often done on a RC or CT. Add an attribute to each node, for it to be an outfeed location; we already use this for machines that are capable of dividing a product. When there is a product that requires a manual removal from a RC or CT, select that node as outfeed node. A new set of routes can be generated from the infeed location

of the job to the selected outfeed node and the time that is required to remove the beam can be added as extra waiting time to the node that is selected as outfeed location.

Global vs local attribute weights

Have global attribute weights that are applied automatically to every job, like we model it now, and allow the option to give individual jobs different attributes weight. This can be useful when the global weights focus on costs for example, but there is a job that needs to be processed as quickly as possible. Then the local attributes weight of that job can be set to focus on throughput time.

Global vs local machine capabilities

We use a table *machine capabilities* to indicate which machine can perform which operations against which relative costs. Based on this table all possible routes are generated. Let us call this table the *Global machine capabilities*, because they apply for every beam. Our suggestion is to also use *Local machine capabilities* when a certain job is not allowed to be processed by a certain machine. It could also be applied on operation level, by setting the allowed machines per hole that must be drilled. This can be the case when a product needs to adhere to a certain quality level for example. The drill can create holes with a higher quality than the plasma cutter for example, so for that job we could use a *Local machine capabilities* where it is possible to disable the "drilling" operation for the plasma cutter. Subsequently we can disable all routes where drilling is done on the plasma cutter, reducing the set of feasible routes for the job, but keeping the product routing rules intact.

A more general approach would be to have different product groups, where every group has their own *machine capabilities* table. For a group that requires high quality, all operations can only be performed on the machines that deliver the highest quality. This means that the set of feasible routes of a job also depends on the job's product group.

Expected processing times

The product routing relies heavily on a good prediction of the expected processing times. In our simulation model, the processing times were based on assumptions, because there was not yet a standard way of accurately estimating the expected processing times. We know MCX is already working on tools to estimate expected processing times, so this recommendation can be seen as an exclamation on the importance of tools like that. Accurate predictions of the processing times increase effectiveness of not only the product routing, but also production planning and is essential in Project X.

Variable saw speed

The speed at which a saw cuts through the material can be changed by the saw user by acting on different parameters and settings. With this option, the saw blade lasts longer, reducing the relative costs, but it increases the processing time. This could be an additional option to choose from. We would apply this by adding it as separate route. This would result in two routes that are exactly similar, except for the relative costs and processing time of the saw.

Test the dispatching on beamlines without shot blaster

All beamlines that we tested in this research start with a shot blaster. We did this, because it is common for a large beamline to start with a shot blaster. However, the batches that need to be made in front of the shot blaster, can disturb the dispatching rules. So, we recommend to test dispatching rules, perhaps on smaller beamline layouts that do not have a shot blaster at the beginning. We did not do this ourselves, because the main focus of this thesis is the product routing, not the dispatching.

Nesting and production planning

Product routing is only a part of the total production process. Besides the product routing, there are the nesting of jobs and the production planning. We already applied some dispatching rules in our solution and showed that there were some positive effects to be gained from it. The dispatching rules only determine the sequence within the phase that arrives at the beamline. A total production planning can have a much bigger impact on the performance of a beamline than the dispatching rules we used, since it includes all upcoming production.

During the nesting process, the amount of processing time, the types of operations and the in- and outfeed locations per beam are determined. This is a lot of data that is used by the product routing. Sometimes, nesting routes are unviable for a nesting, because a certain job in the nesting cannot reach its outfeed location via that nesting route. It might then be the case that the entire nesting needs to take an inefficient nesting route, because of one job in the nesting. It might also be beneficial to not put too much processing time of one operation type in the same nesting.

8 References

Ali, M. (2012). Impact of Routing and Pallet Flexibility on Flexible Manufacturing System. *Global Journal of Flexible System Management*, 141-149.

Brah, S., & Wheeler, G. (1998). Comparison of Scheduling Rules in a Flow Shop with Multiple Processors: A simulation. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, 302-311.

C., R., & D., C. (1992). A multi-stage parallel-processor flowshop problem with minimum flowtime. *Eur.J.Oper.Res*, 137-143.

Cicirello, V., & Smith, S. (2004). Wasp-like agents for distributed factory coordination. *Auton. Agents Multi-Agent Syst.*, 237-266.

