Technische Universiteit
Eindhoven
University of Technology

Department of Mechanical Engineering

Micro Device Fab system modeling

Master Thesis

Department: Mechanical Engineering
Research group: Dynamics and Control
Report number: DC 2016.039
Student: M.A.M.W. Borm
Identity number: 0741899

Supervisors:
Prof. dr. ir. I.J.B.F. Adan
Ir. M.A.M. Grooten

Eindhoven, Juli 2016

Summary

In this research a manufacturing line for hybrid electronics is investigated. After becoming familiar with the prototype design, the behaviour of the system is further observed. With the use of a simulation model the estimated decision rules behind the control software are verified by comparing it with the behaviour of the real life system. From this there could be concluded that the current systems performance was poor, with only a 40% bottleneck machine usage. Besides that, deadlocks could occur while processing multiple different product flows simultaneously.

Based on these insights, new system design rules are formulated, starting with the system architecture. When equipping the robot with a dual arm instead of a single arm the bottleneck machine usage could be increased to 82%. For the other design variables (e.g. buffers, alignment, etc.), design rules are formulated based on research and observations. The other important aspect of the manufacturing line is the job scheduling. The job shop problem is characterized by multiple different product flows with reentry, job transportation by robot, and the ability of applying priority to process steps. Two scheduling heuristics are studied: a greedy algorithm (GA) and the shifting bottleneck heuristic (SBH). For the GA decision rules are dependent on the number of products the robot is holding. The SBH is also adjusted to better perform for the particular job shop problem.

The performance of the determined design rules are tested by simulating general use cases and two real possible manufacturing line use cases. There is concluded that the SBH performs optimal for a small number of products, with a large variance in product flows and deterministic process times. For most use cases the SBH outperforms the GA in shortest makespan. However, the advantage of the SBH over the GA disappears for use cases with a large number of products, with a single process flow and process times that are multiples of each other. The performance of the SBH deteriorates more rapidly, compared to the GA, when there is a variance in process times or a machine drop out, and is therefore less robust. Another performance measure is the yield, which shows the percentage of products of which the priority requirement is met, and therefore the product equality. This requirement is always met for the GA, this is not the case for the SBH. Dependent on the goal of the manufacturing line, there can be chosen between the shortest makespan or the product equality.

In the future predictions of the system behaviour and performance can be made for both the prototype as for the new designs with the use of the simulation models. Due to the research application of the Micro Device Fab, a 10% makespan improvement is probably less important than the product equality, and therefore the GA is more suitable.

Contents

Chapter 1

Introduction

In this report we will model and analyze the Micro Device Fab system. First the topic will be introduced, by explaining the Micro Device Fab and the techniques involved. Secondly, the problem definition is defined and the research questions are formulated. Finally, the organization of the report will be presented.

1.1 Micro Device Fab

DoMicro BV is a company founded after an innovative design idea by the University of St. Petersburg [7] and Meyer Burger [5]. This design is a new production technology system for hybrid electronics. Hybrid electronics is the combination of printed components (batteries, displays, etc.) and integrated semiconductor devices (sensors, chips (IC)) onto a substrate [12]. However, the process capabilities of these technologies are still being explored. Nowadays, when a new hybrid electronic device is developed in the lab, it is immediately scaled to mass production. During this mass production achieving a good product yield is very difficult and reduces the time to market. This can be prevented with the use of the Micro Device Fab during the research and development phase. The Micro Device Fab is a complete manufacturing system for the development of raw materials to products and it can be adapted to the needs of each customer. The manufacturing system consists of multiple production tools, which are connected by a robot track. With the use of a production sequence (flow) for each product, the robot track will move the substrates within the system to the required process steps. This will mimic the eventual mass production and will allow the developers to conduct experiments and increase the yield, which will reduce manufacturing costs and decrease the time to market. The system can also be used for the manufacturing of small to medium product series. Due to the flexible design of the system, parallel production of different products with a low volume can be realized. Figure 1.1 shows an example of a Micro Device Fab.
The complete Micro Device Fab system is enclosed, which enables the creation of a clean room environment inside the system. Therefore, no large clean room is needed, which will result in reduced investments and no need for operators to wear the appropriate clean room clothing.

Micro fabrication processes

There are multiple micro fabrication processes that can be used in the Micro Device Fab. The main process is the use of printed electronics, a possible flow will be further explained. Before printing

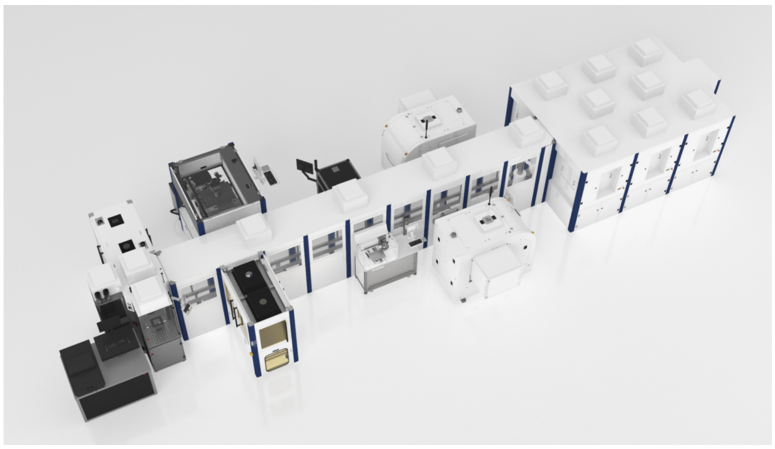

Figure 1.1: Example of a Micro Device Fab of DoMicro BV.

on a substrate, the surface properties have to be modified with a plasma printer, to provide a better bonding surface. Subsequently the printing can be started, which deposits droplets of a solution with nano particles (silver, copper, etc.) on a substrate in a certain pattern [6]. After a layer is printed, the dissolvent has to be evaporated in a hot plate, leaving only the nano particles. To connect these nano particles to construct a printed circuit, photonic sintering is used, which applies a low heat input by using an UV flash lamp[18]. On top of the finished printed circuit an isolating layer is printed in the required pattern to disable conductivity between the printed circuit and the next layer in certain areas. This isolating layer is then dried by using low power sintering, and dependent on the design possibly another silver layer is printed, dried, and sintered. On top of these created structures it is possible to bond a sensor or IC. A product example that is created with the processes as mentioned above is the FlexSmell [4]. This is a smart food label, which can monitor goods on, for example, their temperature as shown in Figure 1.2.

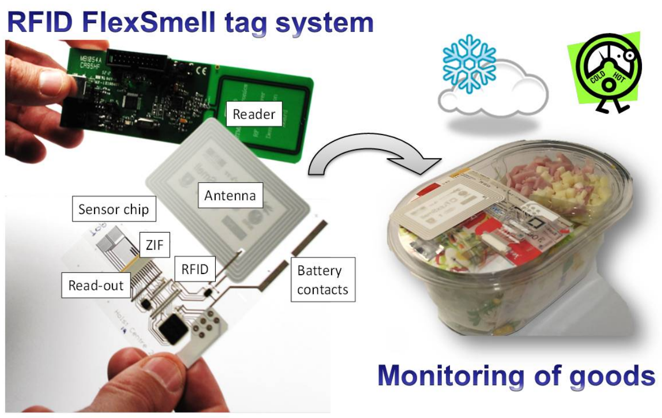

Figure 1.2: FlexSmell, smart food label which can be produced with the MicroDeviceFab

There are also wet chemistry processes. Examples of applications for these processes are microfluidic channels, or base structures of conductive and non conductive layers. First a coating is deposited with the use of a spin coater, which is a rotating table that dispenses a fluid. This coating is then dried in a hot plate, another layer can be coated or the coating can be exposed to cure a pattern. The none cured pattern can then be developed, and if two layer were applied, the layer below can be etched when immersed in a chemical solution [13]. This creates a pattern, which can be repeated dependent on the required structure.

1.2 Problem definition

For the development of new Micro Device Fab systems, the system design rules and the product scheduling need to be defined for achieving the best possible performance. The performance of a system can be measured by four key indicators:

1. The total completion time of a batch of jobs (makespan),
2. The average time a job is in the system (flowtime),
3. The amount of time that the busiest machine (bottleneck machine) is used (utilization),
4. The percentage of correct products (yield).

As mentioned in the previous section, the first prototype of the Micro Device Fab has already been developed. However, the behaviour, the routing which is generated by the scheduling decision rules, and the performance of this prototype is unknown and therefore needs to be determined. With this obtained knowledge of the current prototype, the different design parameters can be explored for the creation of new Micro Device Fab systems. These design parameters include the systems layout (mapping of the machines, design of the robots, buffers, etc.) and the product scheduling. The resulting design rules can than be applied to use cases to test their performance and robustness. The robustness can be tested by adding unexpected behaviour to the system, e.g. machine failures, and variation in the production process, e.g. flows and process times. In this research the following three research questions will be answered:

1. What is the design of the current system (prototype), how does the system behave (product scheduling), and what is the performance of the system?
2. Is it possible, based on the insights from the current system, to formulate design rules for the system layout and product scheduling?
3. What is the system performance when the decision rules mentioned above are applied to use cases, and how robust is the product scheduling for unexpected system behaviour and variation in production processes?

1.3 Report structure

In Chapter 2, the current prototype of the Micro Device Fab will be further explained, including the design, the user interface, and the product scheduling. Based on these insights, in Chapter 3, general design rules for the system architecture and scheduling will be formulated for Micro Device Fab systems. To determine the performance of these system design rules, these rules will be implemented into use cases in Chapter 4. First a general use case will be formulated to test the performance of the scheduling heuristics for a large number of different flows. Afterwards, real life use cases for two possible Micro Device Fab systems will be simulated and the performance and robustness of these systems will be tested. With these results, a conclusion can be drawn on the design rules and recommendations are formulated.

Chapter 2

MicroDeviceFab

The first prototype of this manufacturing system will be further explained, including the design, the user interface, the product scheduling, and the observations.

2.1 Design

The first prototype consists of two robot tracks, which are connected by a conveyer belt. The prototype is shown in Figure 2.1.

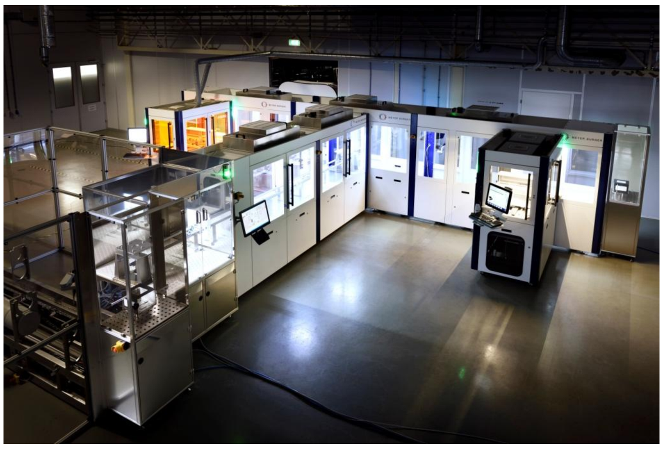

Figure 2.1: Set up of the current Prototype of the MicroDeviceFab

A schematic view of the MicroDeviceFab is shown in Figure 2.2. On the upper side of the conveyer belt, the first robot track is located. This robot track is connected to a carrier (buffer), with 10 substrates positions, conveyer belt and two machines, the IP410 and LP50. The carrier supplies substrates to the system, which can travel between the two robot tracks via the conveyer belt. The IP410 is a inktjet printer and the LP50 is a plasma printer. Both these machines are equipped with a computer in which recipes (pattern instructions) can be created. The second robot track, which is located on the lower side of the conveyer, is connected to another carrier with 10 substrate positions and the conveyer belt. Normally, this robot is also connected to multiple machines, which is the hot and wet cluster, in which etching and layer deposition can be applied. However, in this prototype, these machines were not available and will not be included in the simulation model. Also on each side of the conveyer belt, there is an alignment station, which is shown in Figure 2.3.

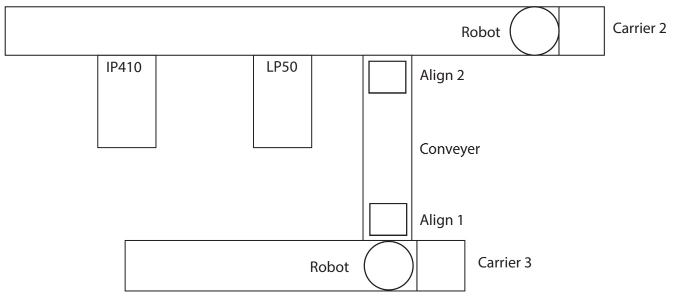

Figure 2.2: Schematic lay out

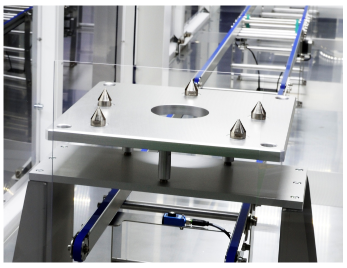

Figure 2.3: Alignment station

This alignment station consists of five pins on which the robot places the substrate two times to align the substrate. A substrate is aligned before and after every process step. The main computer is positioned on the right side of the conveyer belt at alignment station one. From this computer the manufacturing system is controlled and production flows can be started.

MCS: User Interface

The user interface of the Micro Device Fab is shown in Figure 2.4.

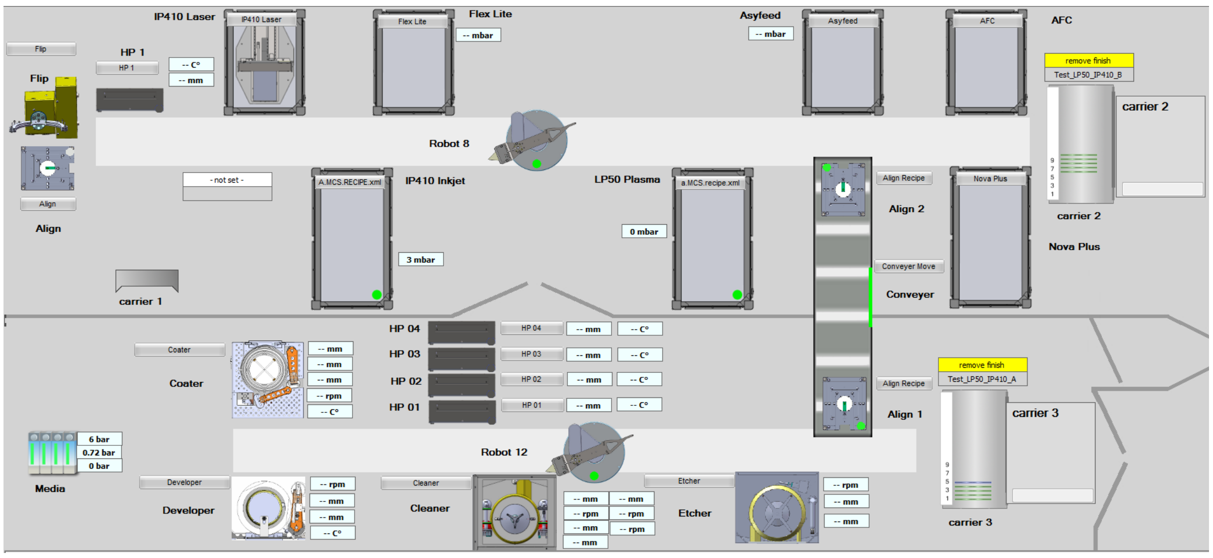

Figure 2.4: User interface of the Micro Device Fab

The user interface and control software is a program called MCS (Machine Control Software) and shows a schematic top view of all the systems in the Micro Device Fab. Robot 8 is the robot which is described as the first robot on the upper side of the system, and robot 12 is the second robot. In each carrier, the number and location of the substrates are shown, and when a substrate enters the system, the substrates are visually moved to were they are located. How flows are created and started is further explained in Appendix A.1.

When production is started with the MCS, log files can be extracted from the program. In these log files, the occupation of each machine is shown and the colour of the blocks in these log files shows which product is occupying what machine, by having a different colour for each product. The robot only shows a coloured block when a product is grabbed. A grey colour in the shows the time that a product is waiting and is finished processing in the machines. An example of a log file for the production of 3 products is shown in Figure 2.5. The flow shown in Figure 2.5 starts in carrier 3, from which the substrates are retrieved by robot 12. Afterwards, the robot aligns the substrate and deposits it onto the conveyer belt, which moves the substrate to robot 8. The substrate is fetched by robot 8, gets aligned, and than transported to be processed in the LP50. When finished, the substrate is removed from the machine by robot 8, again aligned, and moved to the IP410. This is the final process step, after which the substrate is aligned and moved to carrier 2.

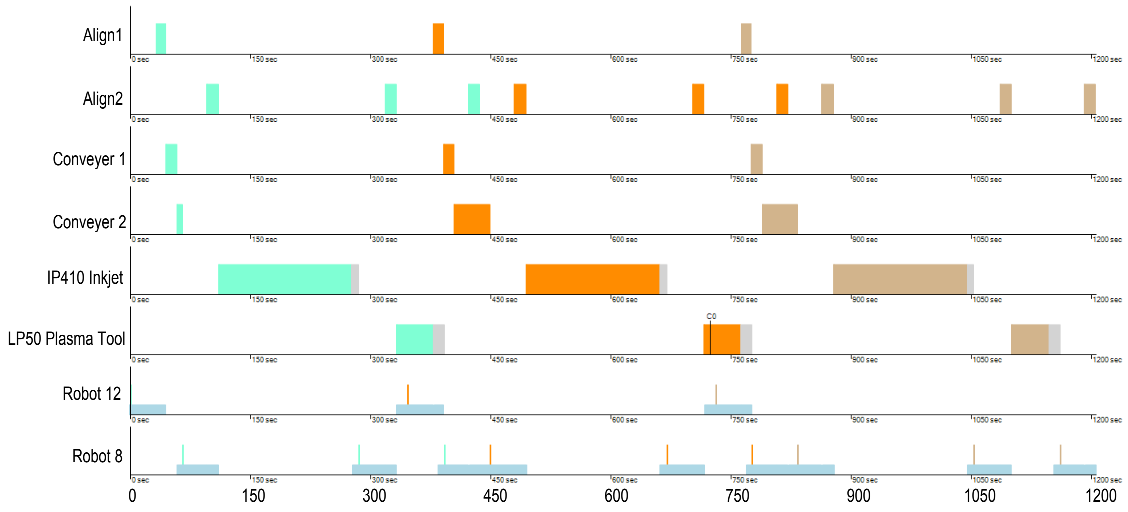

Figure 2.5: Example of a log file from the MCS program

For the flow shown in the log file, the bottleneck machine is the IP410. The utilization of this machine is only 40 %. This is the result of the single armed robot, which can only transport one substrate at a time, and the product scheduling. The product scheduling will be further investigated in the next section.