Cicirelly, V., & Smith, S. (2001). Improved routing wasps for distributed factory control. *The IJCAI-01 Workshop on Artificial Intelligence and Manufacturing*, 33-38.

Ghosh, S., & Gaimon, C. (1992). Routing flexibility and production scheduling in a flexible manufacturing system. *European Journal of Operational Research*, 344-364.

Hill, T. (1985). *Manufacturing strategy, the strategic managment of the manufacturing function.* Hong Kong: Terry Hills.

Kahraman, C. (2008). *Fuzzy Multi-Criteria Decision-Making.* Istanbul: Springer.

Krajewski, L., Ritzman, L., & Malhorta, M. (2007). *Operations management - processes and value chains.* Upper Saddle River, NJ: Pearson Education.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: a survey. *Computers & Industrial Engineering*, 57-61.

Little, J. (1961). A Proof for the Queuing Formula: L = λW. *Operations Research*, 383-387.

Mes, M. R. (2017). *Simulation Modelling using Practical Examples: A Plant Simulation Tutorial.* Enschede: University of Twente.

Meyyappan, L., Saygin, C., & Dagli, C. H. (2007). Real-time routing in flexible flow shops: a self-adaptive swarm-based control model. *International Journal of Production Research*, 5157-5172.

Monat, J. P. (2009). The benefits of global scaling in multi-criteria decision analysis. *Judgement and Decision Making*, 492-508.

Naderi, B., Khalili, M., Taghavifard, M., & Roshanaei, V. (2008). A Variable Neighborhood Search for Hybrid Flexible Flowshops with Setup Times Minimizing Total Completion Time. *Journal of Applied Sciences*, 2843-2850.

Narasimhan, S., & Panwalkar, S. (1984). Scheduling in a two-stage manufacturing process. *Int. J. Prod. Res*, 555-564.

Owen-Hill, A. (2017, December 4). *Job Shop vs Flow Shop: Can Robots Work for Both?* Retrieved from Robotiq: https://blog.robotiq.com/job-shop-vs-flow-shop-can-robots-work-for-both

Ozmutlu, S., & Harmonosky, C. M. (2005). A real-time methodology for minimizing mean flowtime in FMSs with routing flexibility: Threshold-based alternate routing. *European Journal of Operational Research*, 369-384.

Pacini, E., Mateos, C., & Garino, C. (2013). Schedulers Based on Ant Colony Optimization for Parameter Sweep Experiments in Distributed Environments. *Handbood of Research on Computational Intelligence for Engineering, Science, and Business*, 1-39.

Parthanadee, P., & Buddhakulsomsiri, J. (2010). Simulation modeling and analysis for production scheduling using real-time dispatching rules: a case study in canned fruit industry. *Computers and Electronics in Agriculture*, 245-255.

Peng, C., & Chen, F. F. (1998). Real-time control and scheduling of flexible manufacturing systems: An ordinal optimization based approach. *International Journal of Advanced Manufacturing Technology*, 775-786.

Peng, Y., & Mcfarlane, D. (2004). Adaptive agent-based manufacutring control and its application to flow shop routing control. *Production Planning & Control*, 145-155.

Ruiz, R., & Vázquez-Rodríguez, J. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 1-18.

Sanchoy, K. D., & Prashanth, N. (1997). Selection of routes in a flexible manufacturing facility. *International Journal of Production Economics*, 237-247.

Siemens. (2019, September 25). *Tecnomatix Plan Simulation Factsheet*. Retrieved from Siemens.com: https://siemens.mindsphere.io/content/dam/plm/plant_simulation_fact_sheet.pdf

Vignier, A. (1996). Resolution of some 2-stage hybrid flowshop scheduling problems. *IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems*, 2934-2941.

Weng, W., Wei, X., & Fujimura, S. (2012). Dynamic routing strategies for JIT production in hybrid flow shops. *Computers & Operations Research*, 3316-3324.

Zammori, F., Braglia, M., & Frosolini, M. (2011). A measurement method of routing flexibilty in manufacturing systems. *International Journal of Industrial Engineering Computations*, 593-616.

UNIVERSITY
OF TWENTE.