2.2 Product Scheduling

The scheduling decision rules which coordinate the movement of the substrates through the system are unknown, due to externally developed closed software. However, by observing the system, certain aspects can be identified. The robot reacts to the state of the system. The system state consists of the communicated information of available or busy machines/robots, and the location and information of the substrates. The communication structure of the machines with the main system is further described in Appendix A. When the machine of the first product step is available, a new substrates is released into the system and moved to this machine. When a machine has finished a process, it sends a message to the system. Then, when the machine for the next process step is available, the robot retrieves the substrate and moves it to the next machine. If the next machine is located on the other robot track, the product is only placed on the conveyer when the next machine is available. From this evaluation, the following decision rule can be derived:

- Substrates are released or moved in the system when the next machine is available.

To verify whether these decision rules are indeed correct, a simulation model is created in Chi with these decision rules and the design of the current Micro Device Fab. For this model, the handling times have to be determined by observing the Micro Device Fab, these results can be found in Appendix B.2. The structure of the simulation model and the implementation of the decision rules can be found in Appendix B.1. To verify the decision rules, the product flow in the simulation model is compared to the product flow in the real life system. With the Micro Device Fab, multiple experiments were conducted with the machines that were available. These machines were the IP410 and LP50. The first flow that was tested was with carrier 3 as sender and carrier 2 as receiver. This flow exists of first moving the product from carrier 3 with robot 12 onto the conveyer to robot 8, into the IP410, afterwards to the LP50, with process times of 170 seconds and 50 seconds, and finally into carrier 2. In between every step, the substrate is aligned at the alignment station. This resulted in Figure 2.6, which shows the time charts of the actual system and the simulation model.

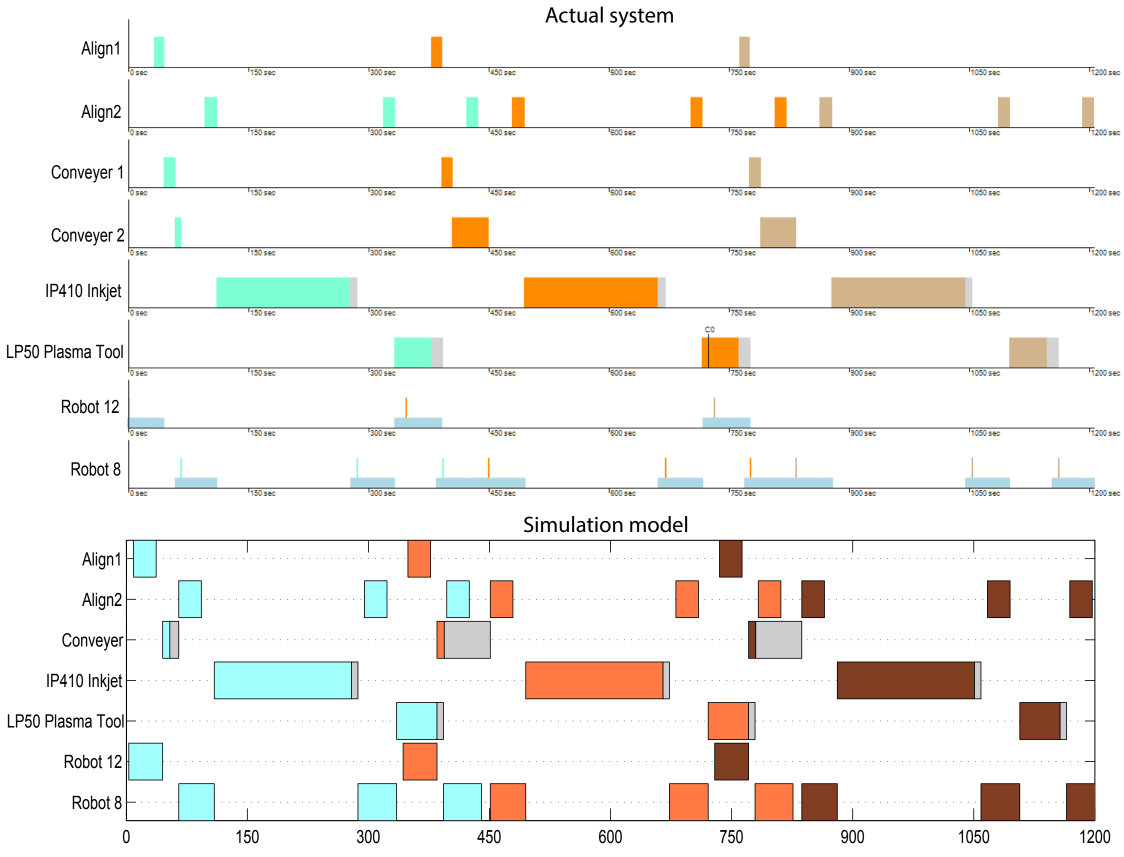

Figure 2.6: Time chart comparison of the actual system and of the simulation model.

In the time chart of the actual system, both sides of the conveyer are viewed separately. These are merged in the simulation time chart. The figure shows that the decision rules that were identified are accurate for this use case, because they result in an almost identical time chart. Besides that, the makespan of the two flows is identical. The same comparison has been executed for other flows, which also resulted in almost equal time charts and identical makespan for the simulation model and the actual system. From these observations, we can conclude that the decision rules are accurate and most likely implemented.

Because the Micro Device Fab was not completed with all equipment, the additional feature of applying priority to processing steps of substrates could not be tested in the real life system. However, the MCS program does allow the possibility of running simulations in which priority to process steps can be applied. This priority is applied to the hot plates. When creating a flow, a maximal allowed time for extracting the substrates from the hot plates can be defined. While observing the system behaviour when priority is applied, another decision rule is discovered:

- If there are multiple jobs in the hot plates, choose the job which has the least excess of the maximal allowed time.

This results in job overtaking, and for the other jobs in the hot plates to stay there for a very long time period. This would result in defect products due to the exceeded baking time. Besides the previously described decision rules and the priority decision rule, no other decision rules have been identified. This eventually results in the complete "Greedy" algorithm which is used in the MCS software.

2.3 Summary

Based on the observations of the Micro Device Fab system, the main troubles in the design and scheduling that need to be solved can be summarized as follows.

Robot
By observing the time charts of the system, the utilization of the bottleneck machine was only 40%. These machines are expensive and therefore need to be operational as much as possible, and should not be limited by the robot transportation times.

Conveyer
Inefficiency of transporting substrates via the conveyer belt from one robot track to the other. Substrates have to wait until the next machine is available to travel onto the conveyer, which results in an increased substrate flow time and a lower machine utilization.

Alignment station
Aligning is very time consuming, because the robot moves the substrate very slowly two times on the alignment station and the alignment is executed between every process step. Besides that, the required alignment accuracy of 100 micron is not achieved, due to the friction between the pins and the substrate.

Deadlock
A deadlock occurs when a job in machine b has to be move to machine a, and the job in machine a has to be move to machine b. These two jobs have to be exchanged, however, the robot only has a single arm and no function to return a job to a buffer, and therefore a deadlock occurs.

Chapter 3

System design

In the previous chapter we studied the prototype of the Micro Device Fab. Based on these insights, general design rules for the system architecture and scheduling for Micro Device Fab like systems can be formulated.

3.1 System Architecture

The system architecture will determine the redesign of the Micro Device Fab. The system architecture consists of the robot design, the number of buffers, the buffer design, the alignment station, the implementation of a conveyer, and the positioning of the machines. The current designs of these components are shown in Figure 3.1. The design considerations and rules for all these components for future systems will be further explained.

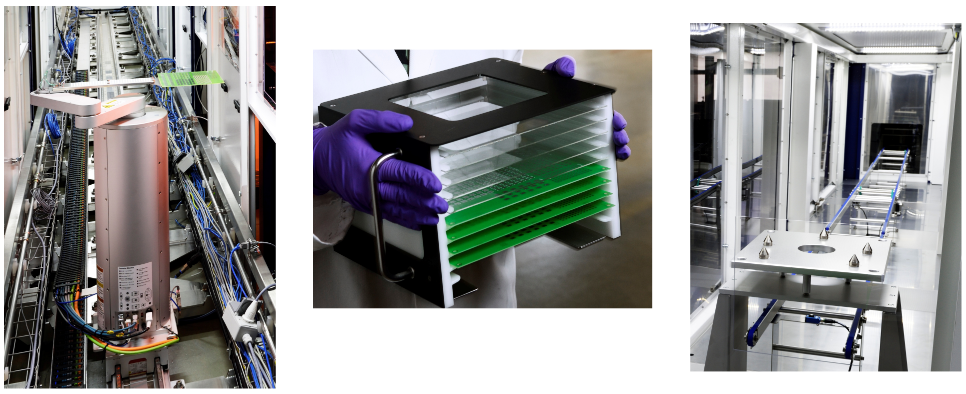
Figure 3.1: Prototype design of the robot, carrier, alignment and conveyor.

Robot

The current Micro Device Fab has a single armed robot, which means it can only carry one job at a time. However, when a job is finished in a machine, and the robot is carrying a job that has to enter that machine, it is impossible to exchange the jobs. Therefore, the effect of equipping the robot with a dual arm, which allows job exchange, will be tested. To do so, the simulation model that mimics the current systems behaviour and flow as described in Section 2.2 will be used. For the dual armed robot, slight adjustments are made to allow the robot to handle two jobs. Figure 3.2 shows the time charts of a single armed robot and a dual armed robot.

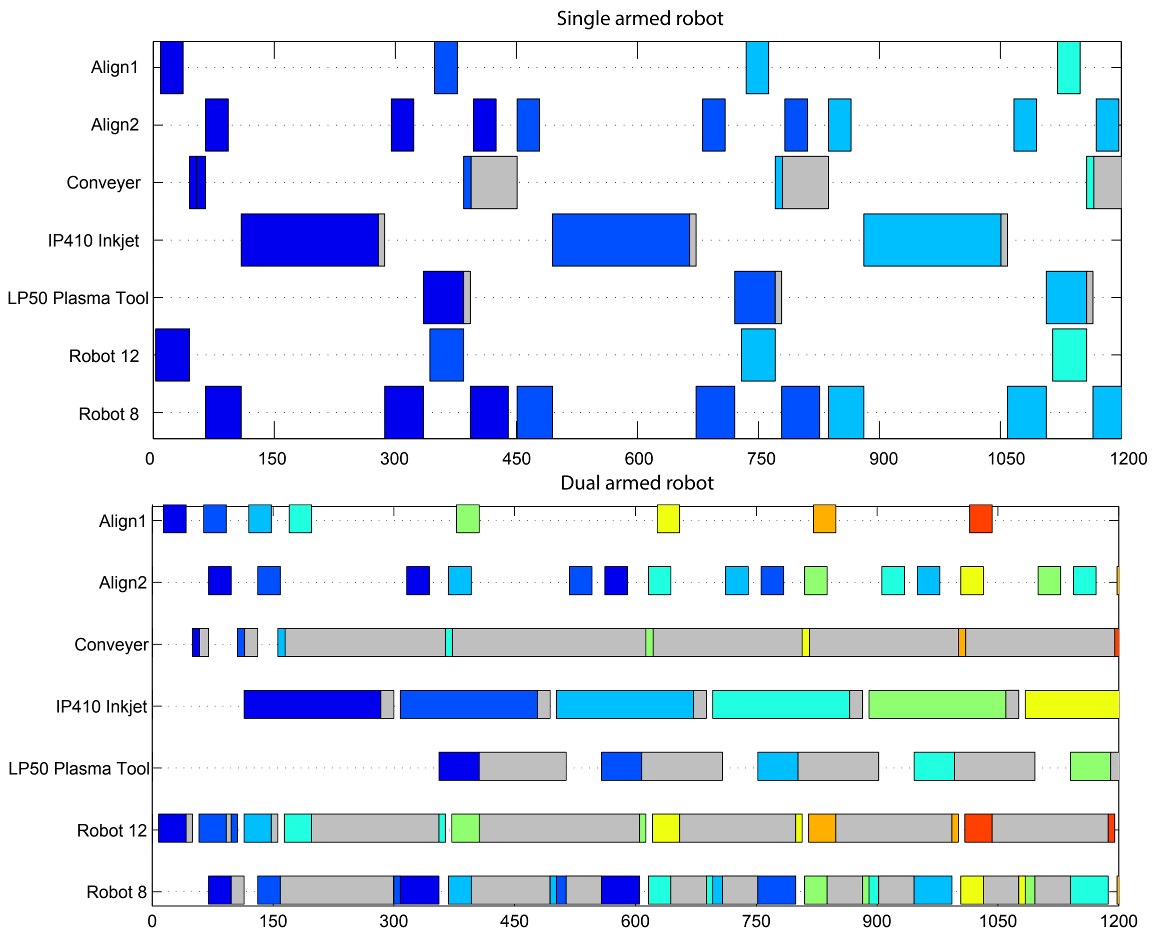

Figure 3.2: Upper time chart of the current system with a single armed robot, lower time chart with a dual armed robot.

Figure 3.2 shows a significant improvement of the throughput and flow times. For the single armed robot, 3 jobs are processed in 1200 time units, while for the dual armed robot 5 jobs are finished. This results in an increase of utilization of the bottleneck machine from 40% to 82%. Hence, we can conclude that a dual armed robot is a mandatory requirement for an efficient and productive manufacturing system.

Buffers

The current system is equipped with two buffers, both connected to a different robot track with space for 10 jobs as described in Section 2.1. For the jobs in each buffer, only one type of flow can be selected, so manufacturing multiple types of jobs simultaneously is only possible when the system has multiple buffers.

The required buffer size depends on the application. When a new manufacturing line is designed with given product flows, this system can be simulated to estimate the resulting throughput of the system. The buffer size should be sufficient to supply for an 8 hours shift. This results in a buffer size of 8 times the throughput per hour. For the example shown in Figure 2.6, this would result in a buffer size of 70. However, when more machines are connected to the system, the product flows may be composed of more processes on different machines and with multiple machine visits, which would decrease the throughput and hence the required buffer size.

Currently all jobs in a buffer have to be of the same type. However, it is requirement to be able to manufacture multiple types of jobs simultaneously. A software adjustment should be made which enables the operator to assign a type to each job. Besides that, it can occur that a job has to wait between process steps. To increase machine availability, there should be a possibility to return a job to a buffer, where it can wait for the next process step. This is also a software adjustment that should be implemented.

An important issue that has be taken into consideration is the contamination of the substrates due to substrate movement in the buffer. To prevent contamination there should always be a load and unload buffer. From the load buffer, substrates are released into the system, this is done bottom up, such that possible dust does not fall down on other substrates as can be seen in Figure 3.3. Therefore, the finished substrates should not be returned to the load buffer, but to the unload buffer. The unload buffer is loaded from top down to prevent dust falling onto the substrates below. Besides the load and unload buffers, also a buffer for waiting between process steps has to be added. This buffer should be equipped with enclosed substrate holders to prevent dust from spreading due to the irregular collection of the substrates.

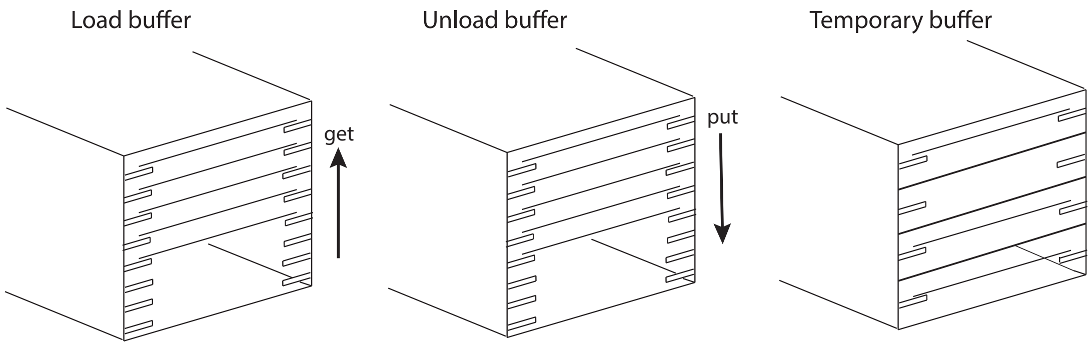

Figure 3.3: Representation of a load, unload and temporary buffer.

Alignment

During the production in the Micro Device Fab, substrates are aligned after each process step. However, it could be possible that the exchange between machines is within the required accuracy of 100 micron, and therefore alignment would not be necessary. This should be experimentally tested. If this appears to be true, only alignment after retrieving or returning a job to the buffer is required. This would already decrease the robot transportation time, but to further decrease the alignment time, the design described in Section 2.1 could be replaced by optical alignment. This would decrease the alignment time to only a few seconds. Also the accuracy of the current alignment station did not satisfy the 100 micron precision requirement, which would be satisfied when optical alignment is used.

Conveyer

The conveyer which connects the two robot tracks was added due to configurational constraints of the facility. For a new Micro Device Fab without these constraints, the conveyer would not be an efficient addition, because only one way traffic is possible, and shifting of the substrate on the conveyer while traveling has been observed. Therefore, connecting the robot tracks directly with a two sided open carrier would be a more efficient option. However, if there are spacial restrictions, an adjusted conveyer could be used. A possible adjustment would be to use two belts to allow two way traffic, and a better grip on the substrate to prevent movement, for example by clamping the substrate between pins.

Machines

The number and type of machines that are necessary for the creation of a new Micro Device Fab depend on the applications. After determining the applications and necessary machines, a layout can be created. Machines are placed in order of the product flows, but due to the high product variation, this is not always possible. Machines that have a large number of multiple visits, e.g. the hot plate, are placed in a central location in the manufacturing line.