Appendix A: Problem cluster and Goals of MCX

This appendix contains confidential information

Appendix B: Cross Transport types

Drag dogs: Drag dogs use collapsible notches that push a beam over the CT. They can move one beam at the time and can only move them in one direction. With large beam, the notch can push de beam against the inside of the profile (Figure B.0.1), however with smaller or closed profiles, the notch needs to push the beam against the outside of beam. This requires some space between beam for the notch to pop up, which means that beams cannot lie directly next to each other (Figure B.0.2). The fact that drag dogs can only move one beam at the time has important effects on the handling within the buffer. When the first beam is pushed from the CT to the RC, the drag dog needs to rearrange the other beams in the buffer one by one, pushing them all a little bit forward to fill the gap.

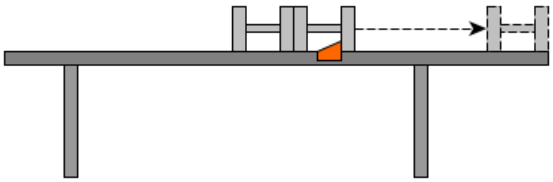


Figure B.0.1: Drag dog pushing large H-profile

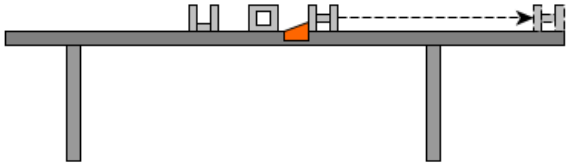


Figure B.0.2: Drag dog pushing smaller or closed profiles

Double drag dogs: Double drag dogs have two notches and are capable of moving beams in both directions. This is required when there is a cross transport at both sides of a roller conveyor

Liftable cross transports: Liftable cross transports lift the beam and are able to transport multiple beams at once. This is used when batches need to be moved to the infeed conveyor of painting or shot blasting machines for example. Moving only a single beam requires a lot of extra space, since the whole liftable unit needs to pop up, which is a big disadvantage (Figure B.0.3).

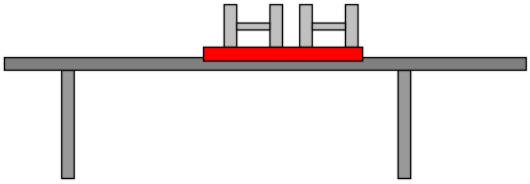


Figure B.0.3: Liftable cross transport

Liftable cross transports with drag dog: This type has both the functionality of the liftable cross transport and a single drag dog. So, it can move multiple beams at once, but also move single beams without the loss of buffer space that is cause by the normal liftable cross transport (Figure B.0.4).

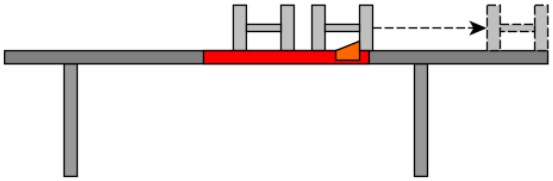


Figure B.0.4: Liftable cross transport with drag dog

Appendix C: Saw Time Analysis

This appendix contains confidential information

Appendix D: Calculating scoring attributes

This appendix contains confidential information

Appendix E: Nesting Generation

Jobs are put into nestings. Since nesting is not the focus of this research, but does have an impact on the routing of jobs, we make some assumptions:

- Within a phase, we combine beams with the same profile type into nestings. The max length of a nesting depends on the type of profile and the dimensions of the beamline. The max lengths that are commonly used in the steel market:

  I-Profile: 24 meters U-Profile: 18 meters M-Profile: 12 meters

- To create a nesting, we add jobs until the cumulative length exceeds the max length of the profile type. The length of the nesting will be equal to the cumulative length of the jobs. Table 4.1 shows a simplified example of how the nesting is done.

Table E.0.1: Nestings

Job	Profile type	Length (meter)		Nesting	Cumulative Length of nesting (m)
1	I	8		1	8
2	I	7		1	15
3	I	11		2	11
4	I	3		1	18
5	U	10		3	10
6	U	5		3	15
7	U	5		4	5
8	M	8		5	8

- We do not include the angled ends of beams. For example, when a beam has a 30-degree cut at the end, another beam with a 30-degree cut could be nested after it to minimize material usage and scrap. Figure E.0.1 shows an example.

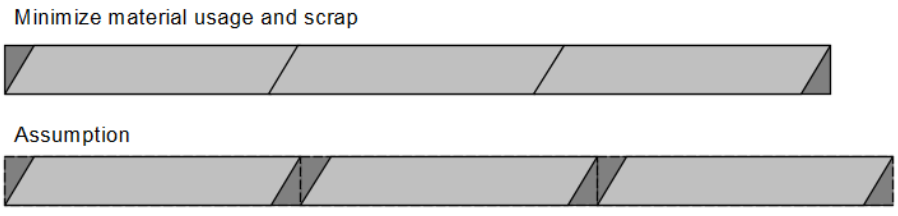


Figure E.0.1: Nesting example

- Jobs that are put in the same nesting will have the same route until the saw. We only nest jobs together when the requirement of every job can be fulfilled via the same saw. For example, when a certain job needs to go via saw 1 to fulfill all of its requirement and another job needs to go via saw 2, they cannot be nested together.

Appendix F: Beamlines

This appendix is confidential

Appendix G: Tables simulation model

The simulation model keeps track of a lot of data. Data that is used as input, data that is used during the simulation and output data.

Table G.0.1: Input tables

Input tables	**Contains**
ProductDatabase	Database with jobs
MachineCapabilities	Capabilities per machine and their relevant costs
BufferOverview	The buffers per machine
SetupTimes	Whenever a job is processed by a machine, the machine performs some standard preparations, like measuring and clamping the beam. These times are the same for every beam and stored in this table. These times are included in the processing time of a beam.
RouteFunctions	All routes, including details per route (Table 4.2 is an example of this)
NestingRoutes	All nodes per nesting routes
Routes	All nodes per physical routes
Merges	All RCs with multiple predecessors, including the concerned predecessors
Splits	All RCs with multiple successors, including the concerned successors

Table G.0.2: Processing tables

Processing Tables	**Contains**
ProductInfo	Information per job
NestingInfo	Information per nesting
BatchInfo	Information per batch
WorkPerMachine	The total amount of expected processing time per machine in the system
Nodes	The total amount of expected processing time per machine per node
MachineOverview	All machine nodes, including their in- and outfeed RC. Whether the machine is busy The total time the machine has been processing on the current day The tons produced on the current day

Table G.0.3: Output tables

Output tables	**Contains**
Phases	Output data per phase
Production	Output data per job
UtilizationRates	The utilization rates, per machines, per day A machine is being utilized when 1) the machine is processing a beam, 2) the outfeed RC of the machine is transporting a beam away from the machine, 3) the infeed RC is transporting a beam towards the machine. As soon as the outfeed RC cannot get rid of the beam it is transporting, due to a full or busy successor, the RC is stopped, as well as the machine and the machine is seen as idle, not utilized anymore. The utilization rate is the percentage of time a machine is being utilized during the active hours of the beamline. These active hours are the working hours between 6:00 and 22:00, but also the hours during the night in which the beamline is busy processing the leftovers of the day.
TonsPerHour	The average tons per hour, per machine, per day
Results	The average costs, tons per hour, make span per phase, total make span and throughput time per job