Conclusion

From the evaluation of the prototype design and the redesigned architecture, the main differences between the modules can be summarized as follows.

Module	**Prototype design**	**Redesign**
Robot	• Single armed	• Dual armed
Buffers	• Size: 10 jobs • One type of buffer	• Size: throughput/hour times 8 (duration shift) • Three types of buffers (load/ unload/ temporary)
Alignment	• Five pin alignment between every process step	• Optical alignment when necessary with an accuracy of 100 micron
Conveyer	• Single way traffic • Substrate movement	• No conveyer • Or two way traffic and better substrate fixture with pins
Machines	• Wet cluster and dry cluster • No specific order	• Wet cluster and dry cluster • In order of most common flows

Table 3.1: Differences between the modules for the prototype design and the redesigned architecture.

The exact design details are determined based on simulation results, when the product flow of the system is known.

3.2 Scheduling

For the creation of new Micro Device Fab systems, a good performing product scheduling algorithm has to be developed. This algorithm should eventually determine the actions of the robot, resulting in the routing of the jobs through the system. To do so first the job shop problem and optimization problem will be described.

Problem description

First the Micro Device Fab design will be translated to a job shop scheduling problem. Figure 3.4 shows a schematic representation of the problem. The job shop exists of a group of m machines, which have a capacity of one job. j jobs enter the system from an infinite buffer. Each job follows a predetermined route of machine visits (flow). This order is fixed, but there can be different types of jobs which have different flows. Jobs can only visit a machine one time during a flow. Beforehand, the process times $p(m, j)$ of each process step of job j on machine m are known. The transportation time of the robot will be neglected and the machines do not have setup times.

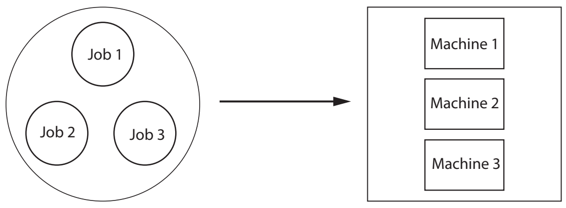

Figure 3.4: Schematic representation of the job shop problem, with $m = 3$ machines and $j = 3$ jobs.

Let C_{max} denote the makespan of all jobs. The starting time of an operation of job j on machine m is denoted as t_{mj}, and p_{mj} denotes the processing time of this operation. The flow of all jobs can be represented using a disjunctive graph G as shown in Figure 3.5. This graph shows the product flows, given by the process steps (m, j). Above each process step, the process time, p_{mj} is indicated. The process steps of a job are connected via conjunctive arcs following the product flow. Node 0 is the source node, from which the first operation of each job starts. The $*$ node is the sink node, in which the final operation of each job ends.

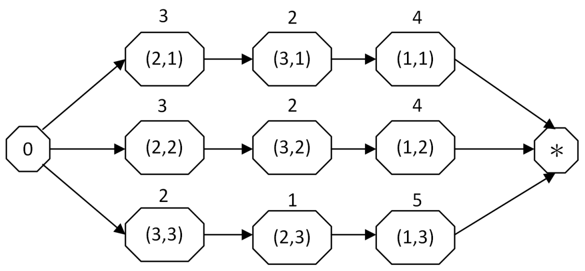

Figure 3.5: Disjunctive graph G of a possible product flow.

N represents the set of all operations (m, j), and the set A is defined as the pairs of operations constrained by process flow or routing, so the arcs in the disjunctive graph between operations. E_k is the set of operations to be performed on machine k. The optimization problem can be formulated as follows [14]:

$$\begin{aligned}
\text{minimize} \quad & C_{max} & \\
\text{subject to} \quad & t_{kj} - t_{mj} \geq p_{mj}, & \forall (m,j) \Rightarrow (k,j) \in A \\
& C_{max} - t_{mj} \geq p_{mj}, & \forall (m,j) \in N \\
& t_{mj} - t_{ml} \geq p_{ml} \text{ or } t_{ml} - t_{mj} \geq p_{mj}, & \forall (m,l) \text{ and} (m,j) \in E_k \\
& t_{mj} \geq 0, & \forall (m,j) \in N
\end{aligned} \tag{3.1}$$

This $J_m||C_{max}$ problem is NP-hard, which means it is very difficult to solve optimally using typical mathematical programming solution techniques due to the "or" condition in the constraints. Therefore, heuristics are determined to find local optima. A common approach that is often used in practice for job shop scheduling, is the use of dispatching rules. This entails that whenever a machine becomes available, a job currently available is selected to be processes on this machine. The advantage of dispatching rules, is easy implementation and short calculation times. However, their inability of considering future effects, may result in a poor long term performance. Such a dispatching rule was also found in Chapter 2 for the current scheduling algorithm. This was found to be a greedy algorithm, which determines an action from the current state of the system following predetermined decision rules. A greedy algorithm with newly determined decision rules will also be considered as an option for the new Micro Device Fab system. Another possibility is using an ant optimization technique for job shop scheduling [8]. This is a combination of the greedy algorithm, which also uses the information of previous routes. The first job (ant) will travel through the system following a greedy algorithm, which will result in good parts (small flow time) and bad parts (long flow time). The good parts will be followed more often, which will eventually converges to good solutions. For the Micro Device Fab, small batches of different types of jobs will be manufactured, and therefore the self learning algorithm will be too slow. There are also heuristics which use the information of the jobs and process times to determine the scheduling beforehand. An example is Johnson's Algorithm, which sorts the process steps of jobs by lowest processing time first [15]. However, for a complex job shop, Johnson's algorithm is too simple (e.g. assuming one job routing) and will therefore not yield good results. Solving a job shop scheduling problem by simulated annealing was proposed by [20]. Simulated annealing is a metaheuristic to approximate global optimization in a large search space. This heuristic considers neighbouring states and probabilistically decides between moving to this state or remaining in the same state, this is repeated until the system reaches a state that is good enough for the application. The simulation annealing in [20] only considers flows of machine permutations, while for the Micro Device Fab, not all machines are always visited and multiple machine visits to one machine are possible. Another heuristic is the shifting bottleneck heuristic, which schedules the machines one by one in order of highest utilization first [2]. In each iteration the scheduling problem is solved for one additional machine until all machines are scheduled. This heuristic can determine schedules for job shops with a large number of machines and jobs with different flows, and therefore seems to be a good match to schedule Micro Device Fab systems. Due to the good performance of the SBH, it has been studied by many [16, 17, 19, 10]. However, the particular job shop problem of the Micro Device Fab with a high product diversity, with multiple machine visits, priority processes, and transportation times, seems to be a new application.

3.2.1 Greedy algorithm

A greedy algorithm is an algorithm that chooses a local optimum following predetermined decision rules. However, this does not always result in a global optimum. The constraints on which the local optimum is found can be different for each problem. The decision rules to find the local optimum are defined as follows:

- If a machine is free and a product has to visit this machine, the product is put into the machine.
- The products are considered by lowest number first.

These decision rules will be applied to a simple example.

Example

The example exists of three machines and three jobs. Table 3.2 shows the machine sequence and process times of the jobs on the different machines.

Jobs	**Machine sequence**	**Processing Times**
1	2, 3, 1	$p_{1,1} = 4$, $p_{2,1} = 3$, $p_{3,1} = 2$
2	2, 3, 1	$p_{1,2} = 4$, $p_{2,2} = 3$, $p_{3,2} = 2$
3	3, 2, 1	$p_{1,3} = 5$, $p_{2,3} = 1$, $p_{3,3} = 2$

Table 3.2: Machine sequence and process times for the three machines, three jobs example

With the defined decision rules, the greedy algorithm results in the time chart as shown in Figure 3.6.

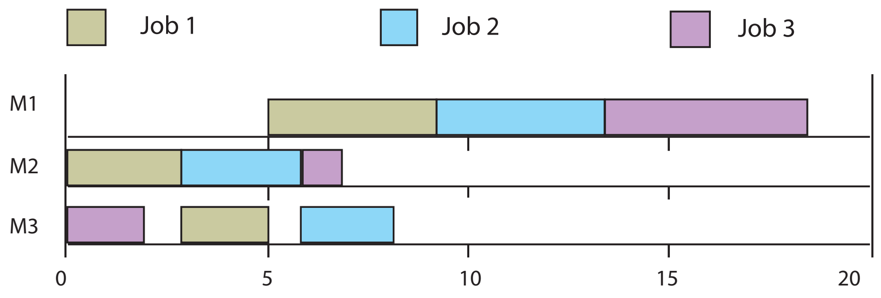

Figure 3.6: Time chart of the example scheduled by the greedy algorithm

The schedule resulting from the greedy algorithm results in a total makespan of 18 time units.

3.2.2 Shifting Bottleneck Heuristic

The shifting bottleneck heuristic is a heuristic that aims at minimizing the total makespan [2], which is relevant for the Micro Device Fab problem. As already mentioned before, the $J_m||C_{max}$ problem is very difficult to solve using mathematical programming solution techniques due to the "or" condition in the constraints. The shifting bottleneck heuristic decomposes the $J_m||C_{max}$ problem into multiple instances of the $1|r_j|L_{max}$ problem [16]. The steps for this approach will be further explained. To do so, let M be the set of all machines and M_0 the set of machines which have already been scheduled. Now the following steps are followed [10]:

Step 1:	Set $M_0 = \emptyset$
Step 2:	Analyze each subproblem associated with the unscheduled machines $k \in M \setminus M_0$.
Step 3:	Identify the critical or bottleneck machine m from the set of unscheduled machines.
Step 4:	Schedule the jobs on machine m by solving the subproblem, set $M_0 := M_0 \cup \{m\}$.
Step 5:	If applicable, reoptimize scheduled machines $k \in M_0$ by identifying and solving its subproblem, taking into account the schedule of the latest scheduled machine. If $M_0 = M$ then stop, else return to Step 2.

The execution of these steps, and the possible decisions variables within these steps will be further explained by applying the shifting bottleneck heuristic to a simple example.

Step 2: Subproblem analysis

The same example as for the Greedy algorithm in Section 3.2.1 will be used for the shifting bottleneck heuristic. First a disjunctive graph G, which was explained is Section 3.2, is constructed as shown in Figure 3.7.

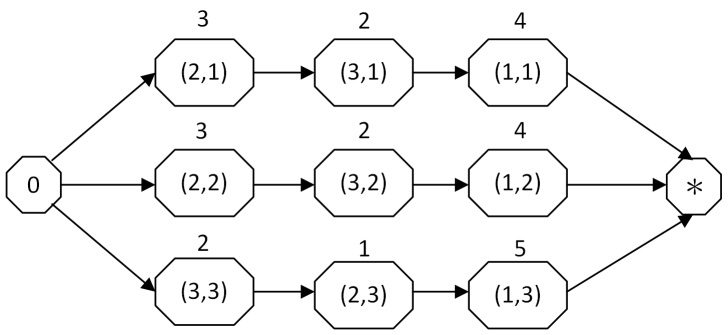

Figure 3.7: Disjunctive graph of the example in Table 3.2.

The interaction between subproblems can be captured by determining the release times r_{mj} and due dates d_{mj} of each process step (m, j) by using longest path calculations in G. The variables that need to be determined are:

LB	:=	lower bound, the current lowest possible value of the total makespan, which is the longest path in G from source node 0 to sink node $*$.
r_{mj}	:=	release time, the earliest time process (m, j) can be started, which is the longest path in G from source node 0 to process (m, j).
d_{mj}	:=	due date, the latest time process (m, j) has to be finished to not increase the makespan, determined by subtracting the longest path in G from process (m, j) to $*$ from the makespan.
CT_{mj}	:=	completion time, time that a process step is completed.
$Tard_{mj}$	:=	tardiness, the amount that the lower bound increases.

The completion time and tardiness are determined as follows:

$$CT_{mj} = max(r_{mj}, CTmj_{prev}) + p_{mj} \tag{3.2}$$

$$Tard_{mj} = CT_{mj} - d_{mj} \tag{3.3}$$

The completion time results from starting time plus the process time. The starting time is either the release time of the job or the completion time of the previously scheduled job on this machine. When the tardiness results in a positive value the due date is passed, and the lower bound increases.

Step 3: Bottleneck selection

Different types of bottleneck machine selection criteria can be used, two possible criteria will be further explained. The first is solving the single machine sub problem (machine scheduling) for all non scheduled machines, the machine that results in the largest tardiness will be chosen as bottleneck machine. This selection criteria will be used for the eventual more complicated use cases. The second option is choosing the machine with the largest total operation process time as the bottleneck machine. This selection criteria is a less precise measure but quick to determine, and will therefore be used for this simple example. This results in the following total operation process time for each machine:

$$M_1 = 13,\ M_2 = 7,\ M_3 = 6 \tag{3.4}$$

This results in a $LB = 13$, for bottleneck machine M_1, and therefore M_1 will be scheduled first. The decision variables of the jobs on this machine are firstly determined and are shown in Table 3.3.

$\boldsymbol{(1, j)}$	$\boldsymbol{J_1}$	$\boldsymbol{J_2}$	$\boldsymbol{J_3}$
$\boldsymbol{p_{1j}}$	4	4	5
$\boldsymbol{r_{1j}}$	5	5	3
$\boldsymbol{d_{1j}}$	13	13	13

Table 3.3: Process time, release time and due date for the jobs on machine 1.

Step 4: Schedule the bottleneck machine

Now the machine subproblem has to be solved to find an "as good as possible" schedule, this can be realized by using a branch and bound method. This method works as follows: create all possible schedules that start with a different job, complete these schedules with the remaining jobs and order these following a predetermined criteria. The starting job which creates the schedule with the lowest tardiness will be set, afterwards, all possible jobs for the second step will be explored and so on. This is continued until all jobs are set and the resulting schedule is the solution of the machine subproblem. A possible criterium to order the unscheduled jobs is in order of earliest due date first. This will also be applied to the simple example. The first three schedules that will be tested start with job 1, 2 and 3 followed by the remaining jobs in order of earliest due date first. This results in the order: $(1,2,3)$, $(2,1,3)$, and $(3,1,2)$, the results are listed in Table 3.4.

Job	J_1	J_2	J_3	**Job**	J_2	J_1	J_3	**Job**	J_3	J_1	J_2
CT_{1j}	9	13	18	CT_{1j}	9	13	18	CT_{1j}	8	12	16
d_{1j}	13	13	13	d_{1j}	13	13	13	d_{1j}	13	13	13
$Tard_{1j}$	0	0	5	$Tard_{1j}$	0	0	5	$Tard_{1j}$	0	0	3

Table 3.4: Completion time, due date and tardiness for the jobs on machine 1.

From these three schedules it can be concluded that a schedule starting with job 3 results in the lowest tardiness. So the third job will be fixed in the schedule, and the remaining possible schedules are created: $(3,1,2)$ and $(3,2,1)$. Both schedules result in a tardiness of 3, when two schedules result in the same tardiness, the schedule with the lowest job number is chosen, therefore the final schedule is $(3,1,2)$. Now the graph in Figure 3.7 can be updated with the schedule of M_1 this results in the disjunctive graph shown in Figure 3.8.

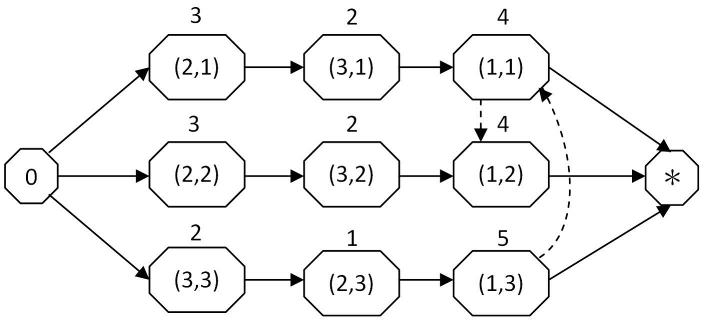

Figure 3.8: Disjunctive graph of the example in Table 3.2 with the schedule of M_1

The schedule of M_1 resulted in a tardiness of 3, which increases the lower bound from $LB = 13$ to $LB = 16$. The next bottleneck machine that will be scheduled is M_2. Similar as for M_1, the release times and due dates of the jobs on machine two will be determined and are shown in table 3.5.

$(2,j)$	J_1	J_2	J_3
p_{2j}	3	3	1
r_{2j}	0	0	2
d_{2j}	6	10	3

Table 3.5: Process time, release time and due date for the jobs on machine 2.

Again three possible schedules are created and with these variables the tardiness of the different schedules can be determined, which is shown in table 3.6.

Job	J_1	J_3	J_2	**Job**	J_2	J_3	J_1	**Job**	J_3	J_1	J_2
$\boldsymbol{CT_{2j}}$	3	4	7	$\boldsymbol{CT_{2j}}$	3	4	7	$\boldsymbol{CT_{2j}}$	3	6	9
$\boldsymbol{d_{2j}}$	6	3	10	$\boldsymbol{d_{2j}}$	10	3	6	$\boldsymbol{d_{2j}}$	3	6	10
$\boldsymbol{Tard_{2j}}$	0	1	0	$\boldsymbol{Tard_{2j}}$	0	1	1	$\boldsymbol{Tard_{2j}}$	0	0	0

Table 3.6: Completion time, due date and tardiness for the jobs on machine 2.

Because there is a schedule which results in a tardiness of zero, there can not be a better solution. Therefore, schedule $(3, 1, 2)$ is chosen for machine two, which will not increase the lower bound. With this schedule the disjunctive graph is again updated, and shown in Figure 3.9.

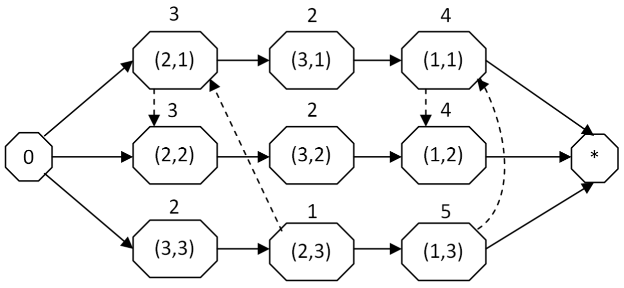

Figure 3.9: Disjunctive graph of the example in Table 3.2 with the schedule of M_1 and M_2.

Finally, the schedule for machine three has to be determined. The lower bound is still $LB = 16$ and the release times and due dates of the jobs are shown in table 3.7.

$\boldsymbol{(3, j)}$	J_1	J_2	J_3
$\boldsymbol{p_{3j}}$	2	2	2
$\boldsymbol{r_{3j}}$	6	9	0
$\boldsymbol{d_{3j}}$	8	12	2

Table 3.7: Process time, release time and due date for the jobs on machine 2.

Again three possible schedules are created and with these variables the tardiness of the different schedules can be determined, which is shown in table 3.8.

Job	J_1	J_3	J_2	**Job**	J_2	J_3	J_1	**Job**	J_3	J_1	J_2
$\boldsymbol{CT_{3j}}$	8	10	12	$\boldsymbol{CT_{3j}}$	11	13	15	$\boldsymbol{CT_{3j}}$	2	8	11
$\boldsymbol{d_{3j}}$	8	2	12	$\boldsymbol{d_{3j}}$	12	2	8	$\boldsymbol{d_{3j}}$	2	8	12
$\boldsymbol{Tard_{3j}}$	0	8	0	$\boldsymbol{Tard_{3j}}$	0	11	7	$\boldsymbol{Tard_{3j}}$	0	0	0

Table 3.8: Completion time, due date and tardiness for the jobs on machine 3.

Again a schedule can be found which results in a tardiness of zero and therefore the schedule for machine three becomes $(3, 1, 2)$. This results in the final disjunctive graph shown in Figure 3.10.

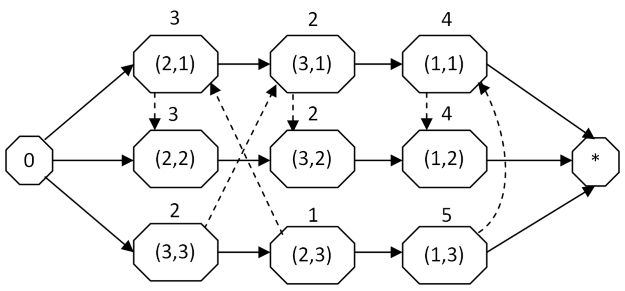

Figure 3.10: Disjunctive graph of the example in Table 3.2 with the schedule of M_1, M_2, and M_3.

This results in a job shop scheduling with a makespan of 16, which is 2 time units shorter than for the greedy algorithm. With the release times, process times, and machine schedules a time chart as shown in Figure 3.11 can be generated.

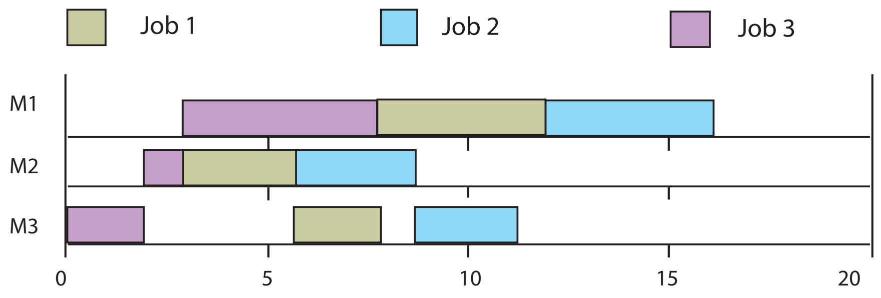

Figure 3.11: Time chart of the schedule derived from shifting bottleneck heuristic.

Step 5: Reoptimization

In the simple example no reoptimization was applied. However, this can improve the schedule of the shifting bottleneck heuristic, especially for the scheduling of a large number of machines. Reoptimization works as follows: when an additional machine is scheduled, all the previously scheduled machines in M_0 are rescheduled. The schedule of the machine from M_0 that is being rescheduled is removed, and a new schedule is computed taking into consideration the additional machine and the other machines in M_0. The new found schedule is than set, and the next machine in M_0 is rescheduled. This is done in order of the list M_0, which is in order of bottleneck machine first.

3.3 System adjustments

The job shop problem described in Section 3.2 is a simple version of the real system. Now the job shop problem will be expanded with a dual armed robot, job transportation times, multiple visits of a job to a machine, and it should be possible to apply priority to a process step. There is only one buffer, which is still assumed to be infinite and jobs can be put into this buffer between process steps. The job shop problem consists of 4 jobs with multiple machine visits, a dual armed robot with transportation times, and 3 machines. This results in an updated schematic representation of the problem shown in Figure 3.12.

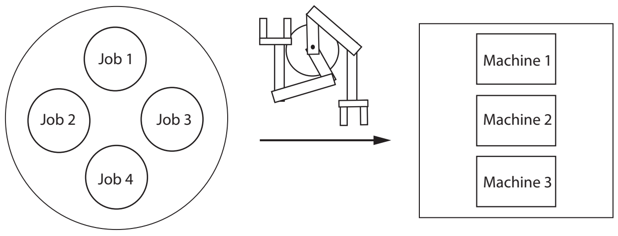

Figure 3.12: Schematic representation of the expanded job shop problem.

Due to the possibility of having multiple machine visits. Process steps are now labeled as: (m, j, n), with m the machine, j the job, n the number of times the job has visited the machine. Table 3.9 shows the flows and process times of the 4 job on the 3 machines.

Jobs	**Machine sequence**	**Processing Times**
1	1, 2, 3	$p_{1,1,1} = 50$, $p_{2,1,1} = 150$, $p_{3,1,1} = 50$
2	1, 2, 3	$p_{1,2,1} = 50$, $p_{2,2,1} = 150$, $p_{3,2,1} = 50$
3	2, 3, 2, 1	$p_{2,3,1} = 50$, $p_{3,3,1} = 150$, $p_{2,3,2} = 100$, $p_{1,3,1} = 150$
4	1, 3, 1, 2	$p_{1,4,1} = 50$, $p_{3,4,1} = 150$, $p_{1,4,2} = 150$, $p_{2,4,1} = 100$

Table 3.9: Job flow and process time for the 3 machines, 4 jobs with multiple machine visits example

The previously described heuristics, greedy algorithm and shifting bottleneck heuristic, are modified and optimized for solving the expanded job shop problem. These adjustments will be further explained in this section.

3.3.1 Greedy algorithm

For the expanded job shop problem new decision rules for the greedy algorithm will be defined. These decision rules are especially made for a dual armed robot, and are designed to better perform for this specific job shop problem. As determined in Section 3.1 the Micro Device Fab will be equipped with a dual armed robot. There is one buffer, which is assumed to be infinite, with a predefined number of jobs the flows of which are known. Below, the decision rules are defined, which depend on the number of jobs the robot is carrying.

If the robot is carrying no jobs:

- If there is a job waiting in a machine which is finished, retrieve this job.
- Else, if there is a job waiting in the buffer of which the next machine is available or finished, retrieve this job.

If the robot is carrying one job:

- If the next machine of the carrying job is available, move this job to the machine.
- Else, if the next machine for the carrying job is occupied, but the process in that machine is finished, retrieve the finished job inside of the machine.
- Else, if there is a job finished and waiting in a machine, retrieve this job.
- Else, if there is a job waiting in the buffer of which the next machine is available, retrieve this job.

If the robot is carrying two jobs:

- If the next machine of one of the carrying jobs is available, move that job to its next machine.
- Else, if for one of the carrying jobs, the job in the next machine is finished, then the other carrying job is put in the buffer.
- Else, the first job the robot is carrying is put in the buffer.

To demonstrate the greedy algorithm with the redefined decision rules, the heuristic is applied to the job shop problem as shown in Table 3.9. This is done by implementing these rules in a simulation model, which consists of the 3 machines, 4 jobs, and a dual armed robot as explained in the expanded job shop problem. Figure 3.13 shows the time chart that results from the simulation model with the greedy algorithm. The transportation times of the dual armed robot are incorporated here. For this example it is assumed that the buffer is the home position of the robot, to travel from the buffer to machine 1 is 0 time units (at the same location), from the buffer to machine 2 is 5 time units, and from the buffer to machine 3 is 10 time units. The time to travel between these machines, is the difference between their transportation time. When the robot has arrived at the machine, the job has to be put into or removed from the machine, which takes around 8 time units. Due to the system adjustments of Figure 3.12, which adds the dual armed robot with transportation times, jobs are not immediately moved from the machine as was for the simple problem of Figure 3.4. Therefore, a grey bar is shown when the process is already finished.

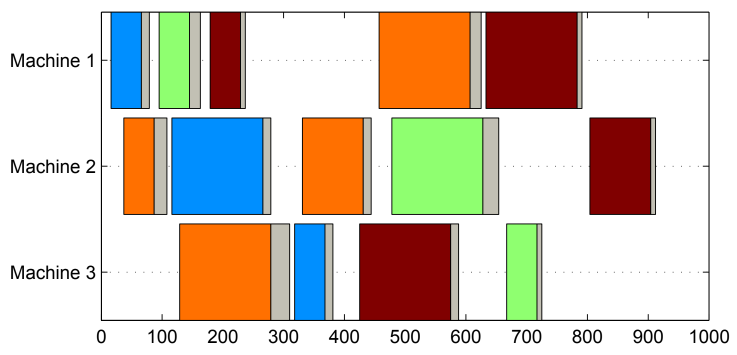

Figure 3.13: Time chart of the greedy algorithm from a simulation model with the system adjustments, applied to the job shop problem of Table 3.9.

Figure 3.13 shows the flows of the different jobs on the machines, when scheduled by the greedy algorithm. The time between processes is increased due to the transportation times of the robot. For this problem the greedy algorithm achieves a makespan of 925, and an utilization of around 60% for the bottleneck machine 2.

Another system adjustment that has to be taken into account is the ability of applying priority to process steps. When a process step with priority is finished, this job should be retrieved as quickly as possible. This means that the grey bar as shown in Figure 3.13 for a priority process has to be as short as possible. To do so, after a priority step is finished, the job is added to a priority list. This list is ordered by longest waiting time first. To apply priority, the decision rules should be slightly adjusted. These additional priority rules always go before the standard rules, if the priority list is non empty:

If the robot is carrying no jobs:

- Retrieve the first job on the priority list from its machine.

If the robot is carrying one job:

- Retrieve the first job on the priority list from its machine.

If the robot is carrying two jobs:

- If the next machine of one of the carrying jobs is available, move that job to its next machine.
- Else, if for one of the carrying jobs, the job in the next machine is finished, then the other carrying job is put in the buffer.
- Else, the first job the robot is carrying is put in the buffer.

If the priority list is empty, the standard rules formulated at the beginning of this section are followed. The result of the implementation of these priority rules will be demonstrated by applying priority to the second process step of job 2. The time charts of the simulation results for the problem without and with priority are compared in Figure 3.14.

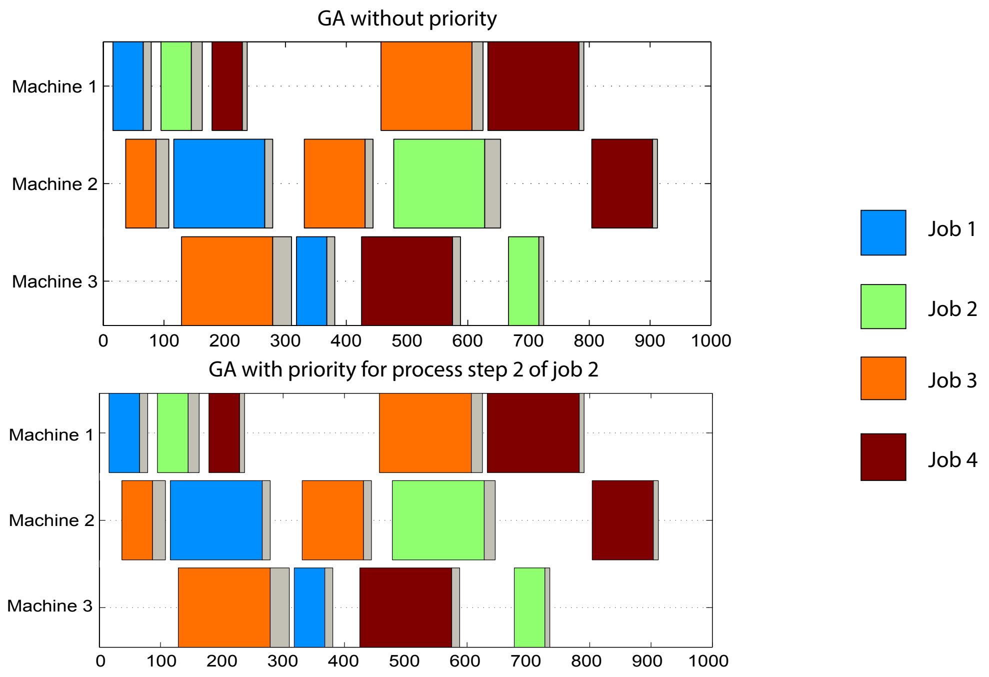

Figure 3.14: Top time Chart, the greedy algorithm without priority, lower time chart, with priority for process step 2 of job 3, for the job shop problem of Table 3.9.

Figure 3.14 shows that the grey bar of the second process step of job 2 is slightly smaller when the process step is given priority. Because the job shop problem only consists of 3 machines and 4 jobs, the workload of the robot is low. Therefore no long waiting periods are found, and only minor improvements of the waiting time for the priority process is shown. Now this greedy algorithm is ready to be applied to use case simulations.

3.3.2 Shifting bottleneck heuristic

The scheduling problem encountered in the Micro Device Fab differs from the standard problem solved by the shifting bottleneck heuristic, as shown in the beginning of this section. Related work in the literature proposes modifications to the shifting bottleneck heuristic for complex job shops. However, this research mostly focusses on re-entrant product flows and batching machines [16], or setup times and prescribed customer due dates [17]. A similar job shop problem with product flows with multiple machine visits has been investigated in [19], however they do not incorporate priority processes and transportation times. To explain the adjustments of the shifting bottleneck heuristic, the more complicated job shop problem of Table 3.9 will be used.

The changes that are added to the Shifting Bottleneck heuristic to cope with the system adjustments and to improve the performance of the shifting bottleneck heuristic, will be discussed one by one. These adjustment are separately applied to the standard Shifting Bottleneck heuristic. The first adjustment improves the performance of more complicated product flows with multiple machine visits by redetermining the release time.Afterwards an adjustment that optimizes the order in which schedules are created is proposed. Followed by a decision rule adjustment to reduce the time between process steps of a job, this decision rule is applied when multiple schedules with the same makespan are found. Another adjustment for the Micro Device Fab, is the possibility of applying priority to a process step, which means that when this process step with priority is finished, the job should be retrieved as quickly as possible. This priority rule is also incorporated by creating a decision rule for when multiple schedules with the same makespan are found. The final adjustment is to take the transportation times into account. These adjustments will be further discussed in more detail.

Multiple machine visits and Release time update

In the Micro Device Fab it is possible that products visit one machine multiple times during a flow. The implementation of multiple machine visits in the shifting bottleneck heuristic will be explained via the job shop example introduced in Table 3.9. To implement this into the shifting bottleneck heuristic the process steps need to be defined as: (m, j, n), with m the machine, j the job, and n the number of times the job has visited the machine. This adjustment is necessary such that the different processes of a job on one machine can be identified, because the order of these processes is fixed and must be met. The disjunctive graph for this job shop problem is shown in Figure 3.15.

To schedule this flow, the bottleneck machine will be scheduled first, which is the machine with the largest total operation process time. This results in bottleneck machine 2 with a lower bound of $LB = 550$. Similar as for the shifting bottleneck heuristic of the previous section for single machine visits, the release time, and due dates for each process are determined and are shown in Table 3.10.

$(2, j, n)$	$J_{2,1,1}$	$J_{2,2,1}$	$J_{2,3,1}$	$J_{2,4,1}$	$J_{2,3,2}$
p_{2jn}	150	150	50	100	100
r_{2jn}	50	50	0	350	200
d_{2jn}	500	500	150	550	400

Table 3.10: Process times, release times and due dates for the jobs on machine 2.

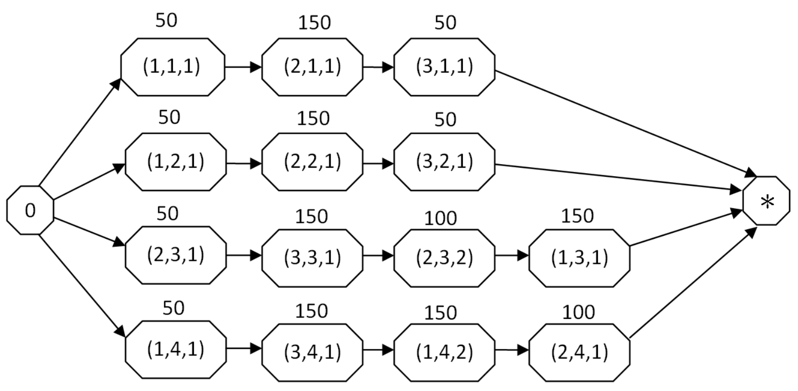

Figure 3.15: Disjunctive graph of product flows with multiple machine visits.

Now the completion time and tardiness can be determined for the different schedules. These schedules are created by setting one job, and the other jobs follow by order of earliest due date first as explained in Section 3.2.2. Table 3.11 shows the values for the schedules with set job 1: $(1,3,3,2,4)$, and for set job 3: $(3,3,1,2,4)$.

Job	$J_{2,1,1}$	$J_{2,3,1}$	$J_{2,3,2}$	$J_{2,2,1}$	$J_{2,4,1}$	**Job**	$J_{2,3,1}$	$J_{2,3,2}$	$J_{2,1,1}$	$J_{2,2,1}$	$J_{2,4,1}$
p_{2jn}	150	50	100	150	100	p_{2jn}	50	100	150	150	100
r_{2jn}	50	0	200	50	350	r_{2jn}	0	200	50	50	350
CT_{2jn}	200	250	350	500	600	CT_{2jn}	50	300	450	600	700
d_{2jn}	500	150	400	500	550	d_{2jn}	150	400	500	500	550
$Tard_{2jn}$	0	100	0	0	50	$Tard_{2jn}$	0	0	0	100	150

Table 3.11: Completion times, and tardiness for two schedules on machine 2.