Appendix H: Static routing rules

This appendix contains confidential information

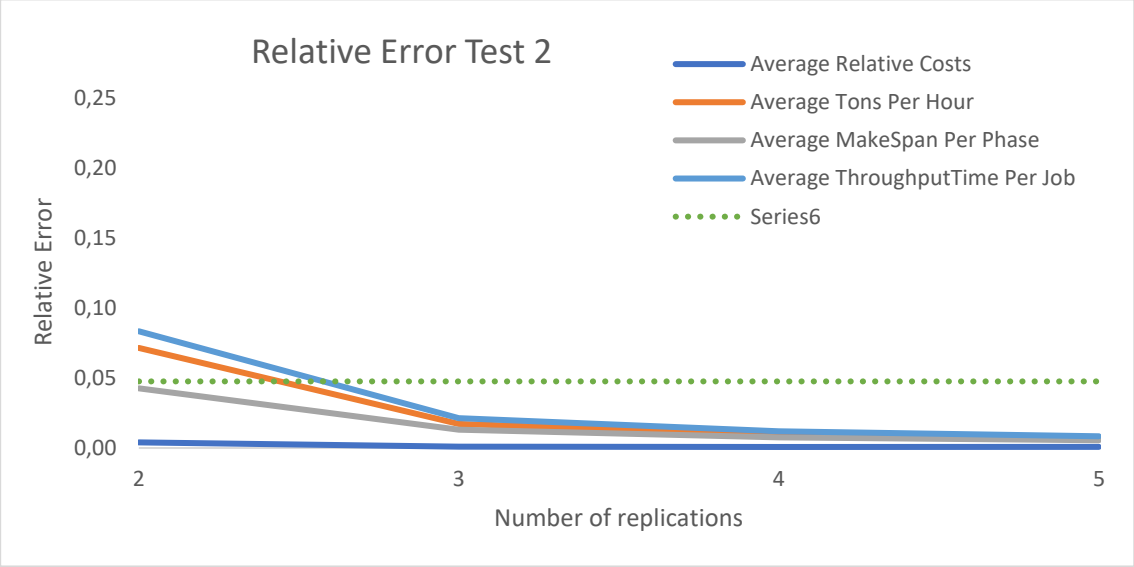
Appendix I: Relative Error tests

Test configuration 2

Dispatching Rule: 4

Pairwise comparison matrix:

	α_1	α_2	α_3
α_1	1	7	3
α_2	1/7	1	1/5
α_3	1/3	5	1

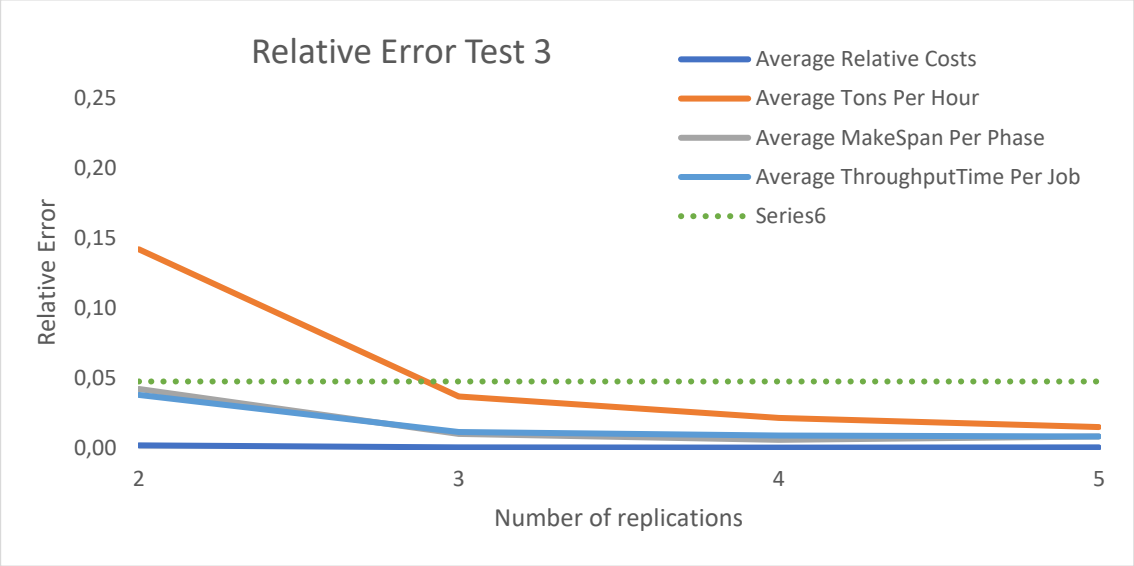


Test configuration 3

Dispatching Rule: 2

Pairwise comparison matrix

	α_1	α_2	α_3
α_1	1	1	1/9
α_2	1	1	1/9
α_3	9	9	1


Relative Error Test 3
Average Relative Costs
Average Tons Per Hour
Average MakeSpan Per Phase
Average ThroughputTime Per Job
Series6
Relative Error
0,25
0,20
0,15
0,10
0,05
0,00
2
3
4
5
Number of replications

Appendix J: Results

This appendix contains confidential information.

Appendix H: Analysis results

This appendix contains confidential information.