The schedule $(1,3,3,2,4)$ results in a tardiness of 100 time units, and the schedule $(3,3,1,2,4)$ results in a tardiness of 150 time units. From this, the conclusion could be drawn that the optimal schedule starts with job 1. However, this is not the case, because if a job visits a machine multiple times, the release time of the second visit is dependent on the completion time of the previous visit. This is due to the fixed flow, after the first process step of job 3 on machine 2 is finished ($J_{2,3,1}$), the job first has to visit machine 3, before it can return to machine 2 ($J_{2,3,2}$). For the first schedule, the completion time of $J_{2,3,1}$ is 250, and the process time of the next process step on machine 3 is 150, this would mean that the earliest time that process $J_{2,3,2}$ can be released is 400. In Section 3.2.2 this problem was not encountered, because jobs could only visit a machine once. Therefore, the release time of processes of which the job has already visited the same machine should be redetermined when scheduled. This property for multiple machine visits was also identified in [19]. As explained, this results in an increased release time for job $J_{2,3,2}$ from 200 to 400. This adjustment results in the updated completion times and resulting tardiness as shown in Table 3.12.
By updating the release time, schedule $(1,3,3,2,4)$ now results in a tardiness of 200 time units. From this a new conclusion can be drawn, which is that the optimal schedule should start with product 3 instead of product 1.

However, the change of release time due to the previously scheduled processes can also occur due

Job	$J_{2,1,1}$	$J_{2,3,1}$	$J_{2,3,2}$	$J_{2,2,1}$	$J_{2,4,1}$	Job	$J_{2,3,1}$	$J_{2,3,2}$	$J_{2,1,1}$	$J_{2,2,1}$	$J_{2,4,1}$
p_{2jn}	150	50	100	150	100	p_{2jn}	50	100	150	150	100
r_{2jn}	50	0	400	50	350	r_{2jn}	0	200	50	50	350
CT_{2jn}	200	250	500	650	750	CT_{2jn}	50	300	450	600	700
d_{2jn}	500	150	400	500	550	d_{2jn}	150	400	500	500	550
$Tard_{2jn}$	0	100	100	150	200	$Tard_{2jn}$	0	0	0	100	150

Table 3.12: Completion times, and tardiness for two schedules with updated release times on machine 2.

to constraints of the already scheduled machines. This will be explained based on a simple example with 3 jobs and 2 machines shown in Figure 3.16, with the flow of the jobs shown in the legend. If machine 1 is already scheduled with the following job order: $(1, 2, 3)$, as shown in the time chart in Figure 3.16.

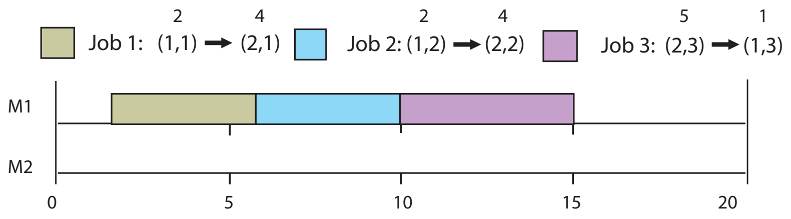

Figure 3.16: Time chart of 3 job flows for 2 machines with machine 1 scheduled.

The release time of job 1 and 2 on machine 2 is zero, and the release time of job 3 on machine 2 is 15. However, if the processes on machine 2 would be schedule in the following order: $(2, 1, 3)$ the release time of job 3 would increase to 17, due to the shift of the processes on machine 1, as shown in Figure 3.17.

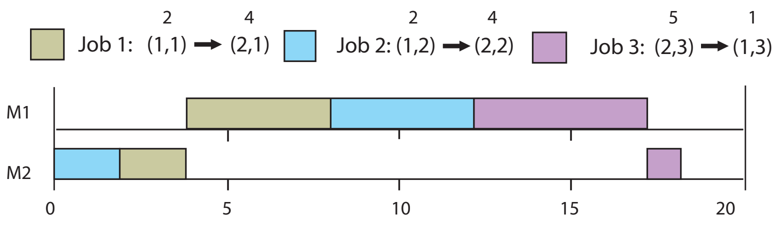

Figure 3.17: Time chart of 3 job flows for 2 machines with machine 1 and 2 scheduled.

These two schedules would never result after solving the one machine subproblems, because schedule $(3, 1, 2)$ for machine 1, and schedule $(1, 2, 3)$ for machine 2, would result in a lower tardiness. However, for more complicated job shop problems it can occur that due to an incorrect release time the wrong schedule is chosen. These shifts in the release times can be neglected, but could possibly decrease the performance of the heuristic, because a less constrained problem is solved [3]. If we do want to take these shifts into account there are two options. Either delayed precedence constraints can be defined [9], which takes all constraints that could effect the release times into account. Or the release time, which is the longest path from the source node to the process, is recalculated when a process is scheduled. The latter solution is chosen. This means that the conjunctive arcs of the previously scheduled processes on a machine (dashed arcs as in Figure 3.8), are already taken into account for determining the release time of the following process, and so on. The due date of a process is not affected by the previously scheduled processes, because the due date is dependent on the longest path of the process to the sink node.

To show the effect of updating the release times while scheduling, both the schedules resulting from the SBH without updated release time, and with the updated release time are shown in Figure 3.18 for the multiple machine visit example of Table 3.9.

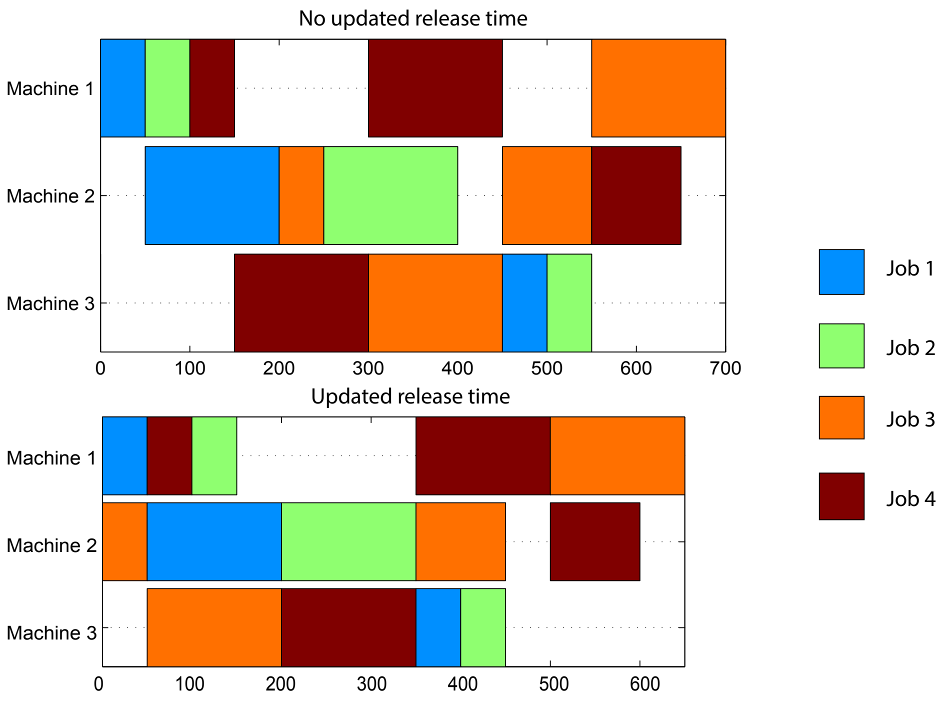

Figure 3.18: Upper time chart shows the SBH schedule without updating the release time, and the lower time chart shows the SBH schedule which does update the release time for the example of Table 3.9.

Figure 3.18 shows that the SBH that does update the release time achieves a makespan of 50 time units shorter and an utilization of around 85% for bottleneck machine 2. While the non adjusted SBH results in an utilization of around 80%. This release time update adjustment is already incorporated in the SBH to which the other upcoming adjustments will be applied.

Scheduling order

As mentioned in Section 3.2.2 the possible schedules for solving the one machine subproblem are created by setting the first job followed by the unscheduled jobs in order of earliest due date first. However, this can sometimes results in a suboptimal solution.
For example, when looking at the created scheduling order $(3, 3, 1, 2, 4)$ in Table 3.11. The due date of process $J_{2,3,2}$ is 400, and therefore earlier then for process $J_{2,1,1}$, which is 500 and thus the

rule of the earliest due date first is followed. However, the release time of process $J_{2,1,1}$ is earlier than for process $J_{2,3,2}$. If process $J_{2,1,1}$ would be scheduled before process $J_{2,3,2}$, it could start at 50 ($\max(CT_{2,3,1}, r_{2,1,1})$), and would be finished at 200. This would not influence the starting time of process $J_{2,3,2}$, because the release time of this process is 200. This schedule $(3, 1, 3, 2, 4)$, would result in a tardiness of zero as shown in Table 3.13.

Job	$J_{2,3,1}$	$J_{2,1,1}$	$J_{2,3,2}$	$J_{2,2,1}$	$J_{2,4,1}$
p_{2jn}	50	150	100	150	100
r_{2jn}	0	50	200	50	350
CT_{2jn}	50	200	300	450	550
d_{2jn}	150	500	400	500	550
$Tard_{2jn}$	0	0	0	0	0

Table 3.13: Completion times, and tardiness for the new scheduling order on machine 2.

Therefore, the criteria for ordering the unscheduled jobs is adjusted as follows; if process a is set, and the order of process b and c has to be determined, process b can be scheduled before process c if:

- the due date of process b is earlier ($d_b < d_c$), and it is impossible for process c to be finished before the release time of process b ($r_b < \max(CT_a, r_c) + p_c$).
- process b can be finished before process c can start ($r_c > \max(CT_a, r_b) + p_b$)

This allows a tighter schedule and therefore a better solution compared to the earliest due date first strategy, as was shown in Table 3.13.

Reduce time between process steps

The handler can not hold a job for a long time, because both arms are needed for job exchanges. Therefore, the scheduling heuristic should try to schedule the process steps of a job as close as possible following each other, to avoid unnecessary handling times. To accomplish this, the time between process steps of a job is used as a decision variable. To measure this, a new variable is created, the quick follow up due date ($QFUd$). This variable is described as follows:

$$QFUd(m, j, n) = r(m_{prev}, j, n_{prev}) + p(m_{prev}, j, n_{prev}) + p(m, j, n) \tag{3.5}$$

with

$QFUd(m, j, n)$	:=	quick follow up due date on machine m, of job j, the nth time the job visits the machine;
$p(m, j, n)$	:=	process time on machine m, of job j, the nth time the job visits the machine;
$r(m_{prev}, j, n_{prev})$	:=	release time of the previous process step of job j, which was on machine m_{prev}, and was the n_{prev}th time the job visited the machine,
$p(m_{prev}, j, n_{prev})$	:=	process time of the previous process step of job j, which was on machine m_{prev}, and was the n_{prev}th time the job visited the machine.

This implies that the quick follow up due date of a process, is equal to the completion time if this process step would be executed directly after the previous process step of its flow. The difference between the actual completion time and the $QFUd$ results in the time between the process steps of a job flow. To reduce the time between process steps, this difference is used as a decision variable. When the bottleneck machine is scheduled, the schedule with the lowest tardiness is always chosen. However, when multiple schedules result in the same lowest tardiness, the schedule with the lowest total time between process steps is chosen. The effect of this rule on the schedule, for the example of Figure 3.15, is shown in Figure 3.19.

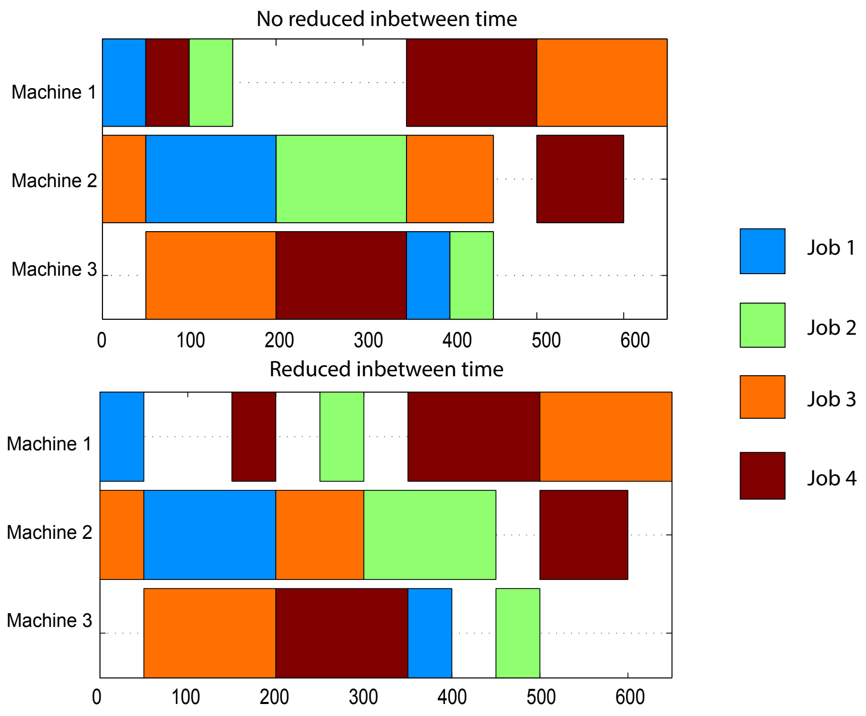

Figure 3.19: Upper time chart shows the schedule of the none reduced in between time SBH and the lower time chart shows the schedule of the adjusted SBH with reduced in between times for the example of Figure 3.15.

Figure 3.19 shows that the adjusted SBH results in a better follow up of job process steps. The sum of the time between all process steps of each job reduces from 550 to 350 time units.

Priority

For certain process steps, jobs can not remain in a machine for a longer time period than the process time. For these process steps applying priority should be possible. When a process step with priority is finished, this job should be retrieved from the machine as quickly as possible. To ensure that the scheduling heuristic assigns this priority, the time between a process step with priority and the next process step of the same job should be added as a scheduling decision rule. Similar as for the reduced time between process steps adjustments. To determine this priority time, different priority types are formulated: if a process step itself has priority, the priority type is 1, if the previous process step has priority, the priority type is 2, and if both the process step itself and the previous process step has priority, the priority type is 3. This results in the following priority time ($PrioT$) definition:

$$
PrioT(m,j,n) = \begin{cases} r(m_{next},j,n_{next}) - CT(m,j,n), & \text{if Type}(m,j,n)=1 \\ CT(m,j,n) - p(m,j,n) - CT(m_{prev},j,n_{prev}) & \text{if Type}(m,j,n)=2 \\ r(m_{next},j,n_{next}) - p(m,j,n) - & \\ CT(m_{prev},j,n_{prev}) & \text{if Type}(m,j,n)=3 \\ 0, & \text{otherwise} \end{cases}
$$

When a process step is of type 1, the process should be finished at the time the next process step starts, to have a direct follow up. If the process step is of type 2, the starting time of this process step (CT minus p) should be equal to the finishing time of the previous priority process step. If a process step is of type 3, both requirements of type 1 and 2 should by satisfied. The sum of the $PrioT$ of all priority process steps of each job results in the total priority time and is also used as a decision variable. When the bottleneck machine is scheduled, the schedule with the lowest tardiness is always chosen, however, when multiple schedules result in the same lowest tardiness the schedule with the lowest time between priority process steps is chosen. These adjustments result in the ability of taking priority of process steps into account. Figure 3.20 shows the schedule without priority and the schedule when priority is assigned to process step 2 of job 1 for the example of Figure 3.15.

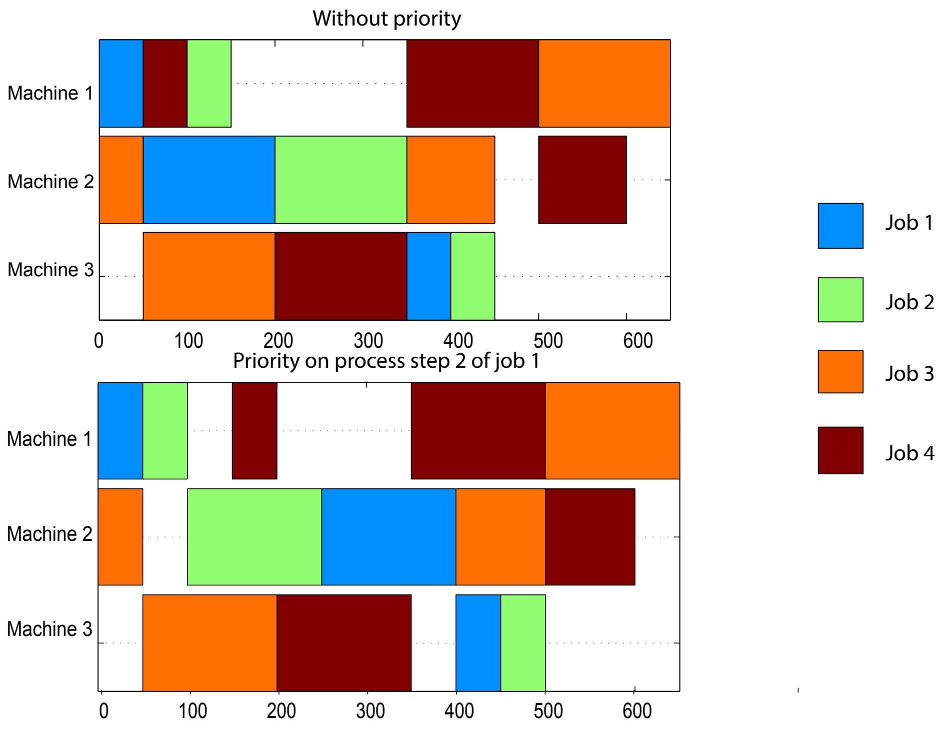

Figure 3.20: Upper time chart shows the schedule of the SBH without priority, and the lower time chart shows the schedule of the SBH with priority on process step 2 of job 1 for the example of Figure 3.15.

Figure 3.20 shows that process step 2 of job 1 is given priority, without increasing the makespan. If applying priority would increase the makespan, the priority will not be applied, because for our optimization problem as mentioned in Section 3.2 the makespan has to be minimized. Besides that, there is a possibility of jobs being moved to a temporary buffer as mentioned in Section 3.1, this would allow a job after a priority process to be moved to the temporary buffer when the next process does not directly follow up. However, this does increase the transportation time. The option of moving a job to the temporary buffer and taking the transportation time into account will be further discussed in the next section.

Transportation times

In reality the machines are spread across the robot track, and the transportation time of the robot depends on the schedule. First the robot has to move to the location of the job and afterwards the job has to be moved to the next location. The real life transportation time of a location is given by the time the robot has to travel from the homing position to this location, as mentioned in Section 3.3.1. For this example it is assumed that the buffer is at the home position of the robot, to travel from the buffer to machine 1 is 0 time units (at the same location), from the buffer to machine 2 is 5 time units, and from the buffer to machine 3 is 10 time units. The time to travel between these machines, is the difference between their transportation time. When the robot arrives at a machine, the job has to be put into or removed from the machine, which takes around 8 time units. The transportation time between the process steps of a job are known, however, while scheduling the machines it is not known if these process steps follow up on each other. Besides that, a lot of different scenarios are possible in the real life system, a job can be moved to the temporary buffer, and the robot can be carrying zero, one or two jobs, which results in different transportation times. This exact behaviour is very difficult to predict, even though the real life system follows the machine schedules of the shifting bottleneck heuristic, how these jobs get at their location is however determined by robot decision rules. These decision rules follow the job list created by the schedules of the shifting bottleneck heuristic. This job list represents the order in which the jobs have to be moved to the next machine. The job that has to be moved, is called the scheduled job. When the scheduled job is moved to the next process, the next job on the job list becomes the scheduled job. The same priority list as for the greedy algorithm is created, to ensure that products are quickly retrieved after a priority process step. The robot decision rules are as follows:

- If the priority list is larger then zero, retrieve the first job.
- If the scheduled job is in the buffer, retrieve this job.
- Else, if the scheduled job is being carried:
  - if the next machine is available, put this job in the machine.
  - if the next machine is occupied, and the robot has one arm free, retrieve the job that is in the machine.
- Else, if the scheduled job is not being carried:
  - if the job list is not finished, and the carrying job is not the scheduled job, put the carrying job to the buffer.
  - if the job list is finished, put the carrying job to the buffer.
  - if the robot has one arm free, and the scheduled job is finished in a machine, retrieve the scheduled job from its machine.
  - if the robot has one arm free, and the final process of a job is finished, retrieve the job

Due to the difficult nature of this transportation time, an average transportation time is added to each process step. By comparing the schedules of the SBH without and with transportation times, the importance of taking the transportation times into account can be shown. To do so, a new scheduling problem is defined. This new problem is defined because the difference in schedule occurs when one machine has to handle more processes than the other machines, and then becomes the bottleneck machine when transportation times are added to these processes. The disjunctive graph of the 2 machine, 4 jobs scheduling problem is shown in Figure 3.21.

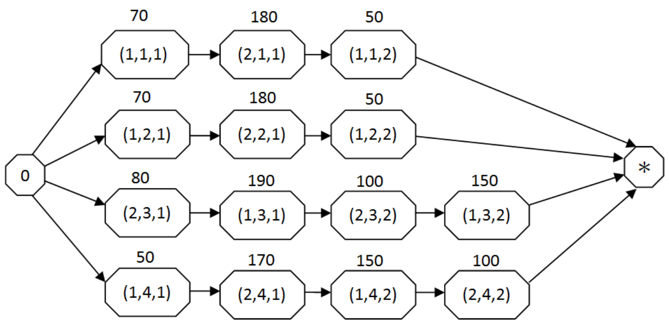

Figure 3.21: Disjunctive graph of the 2 machine, 4 jobs scheduling problem.

When no transportation times are added to this problem, machine 2 is the bottleneck machine with a total process time of 810, followed by machine 1 with a total process time of 780. The transportation time to the buffer is 0, to machine 1 is 5, and to machine 2 is 10. This results in an average transportation time to a location of 5 and the time of putting or extracting a job from a machine is 8. The sum of these values is added twice to the transportation time, because a job has to be moved before and after a process. With a transportation time of 26 and an average process time of around 115, the transportation time has a large effect on the makespan, which simplifies the demonstration of taking the process times into account. However, for real life systems the process times are larger, which will be shown in the next chapter. When a transportation time of 26 is added to all process steps, machine 1 becomes the bottleneck machine with a total process time of 988, followed by machine 2 with a total process time of 966. Due to the different order of machine scheduling, the resulting schedules are different, both time charts are shown in Figure 3.22.

When both schedules are used as input for the simulation model, the difference in performance can be measured. A simulation model is created with the 2 machines, 4 jobs problem and transportation times as mentioned above. From the schedule determined by the shifting bottleneck heuristic a product list is created, which represents the order in which the products have to be moved to the next machine. For the movements of the robot the decision rules as mentioned above will be used. Figure 3.23 shows the results of the simulation model for the schedule without and with transportation times.The makespan of the schedule that neglects the transportation time results in a makespan of 1130, while the schedule that did include the transportation time results in a makespan of 1103. This shows that taking the transportation times into account does result in a better solution. When the results of Figure 3.23 are compared to the results of Figure 3.22, there can be seen that the expected makespan of the schedule with transportation times is almost equal to the result of the simulation model. This shows that the approximation of adding an average transportation to the process times seems reasonable for the determination of schedules, and makespan prediction. By adding all these system adjustments to the SBH, and using reoptimization in step 5 of the heuristic as mentioned in Section 3.2.2. The heuristic is complete and ready to be applied to use case simulations.

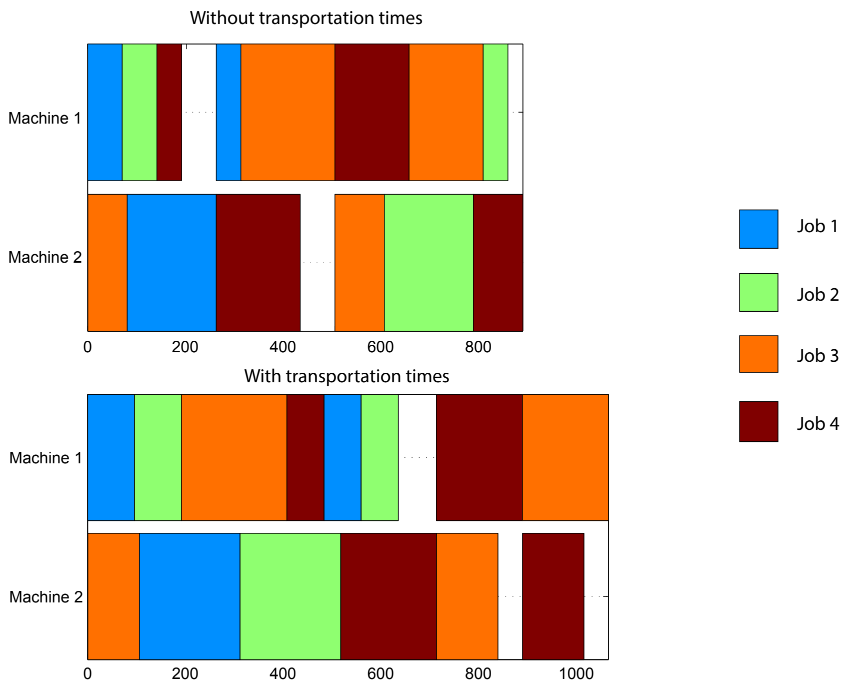

Figure 3.22: Time charts of the scheduling problem of Figure 3.21 with and without transportation times added to the process times.

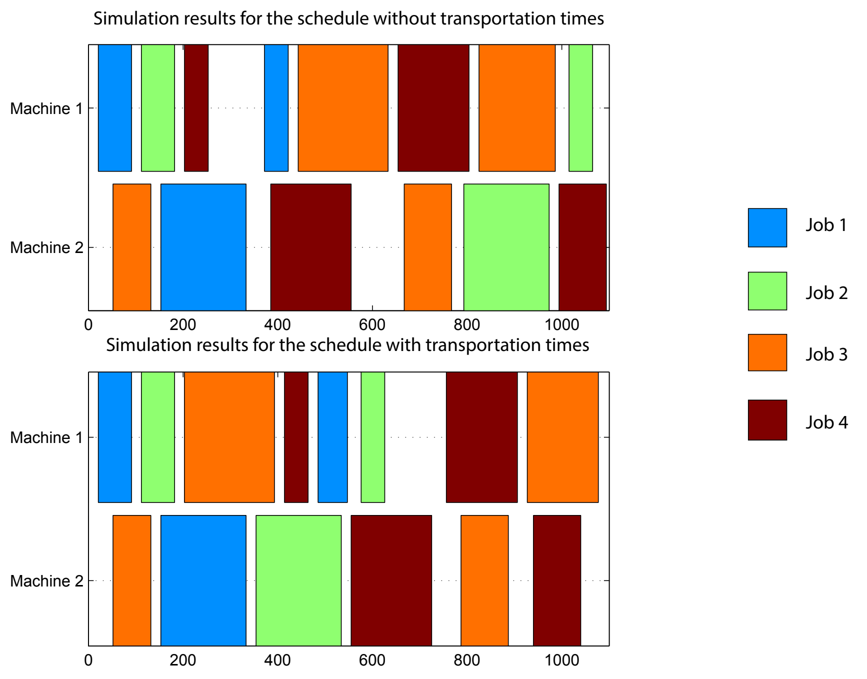

Figure 3.23: Time charts of the simulated scheduling problem of Figure 3.21 with and without transportation times added to the process times.

Chapter 4

Results

To determine the performance of the system design rules formulated in the previous chapter, these will be implemented into use cases. First a general use case will be formulated to test the performance of the heuristics for a large number of different flows. Afterwards, real life use case for two possible Micro Device Fab systems will be simulated and the performance and robustness of these systems will be tested.

4.1 General use cases

The performance of the greedy algorithm, and the shifting bottleneck heuristic will be compared by solving a general use case. An imaginary scheduling problem is formulated. This problem consists of 3 types of products and 5 machines, the process times of each process step are generated by a uniform distribution between 50 and 500 time units. The product flow of the three types of products are defined as follows:

Type 1	:	$3, 4, 5, 2, 4$
Type 2	:	$1, 3, 4, 5, 2, 5$
Type 3	:	$4, 2, 1, 3, 1$

Priority is applied to all processes on machine 3. Ten different sets of process times for all process steps are created, to which both heuristics are applied for different experiments. The transportation times are as follows: 0 seconds to the carrier, 2 seconds to machine 1, 2 seconds to machine 2, 4 seconds to machine 3, 4 seconds to machine 4, and 6 seconds to machine 5, the time to put or grab a product takes 8 seconds. For the shifting bottleneck heuristic, this results in an average transportation time of 22 seconds that will be added to all process times.

The first experiment is for the scheduling of 3 products of each type. This results in an average difference of the makespan between the SBH and GA of 14%, which is shown in Table 4.1. We note that 4% improvement resulted from the reduced time between process steps adjustment as mentioned in Section 3.3.2. However, the flow time of the jobs is 5% higher for the SBH. The yield, which is the amount of products for which the priority requirement is met, is around 8% lower for the SBH. When this experiment is increased to 10 products of each type, the performance of the SBH is only 10% better. This can be explained, due to the difference between the real transportation time and the estimated transportation time which is added to each process time for the scheduling of the SBH as explained in Section 3.3.2, the further the horizon the larger this deviation becomes. Another experiment that is conducted, consists of only products of one type, when 9 products of type 2 are simulated, this results in only a 1 % better makespan for the SBH. When 30 products of one type are simulated, there is no difference in makespan between the two heuristics. When there is only one type of product flow, there is no possibility for smart scheduling of different product types together, and therefore the SBH loses its advantage. In the final experiment the process times are rounded to the nearest hundred for the 3 products of each type use case. This resulted in a 10% better performance, which is 4% less than for the non rounded process times. When the process times are rounded, this results in a higher common denominator for the process times, and thus a larger chance of process steps accidentally following up on each other, which increases the performance of the GA.

Experiment	**Performance measure**	**Difference GA and SBH**
	Makespan (%)	14,0
3 **products of each type**	Flowtime (%)	-5,6
	Yield (%)	-7.8
	Makespan (%)	9,5
10 **products of each type**	Flowtime (%)	-19,0
	Yield (%)	-7,6
	Makespan(%)	0,9
9 **products of type** 2	Flowtime (%)	-14,0
	Yield (%)	-2,2
	Makespan (%)	-0,2
30 **products of type** 2	Flowtime (%)	-22,0
	Yield (%)	-6,3
	Makespan (%)	10, 0
3 **products of each type with rounded process times**	Flowtime (%)	-14,0
	Yield (%)	-11,0

Table 4.1: Mean performance measure comparison for 10 general use cases of the GA and SBH for different experiments.

Another parameter that has to be taken into account is the computational time of the SBH. Experiments are conducted, and the number of processes seems to be the dominant factor for influencing the computational time. This seems logic, since the the number of disjunctive arcs, and possible scheduling orders are directly dependent on the number of processes. So there is found that the limiting behaviour for the computational time behaves as $\mathcal{O}(P^3)$, with P the number of processes. Now that the general performance of the heuristics is determined, these can now be applied to two possible use cases for the Micro Device Fab.

4.2 Real life use cases

Real life use cases for two possible Micro Device Fab systems will be simulated and the performance and robustness of these systems will be tested. The first use case is a manufacturing line for the fabrication of OLED's, with only one type of products. The second use case is a Micro Device Fab like system, for the production of multiple types of products simultaneously.

4.2.1 OLED manufacturing line

As mentioned before, the OLED manufacturing line will be developed for one type of product. This enables the possibility of designing the line for this particular product flow. An OLED is an organic led, which can be created with wet chemistry processes and printed electronics on flexible substrates to create flexible displays as shown in Figure 4.1.

Figure 4.1: OLED flexible display.

The creation of an OLED structure consists of two flows, the first is to create the base layer and the second is to finish the product. Figure 4.2 shows the machine visits and estimated process times to create the base layer.

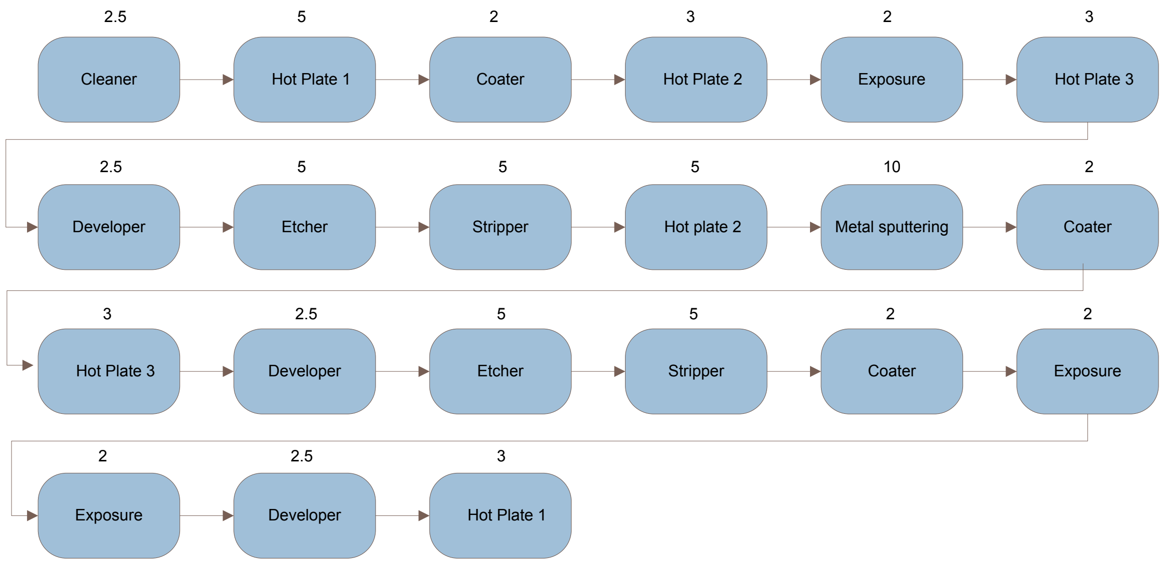

Figure 4.2: Product flow of the base layer for OLED

The flow starts with a preprocessed glass plate with an ITO (indium tin oxide) layer, which is a conductive layer. When a substrate enters the system, it first has to be cleaned and afterwards the production processes can be started. The hot plate processes require priority, and the substrates need to be retrieved within 60 seconds after the process time is finished. With this flow information, the first part of the manufacturing line for the processing of the base layer can be designed. Figure 4.3 shows the design created from the process flow.

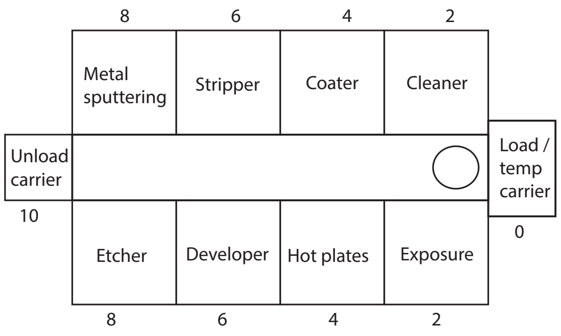

Figure 4.3: Design of the manufacturing line for the first product flow of the base layer for OLED.

The machines that follow up on each other are located as close as possible together. The hot plate procedure is a repeatedly occurring process and is therefore placed in the center of the line, and the robot is equipped with two arms. There are multiple hot plates, which are stashed above each other. On the right side there is an input buffer and a temporary buffer, and on the left side the output buffer. The size of these buffers is still infinite, because the size is dependent on the throughput of the system. The transportation times from the load/temporary carrier to the machines are shown by the numbers at the machines in Figure 4.3. The time to put or grab a product takes 8 seconds, this is assumed for all use cases. For the shifting bottleneck heuristic, this results in an average transportation time of 26 seconds that will be added to all process times. Now the second part of the process can be investigated. Figure 4.4 shows the process flow and estimated process times for the finalization of the OLED.

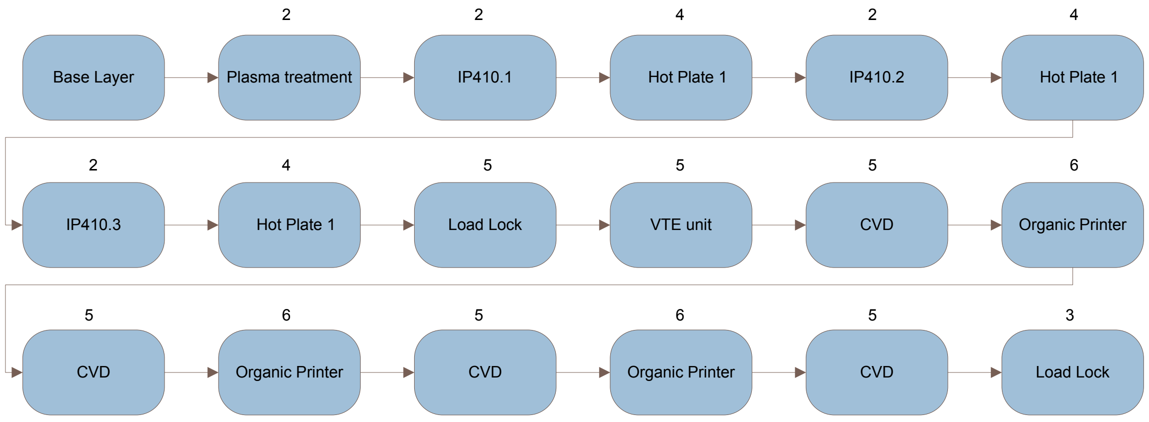

Figure 4.4: Product flow of the final layer for OLED.

For this flow also only the hot plates require priority. From this flow again a manufacturing line can be designed, which is shown in Figure 4.5.

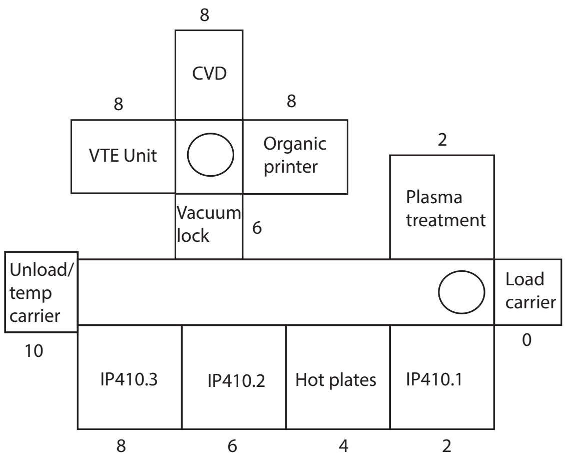

Figure 4.5: Design of the manufacturing line for the second product flow of the final layer for OLED.

The design of the second flow is slightly more complicated due to some processes that have to be performed in vacuum. Again the machines that follow up on each other are located as close as possible together. Products need to go into a vacuum lock to move to the vacuum cluster. For the real design, this vacuum cluster is equipped with a robot. However, in the simulations there is not a vacuum cluster robot, and it will be assumed that the main robot can move into the vacuum cluster. The transportation times from the load carrier to the machines are shown by the numbers at the machines in Figure 4.5. For the shifting bottleneck heuristic, this results in an average transportation time of 27 seconds that will be added to all process times. The two manufacturing lines for the base layer and final layer can be combined by making the unload carrier of the base layer the load carrier of the final layer. Figure 4.6 shows the complete OLED manufacturing line.

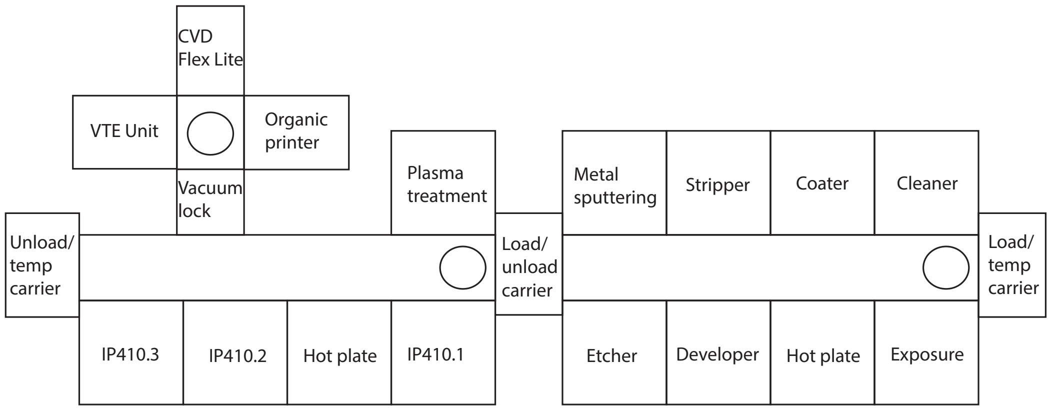

Figure 4.6: Design of the manufacturing line for the complete production of OLEDs.

The load and unload stations are now combined. This requires a special design which can move the products with a finished base layer to the other side of the manufacturing line. This could be accomplished with a carousel of carriers. After a carrier is completely filled, the carrousel turns, the next robot can retrieve the products with the base layer, and the first robot has again an empty carrier. The two parts of the manufacturing line are simulated separately, because for the real system the products do not immediately move to the other track, but start after a complete batch is finished.

4.2.2 Micro Device Fab

The second use case has the same concept as the Micro Device Fab, which is a manufacturing line for the production of different types of products. The design of the line can therefore not be optimized for one product flow. However, some processes almost always follow up on each other. For example, coating, hot plate, exposure, hot plate, developer, etcher and stripper, which are all processes from the wet cluster. By discussing the required machines and processes to be able to develop most products with a technology expert, the necessary machines are added to the manufacturing line. Figure 4.7 shows the new design of the Micro Device Fab concept.

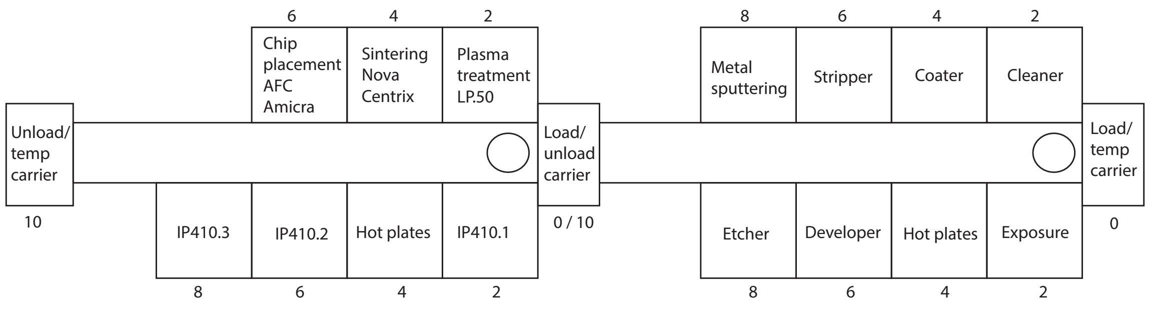

Figure 4.7: Design of the manufacturing line for the Micro Device Fab concept.

The transportation times from the load carrier to the machines are shown by the numbers at the machines in Figure 4.7. For the shifting bottleneck heuristic, this results in an average transportation time of 27 seconds for the left robot track, and 25 seconds for the right robot track, that will be added to all process times. Because the exact products that will be manufactured in the Micro Device Fab are unknown, some estimated flows are designed from real life examples. The first flow represents a similar flow as for the production of an OLED and is shown in Figure 4.8. The double dash shows the transition from the right side of the track to the left side.

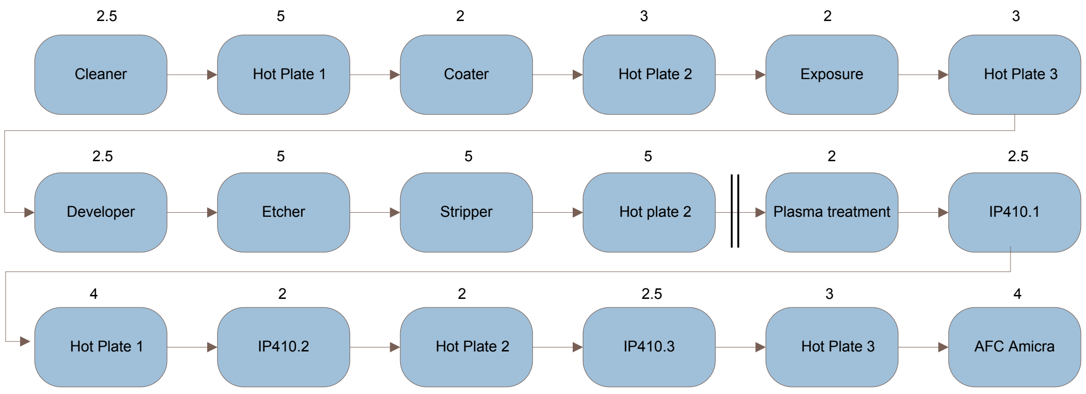

Figure 4.8: First flow for the production of OLED like structures.

The second flow is based on the production of the FlexSmell, this is a smart food label that was explained in the Introduction. Figure 4.9 shows a possible flow for the production of the FlexSmell.

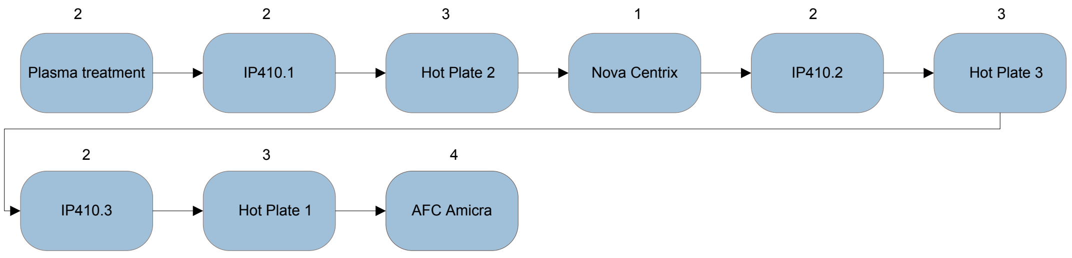

Figure 4.9: Second flow for the production of FlexSmell like structures.

The third flow is for the production of a Lab on Chip, which is a micro fluidic channel system connected to a chip. A lab on chip functions as a small laboratory in which chemical analysis can be performed, e.g. a bloodtest. The production flow of a Lab on Chip is shown in Figure 4.10.

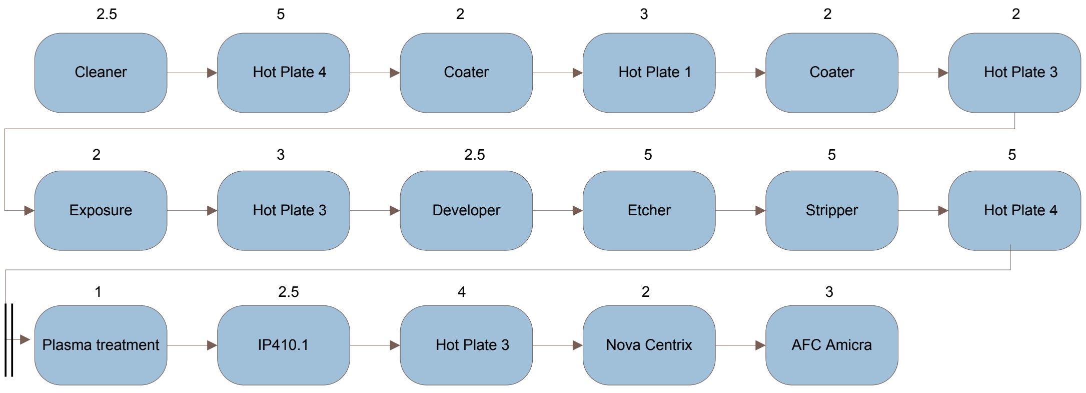

Figure 4.10: Third flow for the production of a Lab on Chip like structure.

Now that the lay out and flows of the different use cases are known, these can be implemented into the simulation models. The two parts of the manufacturing line are also simulated separately, because for the real system the products do not immediately move to the other track, but start after a complete batch is finished.

4.2.3 Results

Simulations are conducted with the above mentioned use cases. For each use case the following experiments are conducted to compare the performance of the two heuristics:

- for deterministic process times.
- for variable process times, with a coefficient of variation of 0.11, and 0.29.
- for machine failure, the bottleneck machine stops once for 10, and 20 minutes

For the experiments with variable process times, during simulation the process time is determined by taking a sample of a gamma distribution with a mean of the deterministic process time. Due to this variability, the results are an average of multiple simulations for a reliable solution within the 95% interval. For the machine failures, the bottleneck machine is stopped once during a predetermined process on that machine. The performance of a system is measured by four key indicators: the total completion time of a batch of jobs (makespan), the average time a job is in the system (flowtime), the amount of time that the busiest machine (bottleneck machine) is used (utilization), and the percentage of correct products (yield). Table 4.2 shows the performance of the greedy algorithm (GA) and shifting bottleneck heuristic (SBH) for each use case and the different experiments. For the OLED use case the makespan is shown for creating 15 products, which are all of the same type. For the Micro Device Fab 7 jobs of the OLED type, 8 jobs of the lab on chip type, and 15 jobs of the flexsmell type are created, so 30 jobs in total. Both sides of the robot tracks of the manufacturing lines are simulated separately, because for the real system the products do not immediately move to the other track, but start after a complete batch is finished

The main performance indicator is the makespan. The results of the makespan show that for deterministic process times, the SBH outperforms the GA for all use cases. For the first OLED use case the SBH performs almost 6% better, in the second OLED use case the difference is 2%. For the first use case of the Micro Device Fab, the SBH performs even 13 % better, but for the second use case only a 4% difference is achieved. Hence, we can conclude that the difference in performance is highly dependent on the flows and process times. Most process times used for these use cases are multiples of each other, which enables the greedy algorithm to have a tighter follow up of products on a machine by "accident". This results in a smaller difference between the two heuristics compared to the results of Section 4.1. For the experiments with variable process times there can be seen that the performance of the SBH decreases quicker than for the GA, when $c_v = 0.11$ the makespan becomes almost equal for both heuristics, but with a variance of $c_v = 0.29$ the GA outperforms the SBH. This difference is expected, since the SBH computes an optimal schedule for deterministic process times in advance, and the GA reacts to the behaviour of the system. For the experiments with a machine failure of 10 and 20 minutes, the difference in makespan between the SBH and GA remains almost the same. When there is a machine failure, the makespan of the SBH increases with the failure time, because the predetermined scheduling does not adjust and therefore has to wait until the scheduled job is finished and can continue. The GA improves slightly compared to the SBH, because the GA continues with the processing on the other machines, but cannot gain much because the processes on the failed bottleneck machine still have to be performed.

		OLED 1		OLED 2		MicroFab 1		MicroFab 2	
Experiment	**Performance measure**	**GA**	**SBH**	**GA**	**SBH**	**GA**	**SBH**	**GA**	**SBH**
Deterministic process times	Makespan (hours)	4.26	4.10	6.59	6.48	2.55	2.33	2.84	2.83
	Flowtime (hours)	2.48	2.46	4.53	4.09	1.04	1.23	1.62	1.60
	Utilization (%)	58.7	61.0	86.0	87.8	52.3	57.3	63.4	63.6
	Yield (%)	100	93.3	100	86.7	100	94.1	100	98.7
Variable process time $c_v = 0.11$	Makespan (hours)	4.43	4.31	6.79	6.66	2.49	2.43	2.85	2.95
	Flowtime (hours)	3.20	3.17	4.59	4.22	1.64	1.70	1.66	1.68
	Utilization (%)	56.5	58.0	83.4	85.1	53.6	54.8	63.1	61.1
	Yield (%)	100	88.9	100	77.8	100	97.1	100	96.0
Variable process time $c_v = 0.29$	Makespan(hours)	4.56	4.81	7.09	7.09	2.58	2.72	2.97	3.26
	Flowtime (hours)	3.32	3.57	4.78	4.51	1.69	1.91	1.72	1.87
	Utilization (%)	54.8	52.0	79.9	80.0	51.6	49.0	60.7	55.3
	Yield (%)	100	80.0	100	73.3	100	89.7	100	90.5
Machine failure, 10 minutes	Makespan (hours)	4.38	4.21	6.83	6.70	2.59	2.46	2.90	2.95
	Flowtime (hours)	3.13	3.10	4.79	4.27	1.70	1.73	1.66	1.67
	Utilization (%)	57.1	59.3	83.0	84.6	51.6	54.1	62.1	60.9
	Yield (%)	100	93.3	100	80.0	100	94.12	100	96.0
Machine failure, 20 minutes	Makespan (hours)	4.49	4.38	6.99	6.87	2.70	2.63	3.07	3.12
	Flowtime (hours)	3.20	3.26	4.93	4.41	1.78	1.87	1.77	1.76
	Utilization (%)	55.7	57.1	81.0	82.5	49.4	50.7	58.7	57.7
	Yield (%)	100	93.3	100	80.0	100	94.1	100	96.0

Table 4.2: Performance measure for the the different use cases for different experiments.

Now the other performance indicators will be further reviewed. No clear difference can be found between the flowtimes of the two heuristics. For both heuristics the flowtime increases when the process time variation is increased and when there is a machine failure. The utilization of the bottleneck machine is a directly related to the makespan, and therefore shows the same results as for the makespan. The yield shows the percentage of priority products that have been retrieved within the required time of 60 seconds after a priority process step is finished. This requirement is always met for the GA, this is not the case for the SBH. The amount of incorrect products in the SBH increases for variable process times and remains the same or increases for machine failure. Dependent on the importance of this rule being satisfied, there can be chosen between the shortened makespan or the product equality.

4.3 Summary

From the experiments the performance of the design rules and behaviour of the scheduling could be investigated, the following insights were found:

- The SBH performs optimal for a small number of products, with a large variance in product flow and deterministic process times.
- The advantage of the SBH over the GA disappeared for use cases with a large number of products, with a single process flow and process times that are multiples of each other.
- The performance of the SBH decreases more, compared to the GA, when there is a variance in process times or a machine drop out, and is therefore less robust.

Chapter 5

Conclusion

After investigating the prototype of the Micro Device Fab, we concluded that the current system performance was poor. There was only a 40% bottleneck machine usage, and deadlocks could occur while processing multiple different product flows simultaneously. With this knowledge, new system design rules could be formulated, starting with the system architecture. When equipping the robot with a dual arm instead of a single arm, the bottleneck machine usage could be increased to 82%. For the other design variables (e.g. buffers, alignment, etc.), design rules were formulated based on research and observations. For the scheduling of the jobs, two options, the greedy algorithm (GA) and the shifting bottleneck heuristic (SBH), were investigated. These two heuristics were adjusted to the job shop problem of the Micro Device Fab, which has multiple types of flows simultaneously, multiple machine visits to the same machine, robot transportation times, and priority.

Afterwards, the determined design rules were applied to use cases to evaluate their performance. This showed that the SBH performs optimal for a small number of products, with a large variance in product flow and deterministic process times. This resulted in an average 14% smaller makespan for the SBH compared to the GA, of which 4% is achieved by a newly developed adjustment in the SBH that decreases time between process steps. However, for use cases with a large number of products, with a single process flow, and process times that are multiples of each other, the advantage of the SBH over the GA disappeared. The robustness of the GA was found to be better than for the SBH, because the performance of the SBH decreased more rapidly when a variance in process times or a machine drop out was applied. For all experiments the SBH has a larger or equal flow time, and a smaller or equal yield compared to the GA. This yield represents the number of correct products, and thus if the yield is lower the product equality is lower. The utilization of the bottleneck machine is directly related to the makespan, and therefore shows the same results. Dependent on the importance of each performance measure, there can be chosen between the shortened makespan, flow time, or the product equality.

With the use of simulation the performance has been measured, and has shown large improvements in system performance compared to the prototype. With the developed design rules new Micro Device Fab systems can be created. In the future, predictions of the system behaviour and performance can be made for both the prototype as for new designs with the use of simulation models. The two scheduling heuristics both showed a good performance, the SBH did outperform the GA for deterministic process times, however, does not cope well with system variations and has a lower yield.

Further research and Recommendations

To improve the SBH, the schedule could be recalculated during the product processing, this would improve the performance for long runs and system variations. Therefore, we can not conclude what the evidently optimal heuristic is for Micro Device Fab systems, because this depends on the application and the importance of the product equality. However, due to the research application of the Micro Device Fab, a 10% makespan improvement is probably less important than the product equality, and therefore the GA seems more suitable. Another application for the SBH could be the smart scheduling of machine clusters, which are complicated flows with a smaller number of machines, which require a high machine utilization. Eventually, a heuristic should be incorporated into the real control software, which can be implemented into the newly designed Micro Device Fab systems.

Bibliography

[1] Introduction to secs/gem - beginner's guide. *http://www.insphere.com.sg/index.php/component/content/article/38-tutorials/67-beginner-guide-to-secsgem.html*. 53

[2] Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401, 1988. 17, 19

[3] Egon Balas, Jan Karel Lenstra, and Alkis Vazacopoulos. The one-machine problem with delayed precedence constraints and its use in job shop scheduling. *Management Science*, 41(1):94–109, 1995. 30

[4] Danick Briand. Flexsmell. *http://samlab.epfl.ch/page-103922-en.html*. 2

[5] Meyer Burger. High-end solutions, for high-tech industries. *http://www.meyerburger.com/en/*. 1

[6] Meyer Burger. Printed electronics. *http://www.meyerburger.com/en/products-systems/competences/coating/printed-electronics/*. 2

[7] DoMicro BV. Automated microsystem technology. *http://domicro.nl/*. 1

[8] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, and Marco Trubian. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994. 17

[9] Stephane Dauzere-Peres. The one-machine sequencing problem with dependent jobs. *Computers & industrial engineering*, 25(1):235–238, 1993. 30

[10] Ebru Demirkol, Sanjay Mehta, and Reha Uzsoy. A computational study of shifting bottleneck procedures for shop scheduling problems. *Journal of Heuristics*, 3(2):111–137, 1997. 17, 19

[11] Industry Directions. Enhancing profitability through structured manufacturing. *http://swiftcourse.com/TodaysManufacturing.html*. 53

[12] electronics LTD EPAK. Hybrid electronics equipment. *http://www.epakelectronics.com/hybridelectronics.htm*. 1

[13] A.J.H. Frijns. Lecture notes: Micro- and nanotechnology. *Vehicular Technology, IEEE Transactions on*, 2013. 3

[14] Philip Ivens and Marc Lambrecht. Extending the shifting bottleneck procedure to real-life applications. *European Journal of Operational Research*, 90(2):252–268, 1996. 17

[15] Selmer Martin Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1):61–68, 1954. 17

[16] Scott J Mason, John W Fowler, and W Matthew Carlyle. A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, 5(3):247–262, 2002. 17, 19, 28

[17] Lars Mönch and Jens Zimmermann. A computational study of a shifting bottleneck heuristic for multi-product complex job shops. *Production Planning and Control*, 22(1):25–40, 2011. 17, 28

[18] Novacentrix. Photonic sintering. `https://www.novacentrix.com/`. 2

[19] Seyda Topaloglu and Gamze Kilincli. A modified shifting bottleneck heuristic for the reentrant job shop scheduling problem with makespan minimization. *The International Journal of Advanced Manufacturing Technology*, 44(7-8):781–794, 2009. 17, 28, 29

[20] Peter JM Van Laarhoven, Emile HL Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125, 1992. 17

Appendix A

Operational and system structure

Production is managed in three main sections, the Enterprise Resource Planning (ERP), the Manufacturing Execution Systems (MES), and the process control. The ERP exists of all the financial, marketing and resource operations. MES exists of namely manufacturing strategies, manufacturing planning and control systems and information technology. The process control is the control of the production line and the operators. MES is the intermediate step between the Enterprise Resource Planning (ERP) system, and the Device control systems, which is shown in Figure A.1 [11].

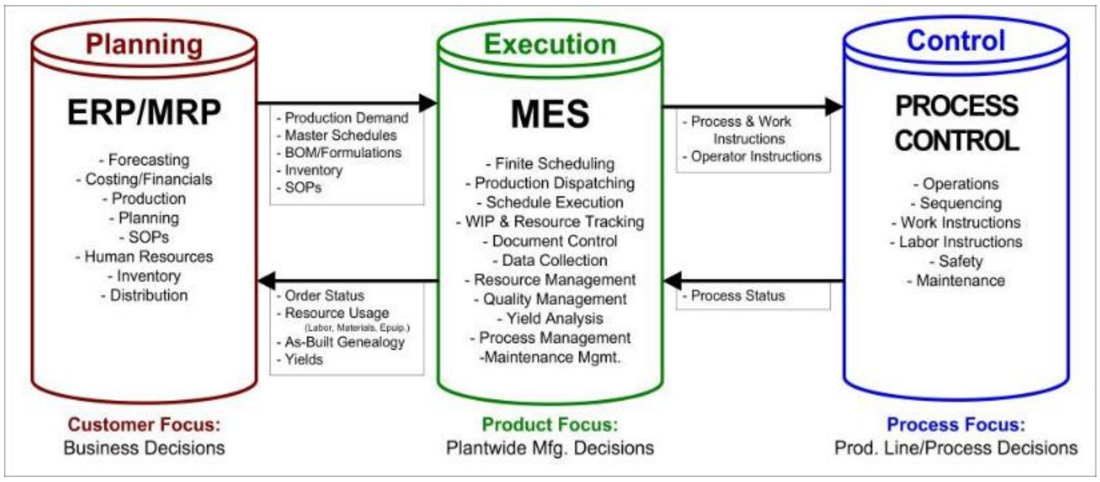

Figure A.1: Production structure.

A.1 Device Control system structure

The robot track and the machines communicate with the MCS via a protocol developed by Solar Semi. Besides that, to ensure compatibility between the communication of the machines and MCS with the MES system, a standard protocol is used. This is SECS/GEM (Generic Model for Communications and Control of Manufacturing Equipment) [1], which is developed for the Semiconductor industry and is used in the Micro Device Fab. This connectivity standard describes how equipment shall communicate with a MES, such that for example measurement data can be collected, variables can be changed and flows for products can be selected. However, this option is not yet implemented, because there is no MES yet and therefore flows are still selected in the MCS.

As was shown in the previous section production is constructed into three sections: Enterprise Resource Planning (ERP), manufacturing execution system (MES), and the Device control systems. For the MicroDeviceFab first there has to be determined how far the Device control system is developed and what is needed to function properly. The parts of the Device Control system that are found relevant for the MicroDeviceFab will be discussed on how much these are developed already. This investigation resulted in a schematic structure of the different devices and the connections between these devices. Figure A.2 shows the Micro Device Fab structure that was found.

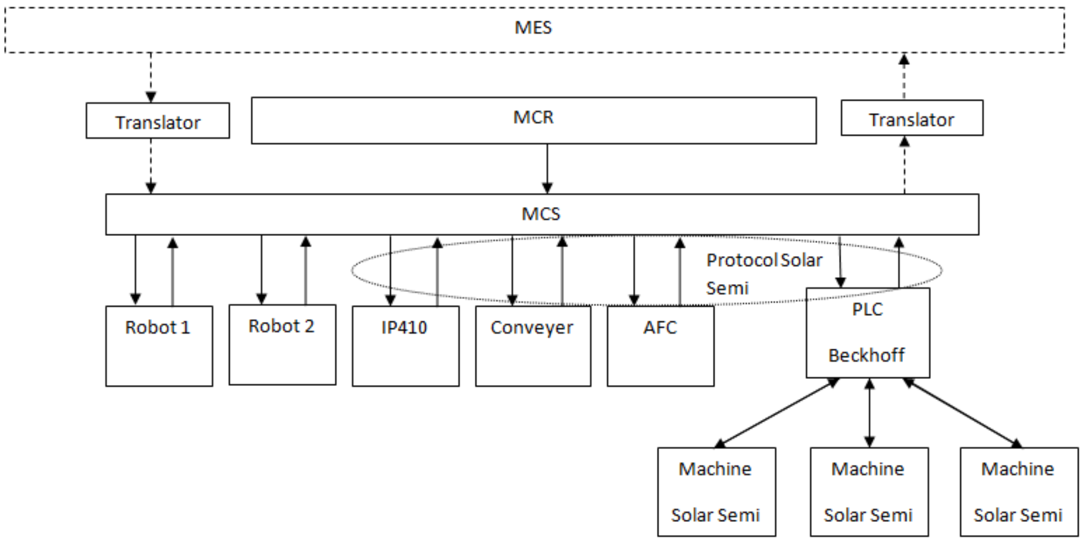

Figure A.2: Process control structure and connection to the MES.

The structure exists of the MCR software, in which flows can be defined, these flows can be loaded into the control and interface software MCS, which was mentioned before. The MCS is again connected to the two different robot tracks, conveyer, non Solar Semi machines and a PLC which is connected to the Solar Semi machines. The communication of the conveyer, machines and PLC follows the protocol defined by Solar Semi. The two robot tracks communicate via ethernet, however, the protocol that is used is unknown. Currently, there is no MES, but if this would be constructed a data connection should be generated between the MES and the MCS. The MCS can not be connected to the MES directly, therefore a protocol translator should be implemented in between.

When a carrier is reset, by lifting the carrier and triggering the sensor, a product flow can be started. First the user has to specify whether the carrier is sender or receiver, so if it sends the substrates into the system or if it receives substrates from the system. Afterwards, the number of substrates in the carrier have to be entered. To this complete batch of substrates a sequence of process steps (flow) can be assigned, which is chosen from a list of flows, how these flows are created will be explained in the next paragraph. In each flow it is described whether the carrier is both the sender and the receiver or only the sender. If the carrier is only the sender, the other carrier should be called the receiver and only one type of flow is possible. If the carrier is both the sender and the receiver, it is possible to start two different types of flows from the two different carriers.

With the program MCR, the product flows can be created. In these flows the process steps of a product are described, e.g.: LP50 recipe 2, IP410 recipe 1, LP50 recipe 1, finish. A recipe exist of a pattern and properties, which are saved and created on the computer of the specific machine. These computers are all connected to the main computer, which allows the possibility of choosing a recipe from a list of recipes for each machine to define a flow. After a flow is described, the flow characteristics are described, e.g. the filling strategy of the carrier (sender/receiver). After a flow is finished it can be saved in the MCR and can then be loaded into the MCS.

Appendix B

Simulation model current software

A simulation model will be created in Chi to verify the determined decision rules of the current Micro Device Fab.

B.1 Model structure

First the structure of the system will be described. A schematic view of the system that will be modeled is shown in Figure B.1. The numbers refer to the different parts of the system, which eventually will be used to create flows.

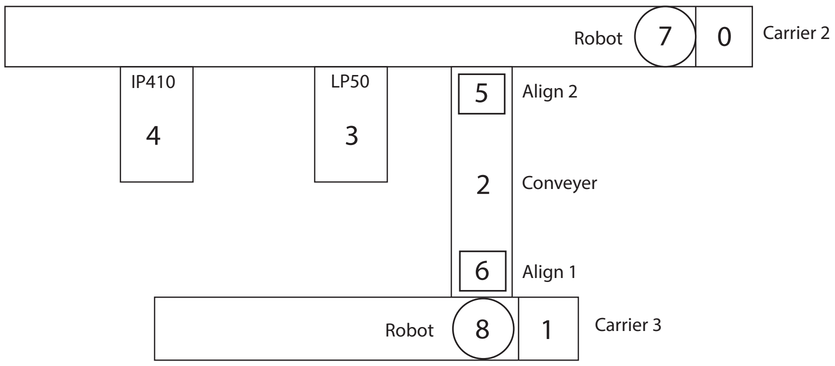

Figure B.1: Schematic view of the system.

The model exists of two robots or handlers, which can transport substrates through the system, a conveyer belt to connect the two robot tracks, two alignment locations at either side of the belt and two machines. This system can be translated into a simulation model by dividing the system parts in different processes which can be connected to each other with channels. This structure is schematically described in Figure B.2.

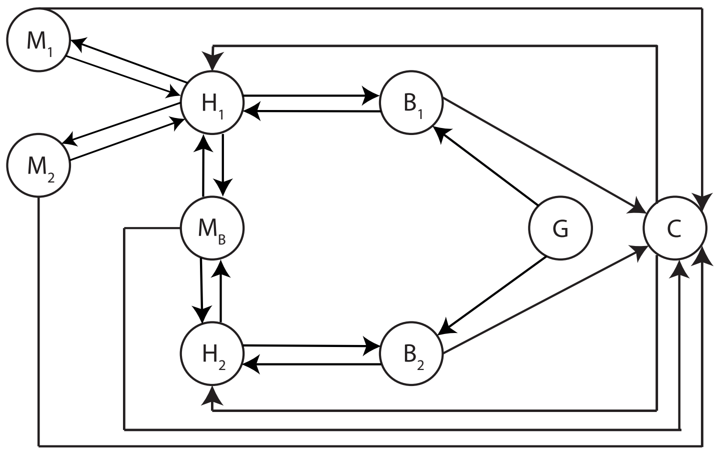

Figure B.2: Structure of the processes and channels of the simulation model.

Figure B.2 shows the different processes and their connection through channels. It starts with the generator (G), which sends the specified products to the carriers, or also called buffers (B_1 and B_2). Because there are two buffers the generator has two channels through which products can be send. In the model a product list is defined, in which for each buffer a list of products and their type is described. The generator will keep sending products until all products in the product list are send. The buffers can receive products from the generator and add them to a list. When this list is non empty a message is send to the controller (C) and a product can be send to one of the handlers (H_1 and H_2). The handlers receive from the controller the product number that they have to move. When this number is received it can retrieve the product at either a buffer or a machine (M_1 or M_2). If the product is received from the buffer, first the handler has to move to the buffer, afterwards, the handler moves to the alignment station and aligns the product and finally the product is moved to the next machine. This is similar for products that are received from machines. Products carried by the handler can be moved to machines, however, this will only happen when the controller sends this message to the handler. The machines also send messages to the controller, such that the state of the machines is known. The two handlers can also exchange products, this can be done via the conveyer belt (M_B), which is modeled as a machine.

Controller

The controller controls the system and decides the movements of the handlers and therefore the product flow. The controller receives the status of the buffers and the machines, such that it is known which machines are available and what products need to be processed. With this information the controller should decide what product should be served. The goal of the model is to mimic the current scheduling. Therefore, the decision rules that were identified for the real system are used:

- Substrates are put or moved in the system when the next machine is available, which allows parallel processing.
- There is one exception: when a product has to move from one robot track to the other via

the conveyer, the substrate can only be moved to the conveyer when the next machine is available, due to the conveyer being one directional.

These decision rules are implemented into the controller of the simulation model and are key in deciding which product should be served. The products in each buffer are served following the first come first serve rule, because the current system only allows one type of flow for all products in the same buffer.

To decide which product should be served, the controller requires some information. The first option is to request information from the buffer, this information is the product that is waiting in the buffer. The second option is to request information of the machines at the beginning and after a product is finished. The information of the states of the machines and buffers is saved in a list. If there is a zero at the location of the machine number in the list, than the machine is available. The state of the processes together with the decision rules results in the controller that should mimic the real control software, which will be verified.

B.2 Handling and Process times

For the eventual modeling of the Micro Device Fab the handling and process times have to be determined. The handling times of the substrates can be determined by observing the Micro Device Fab, which includes: the handling times of the robot, the process time of the alignment procedure, and the transportation time of the conveyer belt. The process time of the machines depends on the kind of job that has to be performed, and is therefore variable. Table B.1 shows the handling times that were identified.

Action	**Time [sec]**	**Remarks**
Handler 8: move from init to carrier	0	Carrier at init position
Handler 8: move from init to alignment	6	
Handler 8: move from alignment to LP50	4	
Handler 8: move from alignment to IP410	8	
Handler 12: move from init to carrier	0	Carrier at init position
Handler 12: move from init to alignment	0	Alignment at init position
Handler 12: move from init to belt	0	Is incorporated in the belt time
Time of putting or taking out a substrate from a machine, the belt or carrier	8	
Move belt from align 1 to align 2	9	
Time of aligning	28	

Table B.1: Transportation times